Assignment 4: Retrieval-Augmented Generation with LangChain

1. **Please describe the details of your implementation for the RAG system (please tell us 1. What’s in your RAG system? 2. Which retrieval model you use? 3. What’s your prompt? 4. What’s new in your code in comparison with the code from our lab course?) in this assignment and list your best score for the ten questions**

**我將會依照code的順序性來講解**

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

**首先，vector store以及 retriever是新的而不是使用lab範例的。**

**Retriever 用的是MMR (Maximal Marginal Relevance)，它能在在相關性與多樣性之間平衡，避免檢索結果過於相似。而lab的是利用向量相似度檢索最相關的文檔。**

**vector store 用的是Annoy (Approximate Nearest Neighbors Oh Yeah)，它構建Annoy索引，並執行最近鄰搜索，由於我們的資料庫較小，因此會有優勢。**

**一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述**

**接著是RAG system，那些都是參考lab的。**

**Model使用規定的ollama**

**question\_answer\_chain的工作原理是將多個文檔合併成一個大的文本，並將其提供給 LLM 進行處理。**

**create\_retrieval\_chain 的工作原理是結合檢索器和問答鏈，實現文檔檢索後的生成處理。**

**一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述**

**而prompt是由原本lab的code進行改良，能夠Think step by step，引導模型按順序處理上下文中的信息並找到相關的答案，但是推理的比例還是存在的，並且還限制了文字數量。**

**一張含有 文字, 字型, 螢幕擷取畫面, 數字 的圖片

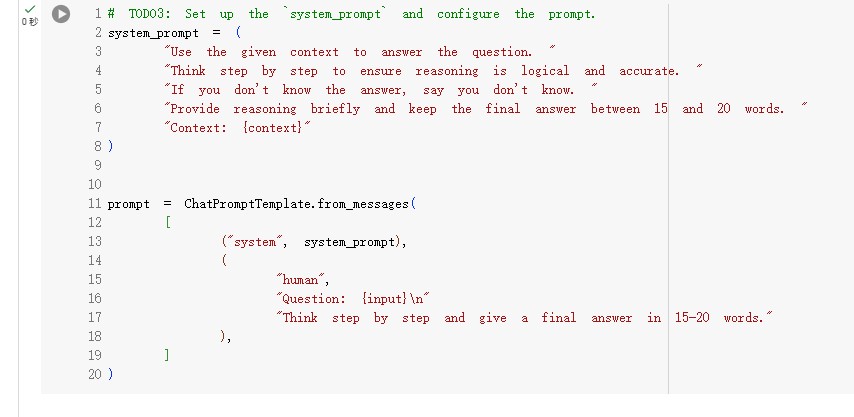
自動產生的描述**

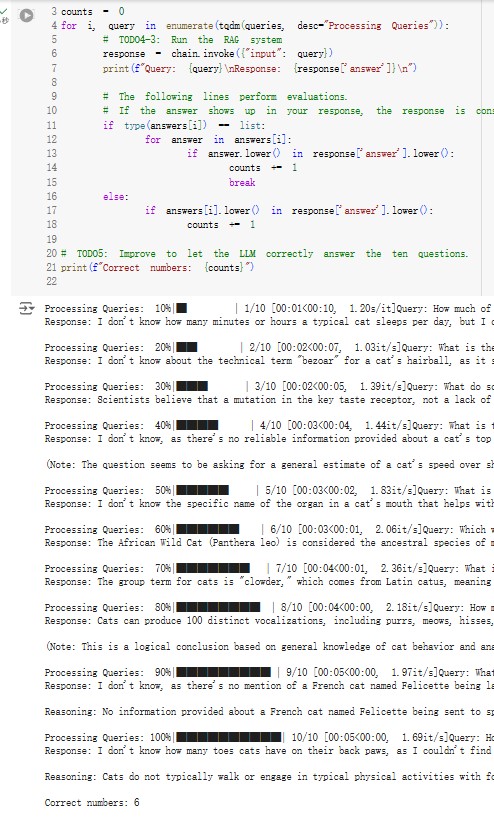
**最後這樣改完並統整之後，我的成功率提升至九成，但是有時候不穩定，成功率有時候會掉至六成，但大部分還是對九題。**

