News Summarization and Text-to-Speech Application

Overview:

This project provides a Streamlit-based interface for extracting and analyzing news articles, generating summaries with BART, and performing sentiment analysis using VADER. It also translates the content to Hindi and converts it into speech using gTTS.

Project Structure:

```
news-summarization/
        │
        ├── models              # put your models in this folder
        ├── app.py              # Streamlit application interface for user interaction
        ├── api.py              # Core functionality for extracting, analyzing, summarizing news
                                                                                    articles
        ├── utils.py            # Utility functions for saving reports and handling file operations
        ├── environment.yml     # Conda environment configuration for dependencies
        └── README.md           # Documentation file providing an overview and setup
                                                                              instructions
```

1. Project Setup:

Prerequisites:

Before running the project, make sure you have the following installed

Steps to Install and Run the Application:

Create a Virtual Environment: It's recommended to create a virtual environment to manage dependencies.

The **environment.yml** file defines all the Python dependencies that are required for the project. This file ensures that everything runs in an isolated environment(conda)

use the provided **environment.yml** file.

Run the following command to create the Conda environment:

conda env create -f environment.yml

Once the environment is created, activate it using

conda activate your_environment_name

Replace `your_environment_name` with the actual environment name specified in **environment.yml.**

Pre-download BART model:. The model `bart-large-cnn` is required for summarization. Ensure that the model directory contains both the model and tokenizer files,you can download the model using below link provided

git clone https://huggingface.co/facebook/bart-large-cnn

Once you have downloaded the models, place it into the model directory inside the project folder like below

```
└── models/
    ├── bart-large-cnn
```

Use that models directory path to load the model

Run the Application:

Launch the Streamlit application (app.py) by running the following command:

streamlit run app.py

Access the Web Application: Open the web application in your browser by navigating to:

http//localhost:8501

You can then enter the company name to analyze news articles related to it.

2. Model Details:

1.Summarization Model:

Pre-trained Model: The BART model is used for text summa sequence-to-sequence

model that uses a denoising autoencoder approach for pretraining.

Model: `facebook/bart-large-cnn`

Usage in the Project:

- The model is used to generate a concise summary of a collection of news articles.
- The summarization model reads the combined summaries of the articles and generates a more succinct version.

How it works:

- The input text (combined article summaries) is encoded using the tokenizer.
- The encoded text is passed through the BART model to generate a summary.
- The summary is decoded and displayed.

2.2 Sentiment Analysis: VADER

- **Model Name**: VADER (Valence Aware Dictionary and sEntiment Reasoner)
- **Pre-trained Model**: VADER is a lexicon and rule-based sentiment analysis tool that is particularly effective for analyzing social media content and news articles.
- **Usage in the Project:**
    - VADER is used to analyze the sentiment of each article's summary (positive, negative, or neutral).
    - The sentiment score is computed using VADER's polarity scores function, which outputs a compound score that helps classify the sentiment.
- **How it works:**
    - The summary of each article is passed through the VADER analyzer.
    - The sentiment score is determined based on predefined rules.
    - The sentiment is categorized as Positive, Negative, or Neutral.

2.3 Translation Model: Google Translate API

- **Model Name:** Google Translate
- **Pre-trained Model:** Google Translate API is used for text translation from English to Hindi.
- **Usage in the Project:**

    - The generated summary and sentiment are translated from English to Hindi before being converted to speech.
- **How it works**:
    - The `googletrans` library is used to interact with the Google Translate API.
    - The text is sent to Google Translate, which returns the translation.

2.4 Text-to-Speech (TTS): Google TTS (gTTS)

- **Model Name**: Google Text-to-Speech (gTTS)
- **Pre-trained Model:** Google TTS converts the provided text into speech.
- **Usage in the Project:**
    - After translating the summary and sentiment to Hindi, the `gTTS` module is used to convert the Hindi text to speech and save it as an MP3 file.
- **How it works:**
    - The text (summary and sentiment) is converted into speech using `gTTS`.
    - The speech is saved as an MP3 file that can be played back in the web app.

3. API Development

How the APIs are being used

In your Streamlite app, the "API" is essentially a series of backend functions that perform various tasks, such as:

- **Extracting news articles** from the BBC website.
- **Summarizing the articles** using the pre-trained BART model.
- **Performing sentiment analysis** using the VADER sentiment analyzer.
- **Translating the summary** into Hindi using Google Translate API.
- **Converting the summary to speech** using Google Text-to-Speech (gTTS).

These backend processes can be considered APIs in the sense that they handle different functionalities and return results (data) that the user interacts with.

How APIs are being accessed in the app:

The primary interaction with these "APIs" happens through **Streamlit's widgets** and **backend functions**:

1. **Input from the User:**
    - The user enters the company name into a text input field (`st.text_input()`).
2. **API-like backend processing:**
    - When the user presses the "Analyze" button (`st.button()`), the app triggers backend functions to process the input, retrieve the news, summarize it, perform sentiment analysis, and generate speech.
3. **Results Display:**
    - The results of the backend processing (summarized news, sentiment analysis, and speech) are returned and displayed directly in the Streamlit app.

In summary:

- The backend functions (`extract_articles`, `generate_combined_summary`, `analyze_sentiment`, `translate_to_hindi`, and `text_to_speech`) simulate API behavior by performing distinct tasks and returning data that is presented to the user.

4. API Usage (Third-Party API Integration):

- **Article Extraction**: To retrieve relevant articles based on the entered company name.
- **Google Translate API**: To translate the text into Hindi.
- **gTTS (Google Text-to-Speech)**: To convert the generated summary into speech.

Details of API Integration:

- **Article Extraction**:
    - **Purpose**: To extract relevant news articles based on the company name entered by the user. This is done through a third-party news API or web scraping tools.

    - **Integration**: If using a **BeautifulSoup**,**with request** articles are extracted from URLs directly.

Google Translate API:

- Purpose: To translate the text (combined summary) from English to Hindi.
- Integration: `googletrans` Python library is used for translation in the `translate_to_hindi` function.

Google Text-to-Speech (gTTS):

- Purpose: Converts the summary (translated to Hindi) into speech.
- Integration: The `gTTS` library is used to generate the speech and save it as an MP3 file

Assumptions & Limitations

Assumptions:

1. The application requires internet access for fetching articles and using third-party services.
2. A valid company name or keyword must be provided for article extraction.
3. The BART model for summarization is available and properly set up.
4. The translation and TTS features support English to Hindi translations.
5. The extracted article text fits within the model's token limits

Limitations:

Article Extraction: Quality depends on the external service, and it may return inconsistent data.

Sentiment Analysis: May not always capture complex sentiments, especially in long articles.

Model Resources: The BART model requires significant resources and may be slow on low-end systems.

Dependencies: The application relies on third-party libraries; issues may arise if these are not installed or compatible.

Conclusion:

This project analyzes news articles about a specified company, summarizes the articles, performs sentiment analysis, translates the summary and sentiment to Hindi, and generates a speech file. It's designed to be simple to set up and, providing a useful tool for analyzing news and generating audio summaries.