**Submitted by: Shaik fayaz ahmed (fayazskyf123@gmail.com)  
  
News Summarization & Sentiment Analysis Application Documentation**

**Note: The code provided provides only the articles with summaries within 5 webpages (less than 10). it is done to save time in computing as it takes 10-15mins to run the code. if you need 10 article you can just remove the page variable in the code and proceed as is.**

**1. Project Setup**

**Prerequisites**

Ensure that **Python 3.8+** is installed on the system.

**Step 1: Install Required Dependencies**

Run the following command to install all required libraries:

pip install -r requirements.txt

If requirements.txt is not present, create it with the following content:

fastapi

uvicorn

streamlit

requests

beautifulsoup4

sumy

transformers

torch

vadersentiment

gtts

googletrans==4.0.0-rc1

**Step 2: Start the Backend (FastAPI Server)**

Run the following command to start the FastAPI backend:

uvicorn api:app --reload

This will start the backend server at http://127.0.0.1:8000/.

**Step 3: Start the Frontend (Streamlit Application)**

Open a new terminal and run:

streamlit run app.py

This will launch the web application where users can enter a company name and fetch the results.

**2. Model Details**

**News Summarization Model**

* The **LSA (Latent Semantic Analysis) Summarizer** from **Sumy** is used to generate concise summaries from extracted news articles.
* It works by identifying **important sentences** based on topic modeling.

**Sentiment Analysis Model**

* **VADER (Valence Aware Dictionary and sEntiment Reasoner)** is used for sentiment analysis.
* It assigns a **compound score** to determine whether a sentence is **positive, negative, or neutral**.

**Topic Extraction Model**

* A **zero-shot classification model** (facebook/bart-large-mnli) from **Hugging Face Transformers** is used to classify articles into topics.
* This allows the system to categorize news into predefined labels such as **Finance, Technology, Healthcare, Business, etc.**.

**Text-to-Speech (TTS) Model**

* The **Google Translate API** is used to translate English text into Hindi.
* The translated text is then **converted to speech using gTTS (Google Text-to-Speech)**.

**3. API Development**

The backend is developed using **FastAPI** to handle news extraction, sentiment analysis, and text-to-speech generation.

**API Endpoints**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /news/?company=Tesla | GET | Fetches news articles related to the given company, along with their summaries, sentiment scores, and topics. |
| /tts/?company=Tesla | GET | Generates Hindi speech from the summarized news and returns the audio file. |

**How to Access APIs using Postman**

1. **Start the FastAPI server:**
2. uvicorn api:app --reload
3. **Open Postman** and send a **GET request** to:
4. http://127.0.0.1:8000/news/?company=Microsoft
   * This will return a structured JSON response containing extracted news, sentiment analysis, and topic classification.
5. **To generate Hindi speech output, send a GET request to:**
6. http://127.0.0.1:8000/tts/?company=Microsoft
   * This will generate and return a **Hindi audio file (MP3 format)** summarizing the sentiment analysis.

**4. API Usage (Third-Party Integrations)**

**External APIs Used**

| **API** | **Purpose** | **Integration Details** |
| --- | --- | --- |
| **Bing News Search** | Fetches news articles related to the given company. | Implemented using requests and BeautifulSoup for web scraping. |
| **Google Translate API** | Translates English summaries into Hindi. | Used within the TTS function before generating speech. |
| **gTTS (Google Text-to-Speech)** | Converts translated Hindi text into speech. | Generates an MP3 file that can be played in the frontend. |

**5. Assumptions & Limitations**

**Assumptions**

1. The system assumes that **Bing News Search** will return at least **10 relevant articles** for the given company.
2. The **LSA summarization model** is assumed to generate **concise and meaningful summaries** for the articles.
3. **VADER sentiment analysis** is assumed to work well for short news summaries, though it is optimized for social media text.

**Limitations**

1. **News Extraction Limitations**
   * The scraper only extracts **non-JavaScript** news content.
   * If an article requires JavaScript to load, it may not be fetched correctly.
2. **Sentiment Analysis Limitations**
   * The **VADER sentiment model** is **rule-based** and may not always capture **complex sentiment** in financial or legal news.
3. **Topic Classification Limitations**
   * The **zero-shot classification model** can only classify topics based on **predefined labels** and may not work well for **unknown categories**.
4. **TTS Limitations**
   * The **Google Translate API** may not always provide perfect **Hindi translations**.
   * **gTTS** uses **online processing**, so an **internet connection is required** for generating speech output.