

Day23 - 如何將內容向量化

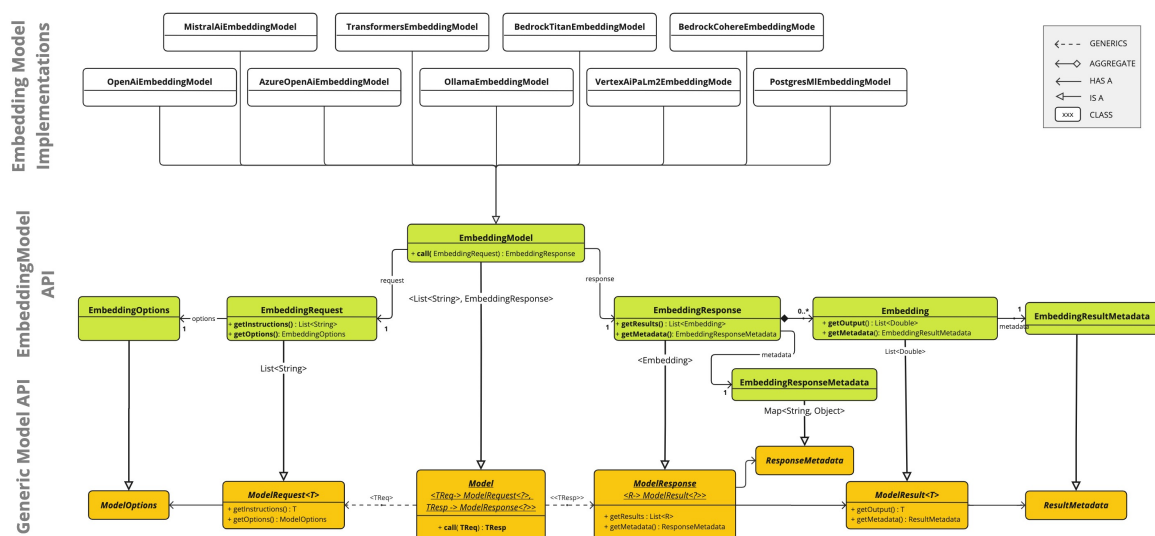
Embedding是塞了甚麼？

剛看到 Embedding 這個名詞一直無法跟向量化聯想在一起，直到看了台大教授陳繡儂的影片才瞭解其中含意，有興趣的朋友可自行看影片學習，這裡簡單說一下概念

Embedding 就是將內容轉成向量數據的一種表現方式，演算法會看有多少維度，依據每個 Token 與一定大小的上下文計算後再去跟 LLM 計算得到向量值，最後將向量數據塞回每個維度中，由於原本向量值是矩陣資料很浪費空間，上面的做法像是把矩陣資料塞進一個維度內，所以才會稱為 Embedding

這些 Embedding 數據有甚麼作用？其實這是向量資料庫用來做近似查詢的依據，也是 RAG 儲存資料跟檢索資料會用到的向量資訊，查詢時會把每個維度的向量利用餘弦相似度或是歐式距離法算出向量間的距離，求出結果最大的前 N 筆數據，最後在將這些資料傳給 AI 幫我們整合內容

接下來就看看 Embedding Model 的 UML



可以看出 Class 結構與前面的 ChatModel 差不多，只差在最後的結果是一堆 Embedding，我們來看看昨天存入向量數據庫的內容，下圖的 embedding 正是每個維度的向量值

```
neo4j$ MATCH (n:Document) RETURN n LIMIT 25
```



Graph



Table



Text



Code

n

1

```
{
  "identity": 0,
  "labels": [
    "Document"
  ],
  "properties": {
    "metadata.conversationId": "1",
    "id": "1cba8f0d-de66-45b6-a977-6deb411ec4a8",
    "text": "我是凱文大叔,以後對話請先稱呼我",
    "embedding": [
      -0.018121637403964996,
      -0.02415776252746582,
      -0.011150246486067772,
      -0.005744267255067825,
      -0.006228483747690916,
      0.023680178448557854,
      -0.009465438313782215,
      0.005429194774478674,
      -0.028867265209555626,
      -0.012828421778976917,
      0.00045892782509326935,
      0.002407816471531987,
      -0.004331416450440884,
```

