國立臺灣大學工學院工程科學及海洋工程學系

碩士論文

Department of Engineering Science and Ocean Engineering

College of Engineering

National Taiwan University

Master Thesis

在語境學習中研究各種提示以改善 Text-to-SQL 任務分解

Exploring Diverse Prompts to Improve Text-to-SQL Task Decomposition in In-Context Learning

林怡萱

Yi-Hsuan Lin

指導教授：黃乾綱 博士

Advisor: Chien-Kang Huang, Ph.D.

中華民國 112 年 7 月

July, 2023

doi:10.6342/NTU202303187

誌謝

感謝指導教授黃乾綱老師的教導，不惜利用額外的時間替我們實驗室的同學們加強機器學習的觀念，也在我們實驗卡住的時候，提供我們不同的角度去想出解決方法；心情不好的時候，也會持續鼓勵我們，告訴我們不用想那麼多，產出論文沒有那麼難。感謝我爸媽，幫我繳學費、房租費，讓我在台北好好生活，每次回家都能一起吃好料，享用美食，以及與家人一起生活的美好時光。感謝仙惠，我們一起修了很累很煩的課，也一起做計畫，一起當助教，一起為實驗及論文煩惱，心情好或心情壞都有你陪著，聚餐和運動也有你一起，以後要吃好料，還是可以揪我的，超愛你的唷～感謝子傑，幫我處理很多雜七雜八，當助教有夠罩！感謝敬翔，實驗室報帳都由你負責，我這個財務不知道幹嘛用的，然後你才是真資工，我是假資工。感謝瑞益實驗室的同學們（荃伊、雅芳、丞彥、孟哲），常常一起吃飯、聊天，偶爾約出去喝酒，還唱KTV，因為有你們，才使得所有的聚會變得如此有趣，在此許願畢業以後，大家還是約得出來，一起敘敘舊、嗨到發瘋。感謝煒智每個禮拜跟我一起打羽球，還借我羽球拍，甚至就沒拿走，送給我了，我會用你的羽球拍繼續打下去的。感謝系籃學妹們，讓我這臭學姊能夠跟你們一起練球，後期常常人都不到，永遠都在吃老本打球，感謝你們的不嫌棄。感謝活大，特別是御喜自助餐，供了我幾乎來學校每一天吃的餐。感謝後期固定一起吃飯的飯友 Mary，幫我撐傘撐過了無數個炎熱的天氣，以及每次都吃 50 元素食，多盛一堆胚芽飯讓我可以吃免錢的飯，還有偶爾帶我離開學校去吃好吃的食物，能遠離學校吃美食的感覺真好。感謝耿先生，讓我在很友善的環境下打工，還不時地有早餐、零食、偶爾的披薩炸雞可以吃，待你那越久，身材越走鐘，當然，不只有食物，我也學到了很多技能，像是做網路線和打雜。感謝鈺文，常跑來系館跟我和煒智他們一起打羽球，偶爾也一起打籃球、重訓，還有跟我一起聊天，看我論文很忙，還給我一堆蛋糕、餅乾吃，讓我好幸福～感謝昱漙，知道我忙，所

doi:10.6342/NTU202303187

以常常是你從基隆來台北找我，心情不好時找你，你都會及時出現陪我，上次喝酒還照顧我，學姊失職了真是抱歉。感謝阿傅阿莉，久久一約都能從白天聊到晚上，從開店聊到關店，與你們分享我碩士生活的大小事，都能讓我輕鬆許多。感謝前男友，也抱歉變成了前男友，但你在我繁忙的時候，願意承擔我的情緒、脾氣，偶爾還要幫我一些跟你科系無關的資工作業，我真是感動不已。感謝庭訓、胤融，回桃園揪吃早餐很好揪，還都會配合我，想在哪吃想幾點吃都讓我決定，在我開心或不開心的時候都一個訊息就約好時間吃飯聊天，雖然一直罵我公主病但還是願意禮讓著我。在這裡也謝謝碩一下時，整天在我旁邊念我嘮叨我的查理，有你才讓我更有勇氣跟陌生人講話、問話，以及讓我懂更多我原本應該就該懂的資訊知識，一起參加減脂比賽時，在我想偷懶時也拉著我去運動，但我還是要說，你很機車。還要感謝的人好多，感謝大家陪著我這又辛苦又快樂的碩士生活，讓我不是一個人與學業、論文孤軍奮戰，love everyone～

