**News Summarization and Sentiment Analysis with Hindi TTS**

**Introduction**

This project aims to provide a comprehensive news summarization and sentiment analysis system for a given company. It extracts news articles from BBC News, performs sentiment analysis, identifies key topics, and converts the summarized content to Hindi speech using Text-to-Speech (TTS). The system is built using FastAPI for the backend and Streamlit for the frontend.

**Project Files**

The project consists of the following files:

1. **Requirements.txt:** Contains the necessary Python libraries and dependencies for the project.
2. **api.py:** The backend API built using FastAPI to handle news analysis and TTS generation.
3. **app.py:** The frontend web application built using Streamlit to interact with the API.
4. **utils.py:** Utility functions for news scraping, sentiment analysis, topic extraction, and TTS.
5. **hindi\_summary.mp3:** The generated audio file containing the Hindi summary of news articles.

**Requirements (requirements.txt)**

The project requires the following libraries:

1. **requests:** For sending HTTP requests to web servers.
2. **bs4 (BeautifulSoup):** For web scraping and HTML parsing.
3. **nltk (Natural Language Toolkit):** For sentiment analysis and text processing.
4. **gtts (Google Text-to-Speech):** For converting text to speech in Hindi.
5. **deep\_translator:** For translating text to Hindi.
6. **fastapi:** For creating the backend API.
7. **uvicorn:** For running the FastAPI server.
8. **streamlit:** For building the frontend user interface.
9. **pydantic:** For data validation and parsing in FastAPI.
10. **textblob:** For sentiment analysis and text processing.
11. **os:** For interacting with the operating system and file management.

To install all dependencies, run:

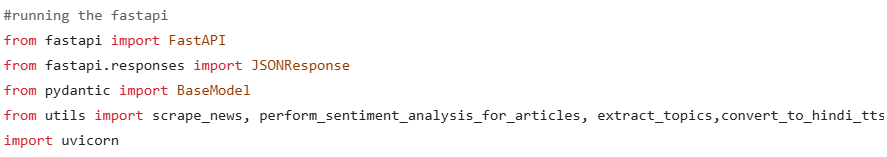
**pip install -r Requirements.txt**

**Backend Explanation (api.py)**

The backend is powered by FastAPI and serves as the API layer to handle requests and responses. It provides endpoints to perform sentiment analysis and generate Hindi TTS audio files.

**Code Walkthrough**

**Importing Libraries and Functions:**

Imports necessary libraries for API creation and response handling.

Uses utility functions for news scraping, sentiment analysis, topic extraction, and TTS.

**Creating the FastAPI App:**

app = FastAPI()

Initializes a new FastAPI application instance.

**Health Check Endpoint:**

@app.get("/")

def home():

return {"message": "News Sentiment Analysis API is running!"}

->Provides a simple endpoint to verify that the server is running.

**Data Model Definition:**

class CompanyRequest(BaseModel):

company\_name: str

->Uses pydantic for defining the structure of incoming requests.

**News Analysis Endpoint:**

@app.post("/analyze")

def analyze\_news(request: CompanyRequest):

->Accepts a company name and returns sentiment analysis and Hindi TTS.

**Generating the Response:**

Scrapes news articles and performs sentiment analysis.

Extracts topics from the summary of each article.

Converts the content to Hindi TTS.

Constructs and returns a structured JSON response.

**Running the API Server:**

if \_\_name\_\_ == "\_\_main\_\_":

uvicorn.run(app, host="0.0.0.0", port=8000)

Runs the FastAPI server using Uvicorn.

To start the server, use the following command:

🡪python -m uvicorn api:app --host 0.0.0.0 --port 8000 --reload

**Frontend Explanation (app.py)**

The frontend is built using Streamlit to provide a user-friendly interface for interacting with the API.

**Importing Libraries:**

import streamlit as st

import requests

Uses Streamlit for the web UI and requests to interact with the backend API.

**Page Configuration and Title:**

st.set\_page\_config(page\_title="News Analysis", layout="wide")

st.title("News Summarization & Sentiment Analysis")

Sets up the page title and layout for the Streamlit app.

**User Input for Company Name:**

company\_name = st.text\_input("Enter the Company name:")

->Provides a text input field for entering the company name.

**Analyze Button and API Call:**

if st.button("Analyze News"):

response = requests.post("http://127.0.0.1:8000/analyze", json={"company\_name": company\_name})

->Triggers news analysis when the button is clicked.

->Calls the backend API and gets the response.

**Displaying Results:**

->Shows the title, summary, sentiment, and topics of each news article.

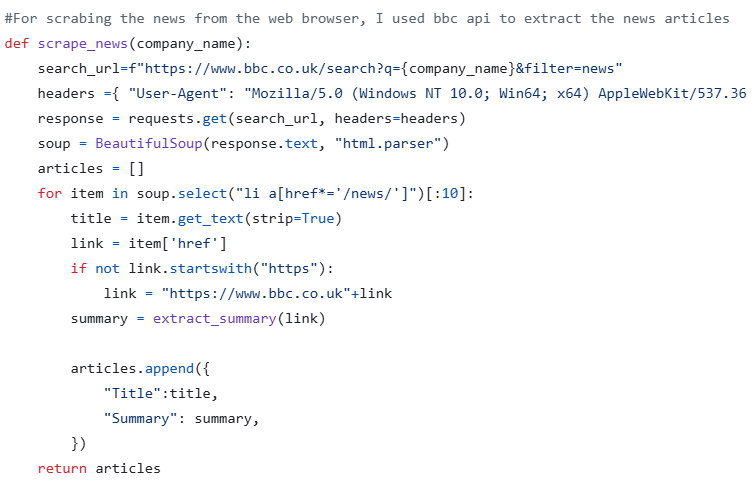
->Plays the Hindi TTS audio if available.