程式實作

下面我們來寫個簡單的程式，測試一下如何將一段內容轉成 embedding

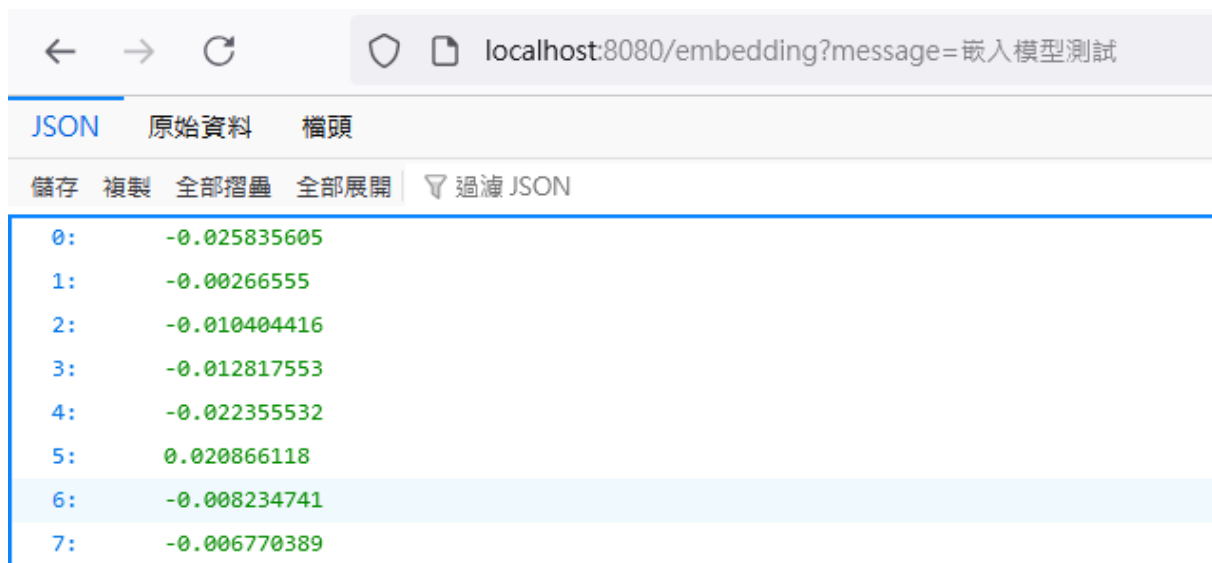
```
@RestController
@RequiredArgsConstructor
```

```

public class EmbeddingController {
    private final EmbeddingModel embeddingModel;
    @GetMapping("/embedding")
    public List<Double> embed(String message) {
        return embeddingModel.embed(message);
    }
}

```

驗證結果



← → ↺		localhost:8080/embedding?message=嵌入模型測試
JSON	原始資料	檔頭
儲存	複製	全部摺疊 全部展開 過濾 JSON
0:	-0.025835605	
1:	-0.00266555	
2:	-0.010404416	
3:	-0.012817553	
4:	-0.022355532	
5:	0.020866118	
6:	-0.008234741	
7:	-0.006770389	

至於 embedding 會有幾個呢？這取決於 Embedding Model 維度的大小，Open AI 預設的 Embedding Model 是 text-embedding-ada-002，它的維度是 1536，所以我們的結果也會有 1536 筆，可以用 `embeddingModel.dimensions()` 方法取得維度

實際上程式碼也是給一段文字取得 embeddings 後在看有多少 size

另外要注意的是向量資料庫儲存資料時除了內容跟 embeddings 外，metadata 也頗為重要，若沒將參考資訊放在 metadata 中，AI 最後只能彙總資料後回答你產生的內容

拿昨天的程式為例最後寫進向量資料庫的內容是下面程式，也就是說我們寫入向量資料庫前將訊息跟 metadata (Map 格式) 合成 Document，這樣向量資料庫就能包含特定資訊

```

this.getChatMemoryStore().write(toDocuments(assistantMessages

private List<Document> toDocuments(List<Message> messages, St
    List<Document> docs = messages.stream()
        .filter(m -> m.getMessageType() == MessageType.USER |
        .map(message -> {
            var metadata = new HashMap<>(message.getMetadata(
            metadata.put(DOCUMENT_METADATA_CONVERSATION_ID, c
            metadata.put(DOCUMENT_METADATA_MESSAGE_TYPE, mess
            var doc = new Document(message.getContent(), meta
            return doc;
        })
        .toList();
    return docs;
}

```

思考一下向量資料庫還能加入甚麼內容

聊天的訊息就是加入了 **conversationId**，這樣我們就能透過 conversationId，篩選內容

RAG 的資料則會包含檔案名稱，這樣查詢時還能列出原始檔案

若 RAG 的資料是從網路爬來的，可以加上 URL，之後能追蹤來源

從企業內部系統整合來的資料則可以加上系統以及資料 ID，之後可快速連結到內部系統