doi:10.6342/NTU202303187

摘要

在面臨大量的查詢資料表需求時，減少人工撰寫 SQL 指令的工作量成為一個重要課題。因此，Text-to-SQL 任務逐漸受到學術界及業界的關注。起初，解決 Text-to-SQL 任務的方法主要是使用 seq2seq 模型；然而，隨著大型語言模型的迅速發展，近年來一些研究人員開始在大型語言模型進行語境學習，以評估其性能。日後，思維鏈（chain-of-thought, COT）技巧的有效性也受到廣泛關注，促使各領域嘗試將其應用於觀念分解及推理過程，其中 DIN-SQL 是個典型的例子。然而，即使它在 Spider 資料集執行準確率（Execution Accurancy, EX）排行榜上表現出色，但在模式連結的處理上仍有不足之處。

本研究主要目的是針對 DIN-SQL 模型進行優化。首先進行錯誤分析，探討其預測失敗的原因，並提出改善性能的提示（prompt）方法。我們使用 GPT-3.5 模型進行測試，並提出了三種改善方向，分別是針對模式連結、分解與分類、與自我修正模組進行改善。實驗結果表示，這三種改善方向都成功提升模型性能。其中，針對模式連結的改善方式——將 RESDSQL 交叉編碼器與 DINSQL 自我修正模組相結合——被證明是最佳的改善方法，與基線模型相比，這種改進方法可以提高 3.82%的性能，應用 GPT-3.5 與 GPT-4 模型下，分別達到 74.1%與 79.79%的準確率。

關鍵字：Text-to-SQL、深度學習、大型語言模型、思維鏈

doi:10.6342/NTU202303187

Abstract

When faced with a large number of query data tables, reducing the workload of manually writing SQL commands has become an important topic. Therefore, the text-to-SQL task has gradually attracted the attention of academia and industry. At first, the method to solve the text-to-SQL task was mainly to use the seq2seq model; however, with the rapid development of large-scale language models (LLMs), some researchers have begun to perform in-context learning on LLMs to evaluate their performance in recent years. After that, the effectiveness of the chain-of-thought (COT) technique has also received widespread attention. It makes more researchers from various fields to try to apply it to the concept decomposition and reasoning process, of which DIN-SQL is a typical example. However, even though it performed well on the Execution Accuracy (EX) leaderboard of the Spider dataset, it still has shortcomings in the processing of schema linking.

The main purpose of this study is to optimize the DIN-SQL model. Firstly, error analysis is carried out to explore the reasons for its prediction failure, and a prompt method to improve performance is proposed. I used the GPT-3.5 model for testing, and proposed three improvement directions, namely for schema linking, decomposition and classification, and self-correction modules for improvement. The experimental results show that all three improvement directions have successfully improved the model performance. Among them, the improvement method for schema linking, combining the RESDSQL cross-encoder with the DINSQL self-correction module proved to be the best improvement method. This method can improve performance by 3.82% compared with the baseline model. When using the GPT-3.5 and GPT-4 models, the EX accuracies were 74.1% and 79.79%, respectively.

doi:10.6342/NTU202303187

Keywords: Text-to-SQL, Deep Learning, Large Language Model, chain-of-thought

doi:10.6342/NTU202303187

目錄

doi:10.6342/NTU202303187

doi:10.6342/NTU202303187

doi:10.6342/NTU202303187

圖目錄

doi:10.6342/NTU202303187

表目錄

doi:10.6342/NTU202303187

doi:10.6342/NTU202303187

第一章　　緒論

1.1　研究背景及動機

隨著資訊化社會的發展，利用資料庫儲存大量數據已是大多數網站或系統所採用的技術。現今關聯式資料庫成為主流，而 SQL（Structured Query Language, SQL）指令則是其主要的數據訪問方式。雖然熟練的專業人員可以自行完成 SQL 查詢，並有效地操作這些數據表，但在針對多變的使用場景，撰寫時要確保 SQL 指令的正確性，且能快速處理大量數據，在目前仍然具有一定難度。為能更高效地產出多樣的 SQL 指令，將自然語言（National Language, NL）問題轉換為機器可執行的 SQL 指令，日漸引起關注。這種轉換使得任何人僅透過簡單的問題描述，便能毫不費力地查詢表格以取得最正確的資訊，並且在現實生活中應用，例如：智能客戶服務、問答或機器人導航等。因此，Text-to-SQL（或 NL-to-SQL）系統在學界與業界均越來越受重視。

