## Sarvam Ai Hiring challenge

Task 1: Semantic Chunking of a Youtube Video

**Problem Statement:**

The objective is to extract high-quality, meaningful (semantic) segments from the specified YouTube video: [Watch Video](https://www.youtube.com/watch?v=Sby1uJ_NFIY).

Suggested workflow:

1. **Download Video and Extract Audio:** Download the video and separate the audio component.
2. **Transcription of Audio:** Utilize an open-source Speech-to-Text model to transcribe the audio. *Provide an explanation of the chosen model and any techniques used to enhance the quality of the transcription.*
3. **Time-Align Transcript with Audio:** *Describe the methodology and steps for aligning the transcript with the audio.*
4. **Semantic Chunking of Data:** Slice the data into audio-text pairs, using both semantic information from the text and voice activity information from the audio, with each audio-chunk being less than 15s in length. *Explain the logic used for semantic chunking and discuss the strengths and weaknesses of your approach.*

**Judgement Criteria:**

1. **Precision-Oriented Evaluation:** The evaluation focuses on precision rather than recall. Higher scores are achieved by reporting fewer but more accurate segments rather than a larger number of segments with inaccuracies. Segment accuracy is determined by:
   * **Transcription Quality:** Accuracy of the text transcription for each audio chunk.
   * **Segment Quality:** Semantic richness of the text segments.
   * **Timestamp Accuracy:** Precision of the start and end times for each segment. Avoid audio cuts at the start or end of a segment.
2. **Detailed Explanations:** Provide reasoning behind each step in the process.
3. **Generalization:** Discuss the general applicability of your approach, potential failure modes on different types of videos, and adaptation strategies for other languages.
4. **[Bonus-1]** **Gradio-app Interface:** Wrap your code in a gradio-app which takes in youtube link as input and displays the output in a text-box.
5. **[Bonus-2]** **Utilizing Ground-Truth Transcripts:** Propose a method to improve the quality of your transcript using a ground-truth transcript provided as a single text string. Explain your hypothesis for this approach. *Note that code-snippet isn't required for this question.*

As an example - for the audio extracted from [yt-link](https://www.youtube.com/watch?v=ysLiABvVos8" \t "_blank), how can we leverage transcript scraped from [here](https://www.google.com/url?q=https%3A%2F%2Fwww.newsonair.gov.in%2Fbulletins-detail%2Fenglish-morning-news-7%2F), to improve the overall transcription quality of segments?

**Submission Format:**

Your submission should be a well-documented Jupyter notebook capable of reproducing your results. The notebook should automatically install all required dependencies and output the results in the specified format.

* **Output Format:** Provide the results as a list of dictionaries, each representing a semantic chunk. Each dictionary should include:
  + chunk\_id: A unique identifier for the chunk (integer).
  + chunk\_length: The duration of the chunk in seconds (float).
  + text: The transcribed text of the chunk (string).
  + start\_time: The start time of the chunk within the video (float).
  + end\_time: The end time of the chunk within the video (float).

sample\_output\_list = [  
    {  
        "chunk\_id": 1,  
        "chunk\_length": 14.5,  
        "text": "Here is an example of a semantic chunk from the video.",  
        "start\_time": 0.0,  
        "end\_time": 14.5,  
    },  
    # Additional chunks follow...  
]

Ensure that your code is clear, well-commented, and easy to follow, with explanations for each major step and decision in the process. The notebook should be able to install all the dependencies automatically and generate the reported output when run.

**Solution:**

**Step1:**

1. **Download Video and Extract Audio:** Download the video and separate the audio component.

