OBJECTIVE:

To build a multilingual speech recognition model without training using pre trained multilingual speech recognition model, such as Multilingual whisper, to enable RAG to perform task in multiple languages.

BACKGROUND:

RAG is a generative model that can be used for a variety of tasks, including speech recognition, translation, and summarization. However, RAG is currently only trained to perform these tasks in a single language. By building a multilingual speech recognition model without training, we can enable RAG to perform these tasks in multiple languages without the need for additional training.

CODE:

```
!pip install git+https://github.com/openai/whisper.git -q

!pip install transformers -q

!pip install torch -q

!pip install librosa -q

import whisper

model = whisper.load_model("medium",device='cpu')

from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

   print(f"Uploaded file: {filename}")

import librosa

audio_path = list(uploaded.keys())[0]

def load_audio_in_chunks(audio_path, chunk_duration=30):

   audio, sr = librosa.load(audio_path, sr=16000)

   total_duration = librosa.get_duration(y=audio, sr=sr)

   chunk_samples = int(chunk_duration * sr)

   for start_sample in range(0, len(audio), chunk_samples):

      yield audio[start_sample:start_sample + chunk_samples], sr

chunks = list(load_audio_in_chunks(audio_path))

print(f"Loaded {len(chunks)} chunks from the audio file.")

import whisper

model = whisper.load_model("medium", device="cpu")

transcriptions = []
```

```
for i, (audio_chunk, sr) in enumerate(chunks):
    result = model.transcribe(audio_chunk, fp16=False)  #FP32 is used
    transcriptions.append(result["text"])
    print(f"Transcription of chunk {i+1}/{len(chunks)}: {result['text']}")
full_transcription = " ".join(transcriptions)
print("Full Transcription:", full_transcription)
from transformers import MarianMTModel, MarianTokenizer
language_models = {
    "de": "Helsinki-NLP/opus-mt-en-de",  # German
    "es": "Helsinki-NLP/opus-mt-en-es",  # Spanish
    "fr": "Helsinki-NLP/opus-mt-en-fr",  # French
    "it": "Helsinki-NLP/opus-mt-en-it",  # Italian
    "zh": "Helsinki-NLP/opus-mt-en-zh",  # Chinese
    # Add more languages here as needed
}
print("Choose the target language code from the following options:")
for lang_code in language_models.keys():
    print(f"{lang_code} - {language_models[lang_code].split('-')[-1]}")
target_language = input("Enter the target language code: ")
if target_language not in language_models:
    print("Invalid language code. Please run the cell again and enter a valid code.")
else:
    model_name = language_models[target_language]
    tokenizer = MarianTokenizer.from_pretrained(model_name)
    translation_model = MarianMTModel.from_pretrained(model_name)
    full_transcription =  " ".join(transcriptions)S
    print("Full Transcription:", full_transcription)
    inputs = tokenizer(full_transcription, return_tensors="pt", padding=True)
    translated_tokens = translation_model.generate(**inputs)
    translated_text = tokenizer.decode(translated_tokens[0], skip_special_tokens=True)
    print("Translated Text:", translated_text)

!pip install datasets

!pip install faiss-cpu

from transformers import RagTokenizer, RagRetriever, RagSequenceForGeneration

tokenizer = RagTokenizer.from_pretrained("facebook/rag-token-nq")

retriever=RagRetriever.from_pretrained("facebook/rag-token-nq",index_name="exact",
use_dummy_dataset=True)

rag_model = RagSequenceForGeneration.from_pretrained("facebook/rag-token-nq", retriever=retriever)

query = translated_text

input_ids = tokenizer(query, return_tensors="pt").input_ids

generated_ids = rag_model.generate(input_ids=input_ids)

response = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)

print("RAG Response:", response)
```

References:

https://github.com/nikkibommu/Building-a-Multilingual-Speech-Recognition-Model-for-RAG-Without-Training