由於人們日趨關注 Text-to-SQL 任務，部分研究者陸續發布了應用於此任務上的資料集。早期有單回合資料集如 GeoQuery [19]、WikiSQL[30]，此類資料庫有下列問題：數據集規模小、問題不夠複雜和不符合真實使用情境等；在 2018 年，耶魯大學發布了 Spider[24]——單回合、多資料庫和多表格——資料集，解決了上述的處境。此後，開始有大量的研究者（[3, 11, 22]等人）以 Spider 資料集作為訓練對象，製造出更適合實際應用的模型。

先前研究多採用微調的 seq2seq 模型來處理 Text-to-SQL 任務；然而，近年來，由於大型語言模型（Large Language Model, LLM）迅速發展，陸續有一些研究者開始用它來解決此任務，如：[12, 14, 17]使用 zero-shot 及 few-shot 下提示（prompt）。前述方法的優點是，使用的資料較少，且無須花時間微調模型，就可以獲得不錯的結果；但與大多數微調模型相比，此技術仍有可進一步改進的空間，如表 1 所示，與目前性能最佳的 seq2seq 模型——RESDSQL[9]相比，仍差了 19.6%的準確率。2023 年 3 月底，GPT-4[13]——最先進（state-of-the-art,

doi:10.6342/NTU202303187

SOTA）語言模型——的出現為 Text-to-SQL 任務帶來了突破性的發展：DIN-SQL[14]提出了一種基於 few-shot 提示的新方法，將該任務分解為多個步驟，並利用 GPT-4 與 CodeX Davinci 評估其性能，成功在 Spider[24]資料集的執行準確率排名取得第一及第五位，成為目前最先進的模型；然而，論文中提及，在模式連結（schema linking）——自然語言問題與模式項（schema item，也就是表格及欄位）之間的連結——處理上表現有待改善。

表 1：Spider[23]開發集中大型語言模型與最先進的微調模型方法性能比較圖。RESDSQL-3B+NatSQL[9]為當時在測試集執行準確率中第一名的模型。

方法	模型	執行準確率（EX）（%）
微調模型方法	RESDSQL-3B+NatSQL（[9]）	84.5
	T5-3B+PICARD（[18]）	79.3
大型語言模型方法	GPT-4 Zero-shot（[14]）	64.9
	GPT-4 Few-shot（[14]）	67.4
	CodeX Zero-shot（[17]）	55.1
	CodeX Few-shot（[17]）	61.5

1.2 研究目的

如上一節所述，儘管 DIN-SQL[14]在 Spider[24]資料集的執行準確率上已成為最先進的模型，並在測試集上獲得了 85.3%；然而，在模式連結方面仍存在著明顯的不足。根據論文的分析，作者選取了 100 個抽樣的錯誤預測結果，超過 50%的錯誤是由於模型連結錯誤造成的。此外，在 group by 或 join 的分類上，錯誤率也達到了 10%以上。為了提高性能，本研究視 DIN-SQL 為基線模型，並根據其可能出現的不同錯誤情況，提出了四種改善方向。同時，再針對其策略方向，進一步思考提示的設計。過程將使用 Spider[24]資料集進行評估。

1.3 研究貢獻

本研究在 Text-to-SQL 任務上的主要貢獻如下：

doi:10.6342/NTU202303187

一、提出了多種提示方法，有效提升最高 3.5%性能：本研究根據三個優化方向，提出多種基於 GPT-3.5 模型的改善方法，其中表現最優異的模型可以有效提升 3.58%的性能，使用 GPT-3.5 與 GPT-4 模型上達到 74.5%與 79.79%的準確率。

二、實驗結果錯誤分析與比較：本研究對實驗結果進行了詳細的分析和比較，包括改進方法的效果、性能指標的提升程度以及模型在不同子任務中的表現。這些分析結果提供了對改進方法的有效性和可行性的評估。