Explanation: I was downloaded video and extracted the audio

**Step 2:**

**2.Transcription of Audio:** Utilize an open-source Speech-to-Text model to transcribe the audio. *Provide an explanation of the chosen model and any techniques used to enhance the quality of the transcription*

Explanation: I have used two models in this transcription of audio **Explanation of the Code**

Here’s a detailed explanation of each line of the code you provided:

python

!pip install huggingsound -q

- This command installs the `huggingsound` library using `pip`. The `-q` option stands for "quiet", which suppresses the usual output of the installation process. `huggingsound` is a wrapper around Hugging Face's models specifically for handling sound data.

python

from huggingsound import SpeechRecognitionModel

- This line imports the `SpeechRecognitionModel` class from the `huggingsound` library. This class provides an interface to load and use speech recognition models.

```python

import torch

```

- This line imports the `torch` library, which is essential for working with PyTorch models. PyTorch is a popular deep learning framework that provides tensor computation and automatic differentiation.

```python

device = "cuda" if torch.cuda.is\_available() else "cpu"

```

- This line checks if a CUDA-compatible GPU is available. If a GPU is available, it sets the device to "cuda", otherwise it sets it to "cpu". Using a GPU can significantly speed up model inference and training.

```python

device

```

- This line outputs the value of the `device` variable. This is useful to confirm whether the script will use the CPU or GPU.

```python

model = SpeechRecognitionModel("jonatasgrosman/wav2vec2-large-xlsr-53-english", device=device)

```

- This line initializes a `SpeechRecognitionModel` with the specified model name `jonatasgrosman/wav2vec2-large-xlsr-53-english` and sets the device to the previously determined device (either "cuda" or "cpu"). The model `wav2vec2-large-xlsr-53-english` is a pre-trained Wav2Vec 2.0 model fine-tuned on English speech data.

### Why Use This Model?

The model `jonatasgrosman/wav2vec2-large-xlsr-53-english` is based on the Wav2Vec 2.0 architecture, which is known for its high performance in speech recognition tasks. Here are some reasons for using this specific model:

1. Accuracy: Wav2Vec 2.0 models are among the state-of-the-art for ASR (Automatic Speech Recognition).

2. Pre-trained: This model is pre-trained on a large dataset of English speech, which makes it capable of recognizing a wide range of English accents and speaking styles.

3. Efficiency: Despite being large, the model is optimized for efficient inference, especially when using a GPU.

Potential Alternatives and Their Pros and Cons

There are several other models available for speech recognition. Here are a few notable ones:

1. wav2vec2-large-960h:

- Pros: Specifically fine-tuned on 960 hours of English speech from the LibriSpeech dataset, very accurate for English.

- Cons: May not perform as well on non-standard accents or noisy environments compared to models trained on more diverse datasets.

2. wav2vec2-large-xlsr-53:

- Pros: Multilingual model trained on 53 languages, versatile for applications needing multiple languages.

- Cons: Might be less accurate for English compared to models specifically fine-tuned on English datasets.

3. DeepSpeech:

- Pros: Open-source and provides a good balance of performance and computational efficiency.

- Cons: Generally less accurate than Wav2Vec 2.0 models.

4. Jasper:

- Pros: Designed for efficient training and inference on GPUs, high accuracy.

- Cons: Large model size, which might be resource-intensive.

Drawbacks of `jonatasgrosman/wav2vec2-large-xlsr-53-english`

While this model is highly effective, it has some limitations:

1. Resource-Intensive: Large models like this require significant computational resources, especially for training or when running on large datasets.

2. Latency: Inference time can be slower on CPUs, making it less suitable for real-time applications without GPU acceleration.

3. Specialization: Although it performs well on English, it may not be as effective for domain-specific jargon or very noisy environments compared to some models fine-tuned on specific types of data.

In summary, `jonatasgrosman/wav2vec2-large-xlsr-53-english` is a powerful ASR model with high accuracy for general English speech recognition tasks. However, depending on the specific requirements (like real-time processing or handling domain-specific language), other models might be more suitable.

**Audio Chunking:**

It is used for to extract audio and make them in to chunks

We have used librosa library

Librosa is a Python library widely used for audio and music analysis. It provides tools to work with audio data, enabling users to load audio files, extract features, and perform various signal processing tasks.

