Akaike Technologies
Assignment: News Summarization and Text-to-Speech Application
Devam Tanwani

Step 1: Project setup

- Initializing a Github repository-

I created a github repository at the start itself to track changes, text certain code snippets and revert back to the previous code in case of any blocks.

- This made it easier to organise and keep track of the changes in the original code as well as maintain the progress I have made using commit messages.
- Installing dependencies-

Since this project would be needing many third party libraries such as Streamlit for app development, beautifulsoup4, gtts for text to speech, requests. FastAPI etc.

- All all the required dependencies were written in the requirements.txt file and installed together.
- Organize the Project Structure-

This project structure is as follows-

```
news-summarizer/
├── app.py                # Web UI (Streamlit)
├── api.py                # API backend (FastAPI)
├── utils.py              # Helper functions (scraping, sentiment analysis, TTS)
├── requirements.txt      # Dependencies
├── README.md             # Documentation
```

Step 2: News Extraction

- I used NewsAPI.org to automatically extract news for a given company by searching for news related to the company name.
- Although the API returns structured data (titles and summaries), I only extracted the news URLs to comply with the assignment's requirement to use BeautifulSoup for scraping.
- Using these non-JavaScript URLs, I employed BeautifulSoup to parse the HTML and extract the article title (from the <h1> tag) and a summary (from the <p> tags).
- For summarization, I integrated the Gemini API to generate a concise summary of the scraped content.
- I looped through 10 article URLs to create a collection of news articles with their title, summary, and URL.

Step 3: Sentiment Analysis

- For sentiment analysis, I used the NLTK VADER sentiment analyzer, since it is best for short text input like summaries using pre-trained sentiment lexicons. It is also pretty fast and lightweight.
- Once the summaries were extracted from each article, I analyzed the tone by classifying each summary as Positive, Negative, or Neutral. This process provided a basis for understanding the media's tone regarding the company and enabled further comparative analysis.

Step 4: Comparative Analysis

- This step involves comparing the overall sentiment of the 10 articles to derive insights on how the company's news coverage varies.
- I aggregated the sentiment results (counts of Positive, Negative, and Neutral articles) from all 10 articles.
- Additionally, using the google ai API again, I printed out at least 2-3 unique topics from each article
- Using the sentiment results, the final output includes a structured Comparative Sentiment Score section that shows if the coverage of the 10 articles combined are positive for the company, negative, or have a mixture of both positive and negative article sentiments.

Step 5: Text-to-Speech (TTS)

- The objective of this step was to convert the summarized news content and the final overall sentiment analysis into Hindi speech.
- I used the gTTS library to generate audio output.
- Since the extracted and summarized text is in English, I integrated a translation step (using googletrans) to convert it into Hindi before TTS.
- Separate audio files were generated for each article summary and for the overall final sentiment analysis.
- These audio files are then referenced in the final JSON output, allowing users to listen to the Hindi summaries.

Step 6: User Interface (Streamlit)

- I developed a web UI using Streamlit.
- Streamlit is used to create a clean and modern UI and it works well for text-based applications.
- The UI contains an input field for the company name.
- Upon submission, the UI calls the backend API to fetch the news data.

- The results are displayed in a clean format showing:
    - Article titles, summaries, sentiment, and key topics.
    - Audio players to listen to each article's Hindi summary.
    - A comparative sentiment analysis section showing overall sentiment distribution.
- This interface makes it easy for users to quickly understand the news landscape for any company.

Step 7: API Development (FastAPI)

- I built a backend API using FastAPI that encapsulates all the processing (news extraction, summarization, sentiment analysis, TTS).
- The frontend (Streamlit UI) can communicate with the backend (news extraction, summarization, sentiment analysis, TTS).
- The API exposes endpoints that accept a company name and return a structured JSON response with:
    - The list of 10 news articles (each with title, summary, sentiment, topics, and audio file reference).
    - A comparative sentiment analysis section.
    - A final overall sentiment analysis along with a corresponding Hindi audio file.
- The API and the Streamlit UI are decoupled, making the architecture modular and scalable.

Step 8: Deployment (Hugging Face Spaces)

- I created a Dockerfile to containerize the application.
- The Dockerfile is set up to install all dependencies and start both the FastAPI backend (using Uvicorn) and the Streamlit frontend concurrently.
- I then pushed the project to a GitHub repository, which I have maintained throughout the project.
- On Hugging Face Spaces, I connected to the GitHub repository.
- API keys and secrets (such as the Gemini API key) are managed via Hugging Face Secrets instead of a .env file for security.
- Finally, I deployed the application on Hugging Face Spaces, making it accessible via a public URL.

Conclusion

The final application meets all assignment requirements by integrating:

- **News Extraction:** Automatically fetching 10 unique news articles related to a given company, scraping the title and summary using BeautifulSoup.
- **Sentiment Analysis:** Classifying the summarized content as Positive, Negative, or Neutral.
- **Comparative Analysis:** Aggregating sentiment data and comparing key coverage differences and topics across articles.
- **Text-to-Speech:** Converting the summaries and overall analysis into Hindi audio using gTTS (with translation).
- **User Interface:** Providing a clean, interactive UI with Streamlit.
- **API Development:** Exposing all functionalities through a FastAPI backend.
- **Deployment:** Hosting the application on Hugging Face Spaces for public access.
- **Documentation:** All steps and configurations are well documented in the README, along with detailed setup instructions and explanations of the models and libraries used.