三、探索 GPT-3.5 模型在此任務的應用潛力：從 Aiwei Liu 等人提供的 zero-shot[12]結果，以及本研究再執行的結果，可以觀察到 GPT-3.5 在這一任務中有著越來越好的表現趨勢。這有助於深入了解 GPT-3.5 模型在自然語言處理任務中的優勢和潛力。

1.4 論文架構

本論文的目標為研究各種提示方法，以提升大型語言模型在 Text-to-SQL 任務上之性能表現。第一章為緒論，背景的部分會說明 Text-to-SQL 任務的重要性，以及前人在該任務上開發的資料集和使用的模型趨勢。同時，說明本研究之動機、目的和貢獻。第二章為相關背景知識與文獻回顧，其中會介紹常見的資料集，以及應用在 Spider[24]資料集上的模型。此外，還將介紹大型語言模型提示的概念，並特別提及本研究所參考的重要模型。第三章為問題定義與研究方法，問題定義會說明 Text-to-SQL 任務本身問題，以及目前面臨到的挑戰。此外，將進行錯誤分析，探討預期的改善方式，以及多種不同的提示研究方法。第四章是研究結果與討論，會說明本研究所使用的模型配置與資料集。在實驗結果中，將對每種改善方式進行比較，並討論觀察到的結果。第五章是結論與未來展望，在此會總結本研究的結果，並探討 text-to-SQL 任務未來的方向。

doi:10.6342/NTU202303187

第二章　　相關背景知識及文獻

本章節會回顧與 text-to-SQL 任務相關的資料集，以及應用於此任務的模型。這些模型可以分成兩大類：seq2seq 模型和大型語言模型。而此部分的介紹會著重在以 Spider[24]資料集做為評估指標的模型。為了更了解提示（prompt）常見的形式，本研究將特別討論運用在大型語言模型中的語境學習（in-context learning）技術。最後，將探討在兩大類型模型方法中兩個重要的參考對象——RESDSQL 與 DIN-SQL 的性能表現和架構設計。

2.1 Text-to-SQL 資料集

在 Text-to-SQL 任務中資料集可分成兩種類型：單回合及多回合。單回合資料集與上下文無關；多回合資料集與上下文相關，問題內可能會有省略及回指，也就是當前問題可能與前次問題所提及之內容相關。

單回合資料集常見的有 WikiSQL[30]和 Spider[24]等，WikiSQL 資料集是在 2017 年提出的，是目前規模最大的英文資料集，問題多為簡易的 SQL 查詢，不包含 order by、group by 或 join 等複雜指令。Spider 資料集是在 2018 年提出的，數據量不比 WikiSQL，但在 SQL 查詢的變化上更符合真實使用情境，使用了多種複雜指令，如：group by、having 或巢狀（Nested SQL）架構。

常見的多回合資料集有 SparC[25]和 CoSQL[23]。SparC 是個基於 Spider 資料集擴展成具有上下文對話的資料集，將單一問題延伸至一系列的連續問題。CoSQL 同樣也是基於 Spider 資料集生成的，但取的問題與 SparC 不完全相同，並且還增加了系統確認機制，在每一次問答中除了回覆 SQL 查詢結果外，也會給予系統所理解的內容。另外，當使用者的問題不夠明確時，系統也會進行確認，而不是在不清楚的狀態下勉強地給予 SQL 查詢結果。

為了能讓模型能在更符合真實情境下有良好的表現，本研究選擇使用單回合的 Spider 資料集進行訓練與驗證。接下來的小節將詳細介紹此資料集的特

doi:10.6342/NTU202303187

點，以及目前在排行榜上的模型現況。這將有助於更好理解本研究所面對的挑戰和現有模型的性能水平。

2.1.1 本研究使用的 Spider 資料集

Spider[24]資料集由 11 位耶魯大學學生註釋而成，具有多樣的 SQL 指令風格。它是個單回合類型的對話，其特點包括涵蓋多個資料庫與表格，並且跨越多個領域。因此，Spider 被廣泛認為是檢測模型泛化能力最有效的資料集。Spider 具備 10,181 個問題和 5,693 個不重複的 SQL 語句，並擁有 200 個資料庫，其中包含 1020 個表，範圍涵蓋了 138 個領域。Spider 資料集根據 SQL 查詢的難易度分為四個級別：簡單（easy）、中等（medium）、困難（hard）和極難（extra hard）。這些級別可用來評估模型在不同難易度的 SQL 查詢之性能。Spider 資料集被拆分為訓練集、開發集與測試集。訓練集包含 8659 個問題和 146 個資料庫；開發集包含 1034 個問題和 20 個資料庫；測試集包含 2147 個問題和 40 個無法預見的資料庫，數據統計如表 2。測試集的具體內容是不公開的。