### Explanation of Each Step

Whisper Model:

* The latest text to speech model, optimized for quality. Whisper is a general-purpose speech recognition model. It is trained on a large dataset of diverse audio and is also a multi-task model that can perform multilingual speech recognition as well as speech translation and language identification.

1. **Download Audio from YouTube**:
   * pytube fetches the audio stream from a given YouTube URL and downloads it as audio.mp4.
2. **Convert to WAV**:
   * pydub converts the downloaded audio file to WAV format to ensure compatibility with ASR models.
3. **Play Audio**:
   * pydub.playback plays the audio directly in your Python environment.
4. **Display Audio Player**:
   * IPython.display.Audio provides an interactive audio player widget in a Jupyter notebook.
5. **Transcribe Audio**:
   * transformers.pipeline initializes an ASR pipeline using a pre-trained Wav2Vec 2.0 model to transcribe the audio to text.

**Step 3:**

**What is semantic chunking?**

* The main idea behind semantic chunking is to split a given text based on how similar the chunks are in meaning. This similarity is calculated by chunking the given text into sentences, then turning all these text-based chunks into vector embeddings and calculating the cosine similarity between these chunks.

**Bonus-1: Gradio-App Interface:**

To create a Gradio app that takes a YouTube link as input and displays the semantic chunks.

**Accuracy:**

### Word Error Rate (WER)

**WER** is the most widely used metric for evaluating the performance of ASR systems. It measures the difference between the reference transcription (ground truth) and the hypothesis transcription (predicted by the ASR system) in terms of word-level errors. WER is calculated as:

WER=S+D+I/N

* **S**: Substitutions - the number of words that were incorrectly transcribed as another word.
* **D**: Deletions - the number of words that were omitted in the transcription.
* **I**: Insertions - the number of words that were added incorrectly.
* **N**: Total number of words in the reference transcription.

WER is particularly important because:

* It directly measures the intelligibility of the transcription.
* Lower WER means higher accuracy and better performance of the ASR system.
* It takes into account all types of errors (substitutions, deletions, insertions), providing a comprehensive measure of performance.

### Character Error Rate (CER)

**CER** is similar to WER but operates at the character level instead of the word level. It measures the difference between the reference and hypothesis transcriptions in terms of character-level errors. CER is calculated as:

CER=S+D+I/N

* **S**: Substitutions - the number of characters that were incorrectly transcribed as another character.
* **D**: Deletions - the number of characters that were omitted in the transcription.
* **I**: Insertions - the number of characters that were added incorrectly.
* **N**: Total number of characters in the reference transcription.

### **Typical Accuracy Benchmarks**

While specific benchmarks can vary, here are some rough guidelines for acceptable ASR accuracy:

* **Very Good**: 95% or higher WER/CER. This level is typically expected in high-stakes applications where accuracy is critical.
* **Good**: 90-95% WER/CER. This range is often acceptable for many practical applications, including customer service applications and general transcription.
* **Fair**: 80-90% WER/CER. Accuracy in this range may be acceptable for less critical applications or in scenarios where the complexity of speech varies.
* **Poor**: Below 80% WER/CER. Accuracy in this range may indicate significant issues with the ASR system, potentially requiring further training, data enhancement, or model improvement.

Links Used in Task 1:

<https://youtu.be/A1I1fBXr0rc?si=IPTaYVX5JDudEPHc>

<https://transkriptor.com/what-is-audio-transcription/>

<https://youtu.be/rhEgDJT_2w0?si=0VM6WorxkfK-5uKK>

**Task 2:**

Task 2: Exploratory Data Analysis of New Testament Audio and Text

**Problem Statement:**

The objective of this task is to conduct a comprehensive exploratory data analysis (EDA) on the audio and text data of the 260 chapters of the New Testament in your mother tongue (excluding English). The data should be obtained through web scraping from [Faith Comes By Hearing](https://www.google.com/url?q=https%3A%2F%2Fwww.faithcomesbyhearing.com%2F).