**Utility Functions (utils.py)**

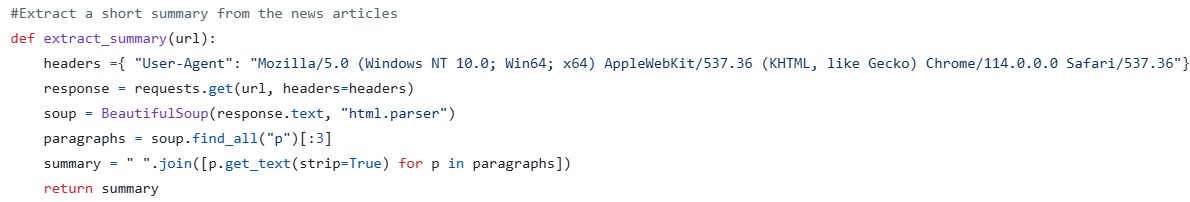
The utils.py file contains utility functions that facilitate news scraping, sentiment analysis, topic extraction, and Hindi TTS generation.

**Major Functions:**

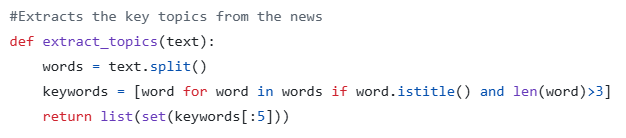
1. **scrape\_news():** Fetches the top 10 news articles related to the given company from BBC News.



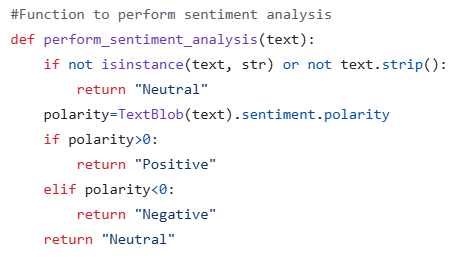
1. **extract\_summary():** Extracts a short summary from each news article.



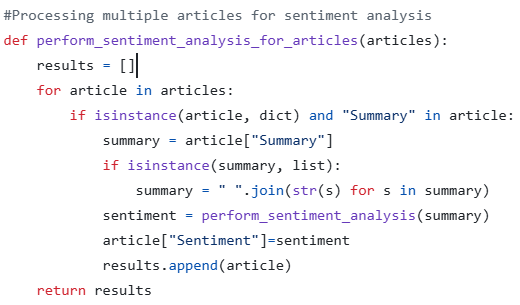
1. **extract\_topics():** Identifies key topics or keywords from the text.



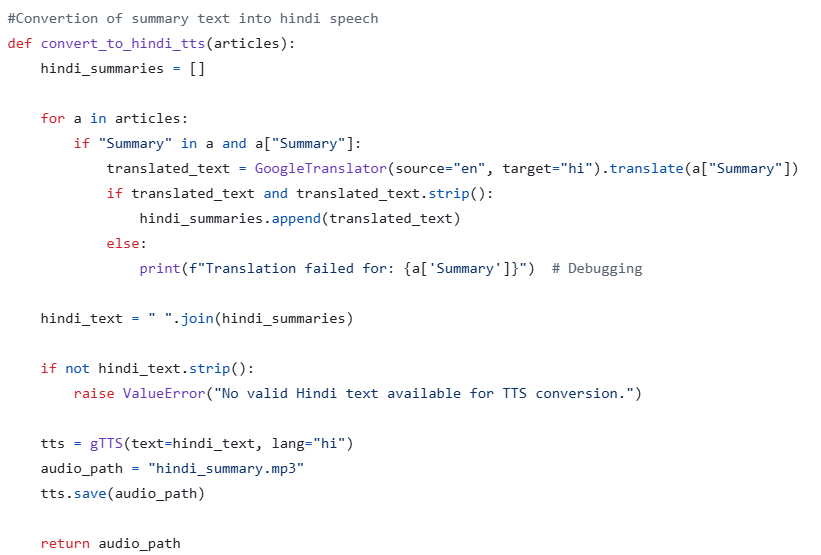
1. **perform\_sentiment\_analysis():** Analyzes the sentiment of the text and returns "Positive", "Negative", or "Neutral".



1. **perform\_sentiment\_analysis\_for\_articles():** Processes multiple articles to determine their sentiments.



1. **convert\_to\_hindi\_tts():** Converts summarized text to Hindi speech using Google Text-to-Speech (gTTS).



These utility functions play a vital role in extracting, processing, and analyzing news data while also enabling Hindi TTS generation.

**The issue with FastAPI on Hugging Face**

FastAPI is designed to work as a backend API framework and is generally run on servers. However, deploying FastAPI on Hugging Face Spaces may pose a challenge as Hugging Face does not directly support running backend servers. It primarily supports apps built using libraries like Streamlit and Gradio. As a result, FastAPI might not function as intended on Hugging Face Spaces.