表 2：Spider 資料集的統計數據[24]。

	問題樣本數	資料庫數
訓練集	8,659	146
開發集	1,034	20
測試集	2,147	40

評估 Spider 資料集常用的指標有兩種：執行準確率（Execution Accuracy, EX）和精確匹配率（Exact Matching, EM）。在 Spider[1]頁面排行榜中，使用這兩種指標分別進行排名。在 Text-to-SQL 任務中講求的是，根據問題與資料庫資訊，所獲得的查詢結果，而執行準確率（EX）即是符合這一目標的評估指標。

[1] https://yale-lily.github.io/spider

doi:10.6342/NTU202303187

因此，現今越來越多研究專注於提升執行準確率的分數，而對於關鍵字等是否精確匹配的指標（EM）較少關注。

在執行準確率（EX）排行榜中，前兩名是近兩個月提的，應用於大型語言模型的方法，也是目前的主要趨勢。而後續的排名，除了第五名同是大型語言模型方法外，皆是 seq2seq 模型。排行結果如表 3 所示。

表 3：spider 執行準確率（EX）排行榜。

排名	模型	測試集準確率
1	DIN-SQL + GPT-4 [14]	85.3
2	C3 + ChatGPT	82.3
3	RESDSQL-3B + NatSQL [9]	79.9
4	SeaD + PQL	78.5
5	DIN-SQL + CodeX [14]	78.2
6	CatSQL + GraPPa	78.0
7	Graphix-3B+PICARD [10]	77.6
8	SHiP+PICARD [28]	76.6
9	N-best List Rerankers + PICARD [26]	75.9
10	RASAT+PICARD [15]	75.5

2.2 應用於 Text-to-SQL 任務的 Seq2seq 模型

在 Spider[24]排行榜中前十名的位置，Seq2seq 架構模型就佔據了七成。Seq2seq 模型通常由編碼器（encoder）和解碼器（decoder）組合而成，這個架構的好處是可以將任務切分成子任務處理：編碼器負責對自然語言文本進行編碼，理解問題與資料庫模式之間的關係；解碼器負責將編碼後的結果進行解碼，學習 SQL 語法和結構，從而產生出符合 SQL 語法的查詢語句。

編碼器可分為有無使用圖神經網路，有使用的模型有：RAT-SQL[20]、LGESQL[2]及 ShadowGNN[3]，此類模型會先根據資料表、欄位或問題等之關係表示為圖形結構，以讓模型學習資訊間的關聯性。例如：RAT-SQL[20]會將欄位間、資料表間及兩者建立連結；LGESQL[2]則是參考 RAT-SQL 的作法，新增了兩項功能：一、增加了問題節點與其他節點關係，以及 local 及 non-local

doi:10.6342/NTU202303187

的概念，也就是若兩者關係長度為 1，則為 local，若兩者關係之間多於一個節點串接，則為 non-local。二、新增了對偶圖（line graph），為 local 的節點創建一個以邊為中心的圖。另外，仍有部分模型不使用圖神經網路架構也同樣達到不錯的性能，如：RESDSQL[9]和 T5-PICARD[18]將所需資訊字串連接，用簡易的分隔線「｜」來區別每筆資訊；RESDSQL 將輸入變成像以下形式：$X = q \mid t_1 : c_1^1, \ldots, c_{n_1}^1 \mid \ldots \mid t_N : c_1^N, \ldots, c_{n_N}^N$，其中$X$為輸入，$q$為問題，$t_n$為第$n$個表，$\mathrm{c}_{\mathrm{n_N}}^{\mathrm{N}}$為表$N$的第$n_N$個欄位。