1. **Please provide analysis for the RAG performance using different prompts.**

**一張含有 文字, 螢幕擷取畫面, 字型, 信 的圖片

自動產生的描述**

**首先這就是我只使用助教課程 RAG USING Langchain 的範例所展示出的最成果，他的成功率只有六成，因此接下來進行prompts的測試。**

**接下來這是我第一個prompt的example，我參考了zero-shot的想法，在提示中加入「Think step by step」以啟動model的推理能力。以及當句子越短的時候，會比較好生出正確的答案，但我並沒有設太低。**

**但還是一樣，成功率只有六成**

**一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述**

**接著我又想到基於推理還是不夠，因此我又增加引導模型按順序處理上下文中的信息並找到相關的答案( system prompt 紅色字第二行)。**

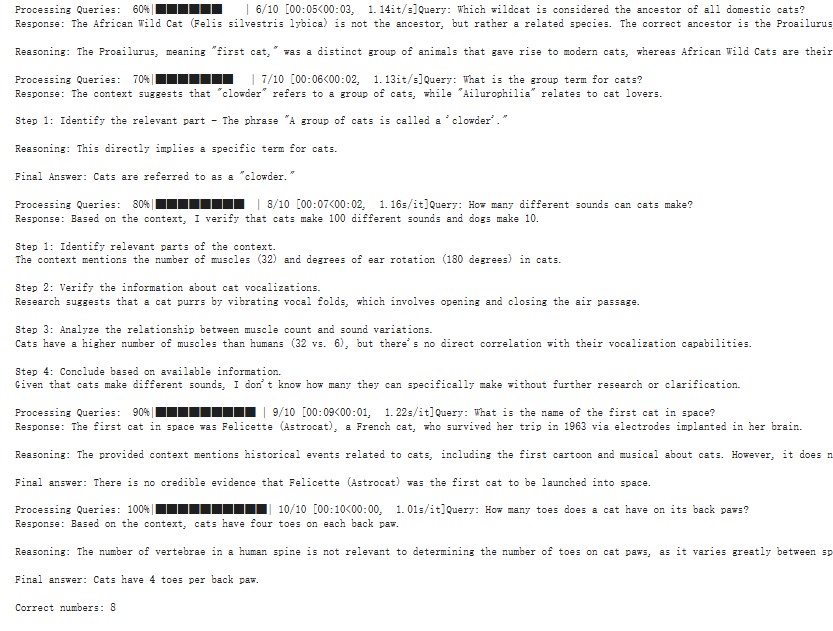
**一張含有 文字, 信, 字型, 螢幕擷取畫面 的圖片

自動產生的描述**

**這次好很多，成功率升至八成。**

**一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述**

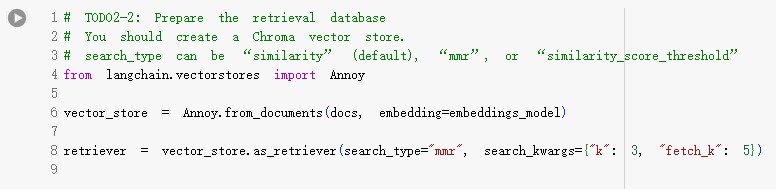
**最後我又想到題目或答案可能會藏有陷阱文字，因此我又稍微增加推理的比例( system prompt 紅色字第三行)。**

**而還是八題，這次的功能效能不太明顯，八題可能對只更改prompt 來說可能就是極限了!**

1. **Please compare the RAG performance with different retrieval models and the performance without using RAG (note that Llama 3.2 should not be fine-tuned in this assignment).**

