Multilingual Speech to Text for RAG

```
import gradio as gr
from transformers import pipeline, RagRetriever, RagTokenForGeneration, RagTokenizer
from langdetect import detect, DetectorFactory
import torch
import os
from googletrans import Translator

DetectorFactory.seed = 0

def initialize_rag_model():
    try:
      rag_tokenizer = RagTokenizer.from_pretrained("facebook/rag-token-nq")
      retriever = RagRetriever.from_pretrained("facebook/rag-token-nq", index_name="exact", use_dummy_dataset=Tru
      rag_model = RagTokenForGeneration.from_pretrained("facebook/rag-token-nq", retriever=retriever)
      return retriever, rag_model, rag_tokenizer

    except Exception as e:
        print(f"Error initializing RAG model: {e}")
        return None, None, None

retriever, rag_model, rag_tokenizer = initialize_rag_model()

if not retriever or not rag_model or not rag_tokenizer:
    raise RuntimeError("Failed to load RAG model and components.")


translator = Translator()

def speech_to_text_with_lang_detection(audio_file):
    """
    Convert speech to text using Whisper model and detect the language.
    """
    if not os.path.exists(audio_file):
        raise FileNotFoundError(f"Audio file not found: {audio_file}")

    asr = pipeline("automatic-speech-recognition", model="openai/whisper-base")
    try:
        result = asr(audio_file)
        text = result['text']
        detected_lang = detect(text)
        return text, detected_lang
    except Exception as e:
        print(f"Error in speech-to-text conversion: {e}")
        return "Error during transcription", "unknown"
```

Translation Function using Google Translate API

```
def translate_text(text, src_lang, tgt_lang='en'):
    """
    Translate text from source language to target language using Google Translate API.
    """
    try:
        translated = translator.translate(text, src=src_lang, dest=tgt_lang)
        return translated.text
    except Exception as e:
        print(f"Translation error: {e}")
        return text
```

RAG Function

```
def generate_answer_with_rag(question):
    """
    Use RAG to generate an answer based on the input question.
    """
    inputs = rag_tokenizer(question, return_tensors="pt")
    try:
        print(f"Input to RAG: {inputs}")
        with torch.no_grad():
            outputs = rag_model.generate(**inputs)
        print(f"RAG outputs: {outputs}")
        answer = rag_tokenizer.batch_decode(outputs, skip_special_tokens=True)[0]
        return answer
    except Exception as e:
        print(f"RAG generation error: {e}")
        return "Error generating answer with RAG."
```

Process Audio Query Function

```
def process_audio_query(audio_file, target_lang='en'):
    """
    Process an audio file by detecting the language, transcribing it, translating it, and using RAG to generate
    """

    text, detected_lang = speech_to_text_with_lang_detection(audio_file)
    print(f"Detected language: {detected_lang}")
    print(f"Transcribed text: {text}")


    if detected_lang != target_lang and detected_lang != 'unknown':
        translated_text = translate_text(text, src_lang=detected_lang, tgt_lang=target_lang)
    else:
        translated_text = text


    answer = generate_answer_with_rag(translated_text)

    if answer and target_lang != 'en':
        rag_answer_translated = translate_text(answer, src_lang='en', tgt_lang=target_lang)
    else:
        rag_answer_translated = answer

    return (f"Detected Language: {detected_lang}\n\n"
            f"Transcription: {text}\n\n"
            f"Transcription (translated to {target_lang}): {translated_text}\n\n"
            f"RAG Answer: {answer}\n\n"
            f"RAG Answer (translated to {target_lang}): {rag_answer_translated}")
```

Gradio Interface

```
def gradio_interface(audio, target_lang='en'):
    return process_audio_query(audio, target_lang)

iface = gr.Interface(
    fn=gradio_interface,
    inputs=[gr.Audio(type="filepath"), gr.Textbox(label="Target Language (e.g., 'en', 'es', 'te')")],
    outputs="text",
    title="Multilingual Speech-to-Text and RAG-based Answer Generation",
    description="Upload an audio file, select the target language, and the system will automatically detect the l
)


if __name__ == "__main__":
    iface.launch()
```