在解碼器方面，當前研究者主要採用中間表示（Intermediate Representation, IR）、PICARD[18]和抽象語法樹（Abstract Structure Tree, AST）方法。中間表示的做法是將 SQL 語法簡化並統一為相同的架構，主要用於解決自然語言問題與 SQL 查詢之間存有的差距，並減輕轉換過程的負擔，例如：在問題中並無任何詞彙會具體描述 GROUP BY 關鍵字，故在中間表示當中會直接省略和處理它，如圖 1。比較著名的兩種中間表示是 SemQL[7]和 NatSQL[4]，它們在[5, 9, 14]等人研究中被使用以提高模型性能。PICARD[18]方法是為了避免模型在生成 SQL 語法時產生無效的語句，在生成途中增加了檢查的步驟，若有不合法的語句出現，則會立刻被拒絕，以防止錯誤的指令繼續生成。此一方法現已被[10, 26, 28]等人結合使用。另一個抽象語法樹是早期在 PICARD 還沒釋出前，大部分研究（[2, 20]等）會使用的方法。該方法是在生成 SQL 時直接照著既定的架構下生成，以防止產生違反規範的語法。

doi:10.6342/NTU202303187

Question：
Which film has more than 5 actors and less than 3 in the inventory?

SQL：
```
SELECT T1.title FROM film AS T1 JOIN film_actor AS T2 ON T1.film_id = T2.film_id GROUP
BY T1.film_id HAVING count(*) > 5 INTERSECT SELECT T1.title FROM film AS T1 JOIN
inventory AS T2 ON T1.film_id = T2.film_id GROUP BY T1.film_id HAVING count(*) < 3
```

The IR of RAT-SQL：(Remove the JOIN ON Clause)
```
SELECT title FROM film, film_actor GROUP BY film_id HAVING count(*) > 5
INTERSECT SELECT title FROM film, inventory GROUP BY film_id HAVING count(*) < 3
```

The IR of SyntaxSQL：(Remove the JOIN ON and FROM Clause)
```
SELECT film.title GROUP BY film.film_id HAVING count(*) > 5 INTERSECT
SELECT film.title GROUP BY film.film_id HAVING count(*) < 3
```

SemQL：(Remove the JOIN ON, FROM and GROUP BY Clause. Merge the HAVING and WHERE clause)
```
SELECT film.title WHERE count(film_actor.*) > 5  INTERSECT
SELECT film.title WHERE count(inventory.*) < 3
```

NatSQL：(Further remove the set operators based on SemQL)
```
SELECT film.title WHERE count(film_actor.*) > 5 and count(inventory.*) < 3
```

圖 1：NatSQL 與其他不同規則的中間表示比較圖[4]。

本研究計劃採用 RESDSQL 的編碼器部分，該部分生成的內容以文本形式呈現而不是向量，這種特性非常適合應用於提示輸入。

2.3 應用於 Text-to-SQL 任務的大型語言模型（Large Language Model, LLM）

與 seq2seq 模型相比，應用大型語言模型於 Text-to-SQL 任務時，無需預訓練或微調模型，也不需使用大量資料集，只要下適當的提示，就能在各種自然語言處理任務中表現出色。在 text-to-SQL 任務中，[12, 17]等人利用大型語言模型對 Spider[24]資料集進行了 zero-shot 和 few-shot 的實驗，以評估它們在 Text-to-SQL 任務上的性能。[17]使用了 GPT-3 和 CodeX 與當前流行的 T5[16]模型做比較，在當時參數量最高的 Codex davinci 的實驗之下，分數已到達一定的水準，如表 4。[12]則使用 ChatGPT 進行了 zero-shot 的實驗，如表 5。以上兩項研究皆能證明，在 text-to-SQL 任務中，僅使用 zero-shot 及 few-shot 就能有顯著的性能。

doi:10.6342/NTU202303187

表 4：各模型在 Spider[24]測試集上 zero-shot 實驗之性能比較表[17]。其中評估指標有有效 SQL 查詢（valid SQL, VA）、執行準確率（execution accuracy, EX）和測試套件準確率（test-suite accuracy, TS）。大型語言模型根據規模大小，性能也由低到高。