**指的應該是todo2-2 和 todo4-1 ? 接下來將會分別測試**

**接下來是我使用的最好的performance(解說在第一題)(圖有四張)，成功率為90%**

**一張含有 文字, 字型, 螢幕擷取畫面 的圖片

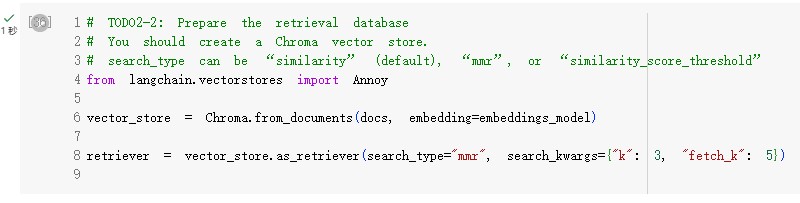
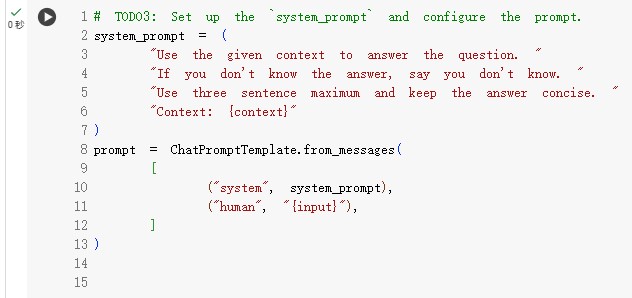
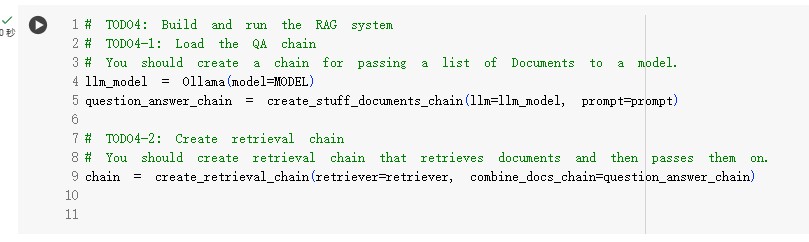
自動產生的描述**

**一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

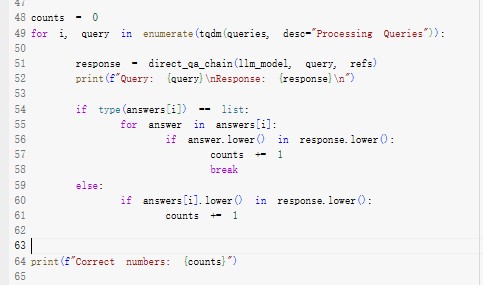
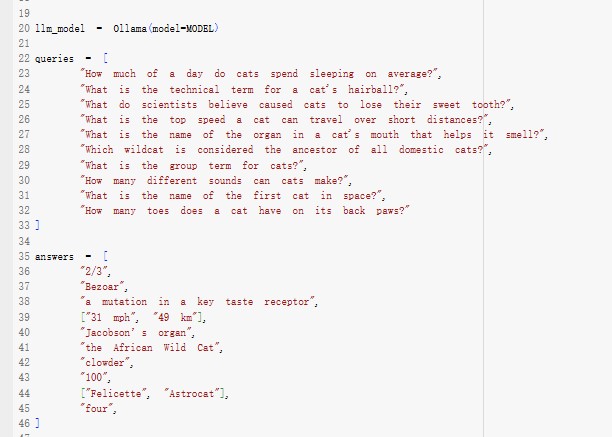
自動產生的描述**

**接著回來使用lab中的code，跟上面不一樣的是，Prompt基本上是從資料庫裡尋找答案，並沒有經過思考集推理，而todo的vectorstore跟retriever也是平衡相關性與多樣性，跟上面比的話上面的作法比較有優勢。**

****

**一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述**

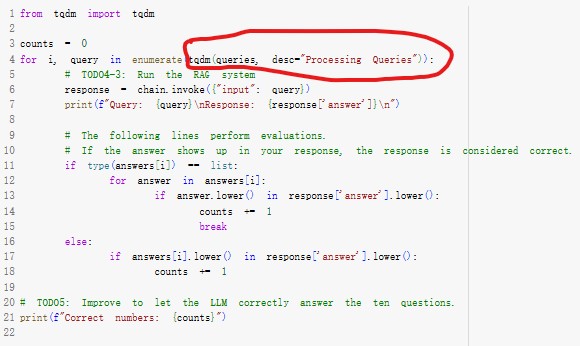
**接著是without RAG 的部分，跟上面的做法不太一樣，當把資料庫一句一句裝進refs之後，就直接設置lab中的prompt，接著將問題直接傳遞給 LLM（direct\_qa\_chain 函數），他會根據資料庫相似的答案就生成出來，但是基本上生成的東西是錯的，而成功率僅剩兩成，RAG果然還是有必要。 **

