使用 RAG 解決專有名詞或同音詞的可能方法

一、RAG 程序的詳細流程

1. **數據準備**
    - 收集和清理文本數據，確保數據中包含足夠的上下文信息。
    - 準備一個包含正確詞彙和用法的參考語料庫。
2. **文本嵌入和索引建立**
    - 使用預訓練的語言模型將文本數據轉換為向量嵌入。
    - 使用相似性搜索庫（如 FAISS）建立向量索引，便於快速檢索。
3. **查詢處理和檢索**
    - 將用戶查詢（或 Whisper 的轉錄結果）轉換為向量嵌入。
    - 在索引中檢索與查詢最相似的文檔或句子。
4. **生成和結合**
    - 使用生成模型（如 GPT）結合檢索到的文檔和原始查詢生成回答。
    - 根據需要調整生成的結果以解決同音詞問題。

二、所需的套件和函式庫

- **PyTorch**：深度學習框架，用於訓練和運行模型。
- **Transformers**：來自 Hugging Face 的庫，提供預訓練的語言模型。
- **FAISS**：Facebook AI Similarity Search，用於高效的相似性搜索。
- **Sentence-Transformers**：用於生成文本嵌入的庫。
- **Librosa**：用於音頻處理的 Python 庫。
- **Whisper**：OpenAI 的語音識別模型。

範例：解決 whisper 中的同音字問題

```
import torch from transformers
import WhisperProcessor, WhisperForConditionalGeneration
from sentence_transformers import SentenceTransformer
import faiss import numpy as np
import librosa
```

```
# 1. 準備 Whisper 模型
processor = WhisperProcessor.from_pretrained("openai/whisper-base")
model = WhisperForConditionalGeneration.from_pretrained("openai/whisper-base")

# 2. 準備 RAG 組件
embedder = SentenceTransformer('paraphrase-MiniLM-L6-v2')

# 假設我們有一個包含正確詞彙用法的文本語料庫
corpus = [ "The weather is fair today.", "The fare for the bus is $2.", "I need to
pair these socks.", "Please peel the pear for me.", # ... 更多相關句子 ]

# 創建語料庫的嵌入
corpus_embeddings = embedder.encode(corpus, convert_to_tensor=True)

# 使用 FAISS 建立索引
index = faiss.IndexFlatL2(corpus_embeddings.shape[1])
index.add(corpus_embeddings.cpu().numpy())

# 3. 語音識別函數
def transcribe_audio(audio_file):
    audio, _ = librosa.load(audio_file, sr=16000)
    input_features=processor(audio,sampling_rate=16000,return_tensors="pt").input_f
    eatures
    predicted_ids = model.generate(input_features)
    transcription = processor.batch_decode(predicted_ids, skip_special_tokens=True)
    return transcription[0]
# 4. RAG 修正函數
def rag_correction(transcription):
    # 將轉錄文本轉換為嵌入
    query_embedding = embedder.encode(transcription, convert_to_tensor=True)
    # 使用 FAISS 進行相似性搜索
    _, I = index.search(query_embedding.cpu().numpy().reshape(1, -1), k=5)
    # 獲取最相似的句子
    similar_sentences = [corpus[i] for i in I[0]]
    # 這裡可以實現更複雜的邏輯來選擇最合適的修正
    # 簡單起見，我們只返回最相似的句子
    return similar_sentences[0]
```

```
# 5. 主程序
audio_file = "path_to_your_audio_file.wav"
transcription = transcribe_audio(audio_file)
print("Original transcription:", transcription)
corrected_text = rag_correction(transcription)
print("Corrected text:", corrected_text)
```

在這個範例中，我們首先使用 Whisper 模型進行語音識別。然後，我們使用 RAG 方法來改進識別結果，特別是處理同音詞的問題。

RAG 的工作原理如下：

1. 我們預先準備了一個包含正確詞彙用法的文本語料庫。
2. 使用 Sentence-Transformers 將語料庫中的每個句子轉換為向量嵌入。
3. 使用 FAISS 建立這些嵌入的索引，以便快速檢索。
4. 當 Whisper 生成轉錄文本時，我們將其轉換為向量嵌入。
5. 使用這個嵌入在我們的語料庫中搜索最相似的句子。
6. 返回最相似的句子作為修正後的文本。

這個方法可以幫助解決同音詞的問題，因為它利用了上下文信息。例如，如果 Whisper 錯誤地將"The fare for the bus"轉錄為"The fair for the bus"，我們的 RAG 系統可能會找到正確使用"fare"的相似句子，從而提供正確的修正。

這是一個基本概念的實作，可能需要更複雜的邏輯來決定是否接受 RAG 的建議，可能結合其他 NLP 技術如詞性標註、命名實體識別等來進一步提高準確性。此外，為了獲得最佳效果，儘量讓語料庫足夠大且多樣化，能夠覆蓋各種可能的用例和上下文。