Model	VA	EX	TS
Finetuned			
T5-base	72.7	57.9	54.5
T5-large	84.1	67.2	61.4
T5-3B	87.6	71.4	65.7
T5-3B*	88.2	74.4	68.3
T5-3B + PICARD*	97.8	79.1	71.7
BRIDGE v2*	–	68.0	–
Inference-only			
GPT-3 ada	33.8	2.3	0.3
GPT-3 babbage	48.8	5.7	3.9
GPT-3 curie	70.9	12.6	8.3
GPT-3 davinci	65.0	26.3	21.7
Codex cushman*	86.3	63.7	53.0
Codex davinci*	91.6	67.0	55.1

表 5：在 Spider 開發集，ChatGPT 與當前最先進模型在 zero-shot 實驗上之性能比較圖[12]。

Methods / Datasets	SPIDER		
	VA	EX	TS
T5-3B + PICARD	98.4	79.3	69.4
RASAT + PICARD	98.8	80.5	70.3
RESDSQL-3B + NatSQL	99.1	84.1	73.5
ChatGPT	97.7	70.1(14↓)	60.1

為了評估所設計的提示對效能提升的有效性，本研究將會把實驗結果與在 ChatGPT[12]上的 zero-shot 學習的結果相比較。

2.4 語境學習（In-Context Learning）

隨著大型語言模型的普及和發展，語境學習的概念由此誕生。它是一種無需對模型微調的提示技巧，可分為三類：zero-shot、one-shot 和 few-shot。這三類是根據輸入的樣本數量來劃分的。若在不給予任何示例的情況下要求模型產生結果，稱為 zero-shot；若僅提供一筆示例，則為 one-shot；若給予多筆示

doi:10.6342/NTU202303187

例，則是 few-shot。在 zero-shot 設置的提示下，可以獲得大型語言模型在指定任務中的下限表現[1, 6, 21, 27]。

2.5 Spider 資料集執行準確率最先進模型——「DIN-SQL + GPT-4」

DIN-SQL[14]於今年 4 月時發布，在 Spider 排行榜頁面中應用 GPT-4[13]模型後，執行準確率（EX）中獲得了最高分，達到了 85.3%。

該研究採用了思維鏈[21]策略，將 Text-to-SQL 任務分解成多個子問題，並將整個生成過程分成四個階段：一、提取相關的資料庫表格及欄位。二、分析並分類 SQL 結構。三、根據分類結果，個別處理 SQL 查詢。四、修正 SQL 查詢。這四個階段對應的模組分別為：一、模式連結（schema linking）。二、查詢分類和分解（Classification & Decomposition）。三、SQL 生成（SQL Generation）。四、自我修正（self-correction），整體架構如圖 2 所示。每個模組內部的運作方式如圖 3 所示。每個模組都設計了相應的提示，以及前一次模組輸出的結果。兩者結合後對 GPT 下提示，就會得到系統回應。此回應的輸出結果便成為下一個模組的輸入，這樣的循環進行下去。