The workflow for this task should include:

1. **Web Scraping:** Systematically download the audio files and their corresponding textual content for each of the 260 chapters of the New Testament from the specified website.
2. **Data Preparation:** Organize the data by chapters, ensuring each audio file is matched with its corresponding text.
3. **Exploratory Data Analysis:** Analyze the data to uncover patterns, anomalies, or insights that could benefit applications such as Text to Speech (TTS) and Speech to Text (STT) technologies. Your analysis should explore various facets of the data, including audio quality, text clarity, and alignment between text and spoken content.

**Judgement Criteria:**

Your submission will be evaluated based on:

* **Efficiency and Reliability of Web Scraping Techniques:** The methods and tools used for downloading the chapters efficiently and reliably.
* **Data Analysis Methods:** The techniques and approaches used for analyzing the audio and text data.
* **Quality of Data Analysis:** How effectively the analysis addresses potential applications for the Speech team, including TTS and STT technologies.
* **Creativity in Analysis:** Innovative approaches in data handling and analysis, and the use of relevant metrics to assess data quality and applicability.

**Submission Requirements:**

Your submission should include the following components:

* **Report on Key Performance Indicators (KPIs):** A concise report detailing the key findings from your analysis, focusing on aspects that are critical for improving TTS and STT applications.
* **Methodological Explanation:** A thorough explanation of the methods used for both web scraping and the exploratory data analysis. This should include challenges faced and how they were overcome.
* **Supporting Materials:** Include code snippets and visualizations that highlight significant insights from the data. These should be well-documented and easy to understand, demonstrating the logic behind your analytical decisions.

The report should be structured to clearly present the methodology, findings, and implications of your analysis. It should be technical yet accessible, aimed at stakeholders who may have varying levels of familiarity with data analysis techniques.

**Step 1:**

**Web Scrapping:**

**We use selenium and beautiful soup:**

Web scraping using Selenium and Beautiful Soup is a powerful combination for extracting data from web pages. Selenium is used to automate browser actions and handle dynamic content, while Beautiful Soup is used to parse the HTML and extract the required data.

**Web driver Manager:**

Using webdriver-manager simplifies the management of browser drivers for Selenium. It automatically handles downloading, setting up, and updating the drivers, such as ChromeDriver, eliminating the need for manual management. This ensures compatibility with the installed browser versions and avoids issues related to outdated drivers.

Here’s a detailed explanation of why and how to use webdriver-manager:

### Why Use WebDriver Manager?

1. **Automatic Downloads**: It automatically downloads the correct version of the driver that matches your browser version.
2. **Compatibility**: Ensures that you have a compatible driver version for your browser, avoiding issues due to version mismatches.
3. **Simplifies Setup**: Eliminates the need to manually download and specify the path to the driver executable.
4. **Updates**: Automatically checks for and downloads driver updates, ensuring you always have the latest version.
5. **Cross-Platform**: Works seamlessly across different operating systems

Step 2: By Using above step we have used to extract all text,audio from the specific website.

Step 3: We have organized all text and audio data and we have passed a print message as “Data Organized Successfully”.

Step 4: We have Done Analysis part between Audio duration and Text length.It has been “Disturbution of audio durations” and “Disturbution of Text Lengths”.We used matplot library for visualization.

Step 5:Transcript the audio to Text we have done this same like task 1.

Step 6:We created a data frame and displayed only first 5 rows .

Step 7: Exploratory data analysis is done first we will single noise Ratio ,Single noise disturbution ratio,We will count how many sentences and words are present ,We will see audio duration it has been taken .

Step 8: We find Accuracy

BLEU Score: Measures precision of n-grams in the candidate translation against reference translations. Higher BLEU score indicates better translation quality.

The models 90-100 percent is excellent, 80-70 is Good,60-50 we have to work on model again.

**Calculate ROUGE Score:**

ROUGE Score: Measures recall of n-grams. Useful for tasks like summarization.

Links :

Youtube

Github

Chatgpt .