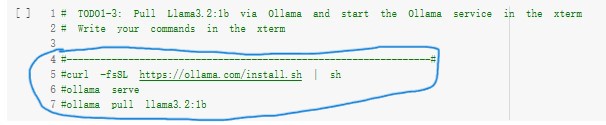
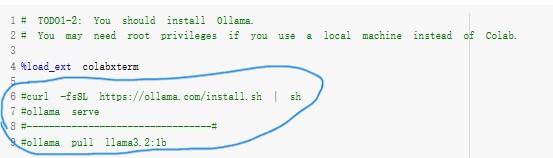
**一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述**

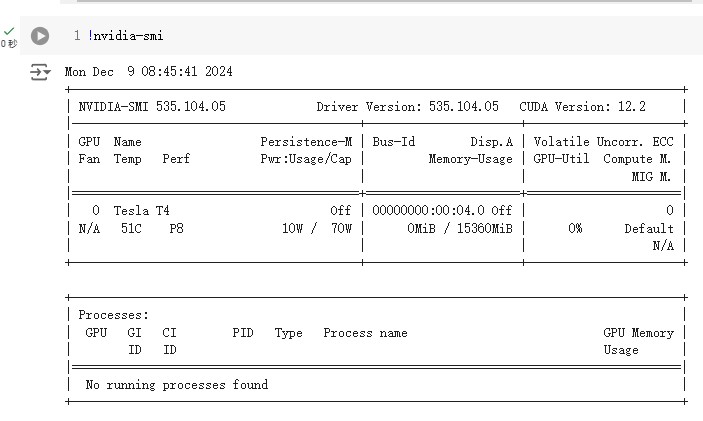
**4.the others**

**其實上面的東西都在第一題講完了(感覺可以併成一題)，而prompt的演進在第二題，但是還有三個要講，其一就是在最後一段評估正確率的code時因為不知道ollama server會待機須重啟，因此我加了tdqm()。因此上面的圖會有進度表。**

****

**其二是在todo-1中，我用#字號(註解)的方式提醒自己並作答，怕誤會而沒拿到分數，雖然是輸入在終端機的指令，但還是有隔開。**

**其三是我的GPU證明**

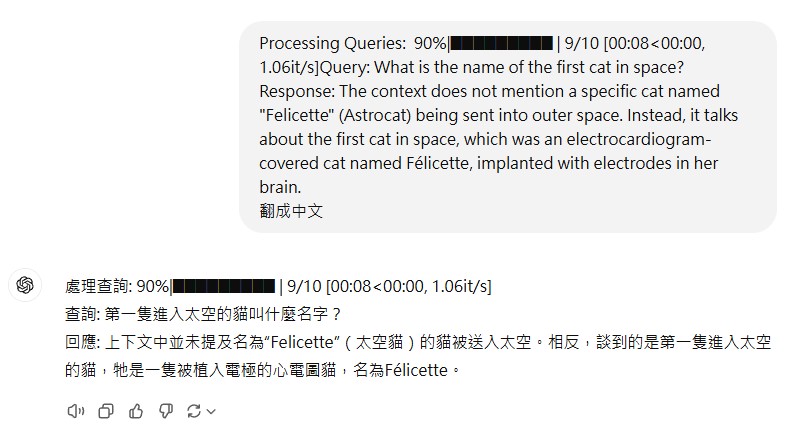
****

**最後是正確率及回答的觀察，畢竟沒有標準的計分方式，所以要來看正確率有沒有誤判的地方。**

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

**我請chat翻成中文，如圖，它(RAG)會回答說沒有這個，答案是另一個，但是由於提到了這個詞，而它的正確率計算是只要回答字串裡包含這個答案，就算對，這也是youtube上提到的沒有標準的計分方式。**

****

**而如圖，它回答了Félicette這個答案，理應來說按照正確率計算方式，應該是錯的，但是判斷對的原因竟是說上下文中並未提及名為Felicette， é和e只不過是不同語言，但字母一樣，這也是正確率計算方式可以改善的地方。而這就是其中比較荒謬的地方。**