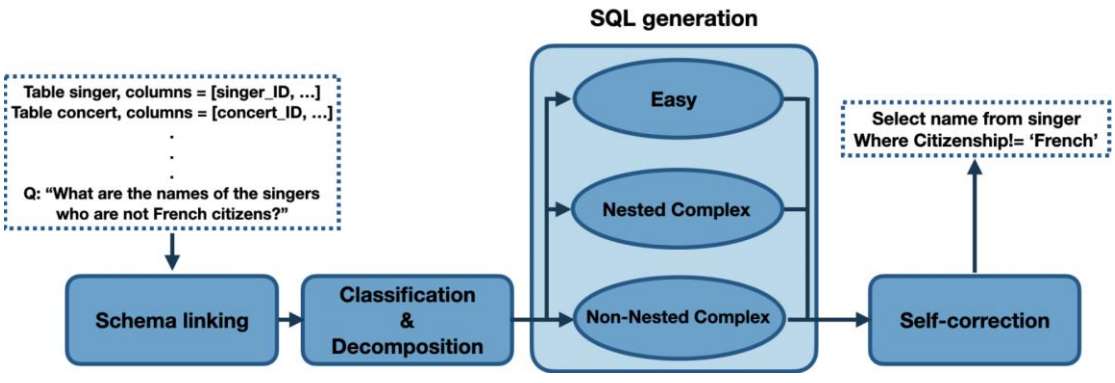

圖 2：DIN-SQL 模組架構圖[14]。

doi:10.6342/NTU202303187

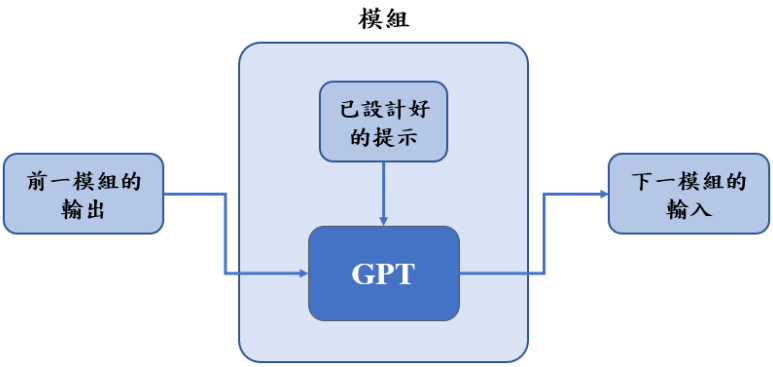

圖 3：模組內部運作模式示意圖。

模式連結模組旨在識別自然語言問題與資料庫模式項和條件值之間的引用，此步驟有助於實現跨領域的泛化性和處理複雜的 SQL 查詢[8]，也成為 Text-to-SQL 任務中必要的步驟之一。

分類和分解模組主要是想解決不同表格之間 join 時可能出現的錯誤，以及處理包含巢狀（nested query）結構的查詢。因此，查詢分成三種類型：簡單類（easy）、無巢狀類（non-nested）和巢狀類（nested）。簡單類即是僅含單表格查詢；無巢狀類需要多個表格，但不包含巢狀架構；巢狀類可能涉及多個表、巢狀查詢和集合（set operation）操作（如：UNION 等）。

SQL 生成模組會繼承前一個模組的分類結果，並制定不同的提示格式，目的是為了生成不同難度問題所需的 SQL 查詢。相較於簡單類，無巢狀與巢狀類會增加一個中間表示，再輸出 SQL 指令。

自我修正模組主要用於修正生成的 SQL 可能缺少或產生多餘的關鍵字，因而提出兩種不同的提示策略：generic 和 gentle。Generic 提示是直接請求模型識別並更正 SQL；gentle 提示則不會假設 SQL 查詢存在錯誤，而是提供有關需要檢查的子句的信息，要求模型檢查是否有需要修正的部分。根據論文中的說明，gentle 提示較適合用在 GPT-4 模型；generic 提示則適合用在 CodeX 模型。

然而，論文中提及，它在識別問題對於欄位名、表名或實體實例（在文中稱為 schema linking）引用方面，仍然存在著相當數量的失敗案例。本研究將針對此問題進行改善。

doi:10.6342/NTU202303187

2.6 Spider 資料集執行準確率最先進的 seq2seq 模型——「RESDSQL-3B + NatSQL」

RESDSQL-3B + NatSQL[9]是目前在 Spider 數據集[24]執行準確率（Execution Accuracy, EX）分數中第三名的模型，在 2023 年 2 月剛釋出時，它被視為最先進（SOTA）模型，並在接下來的兩個月中獲得了第一名的位置。該模型在驗證集達到了 81%及測試集達到了 79.9%的性能。

作者提出一個排名增強的編碼器和骨架感知的解碼器框架。具體來說，編碼器的部分並不會將所有模式項作為輸入，而是只考慮最相關的模式項。為此，作者設計了一個交叉編碼器（cross-encoder），目的是對表與欄位進行分類，並根據它們出現的機率進行排名，而後將根據排名順序過濾不重要的模式項，如此即能減輕 SQL 解析過程中的模式連結（schema linking）工作量，更容易捕獲重要資訊。在編碼之前，使用了 RoBERTa[31]將每個模式項標記為一到多個 token，讓表名及欄位名更易理解。在解碼器方面，使用了 T5-3B[16]作為預訓練模型進行微調，並引入了 NatSQL[4]中間表示技巧，生成該問題解答的 SQL 框架，再從輸入序列中選擇所需數據（即表、欄位或值）來填充框架中的插槽。完整架構如圖 4。

doi:10.6342/NTU202303187

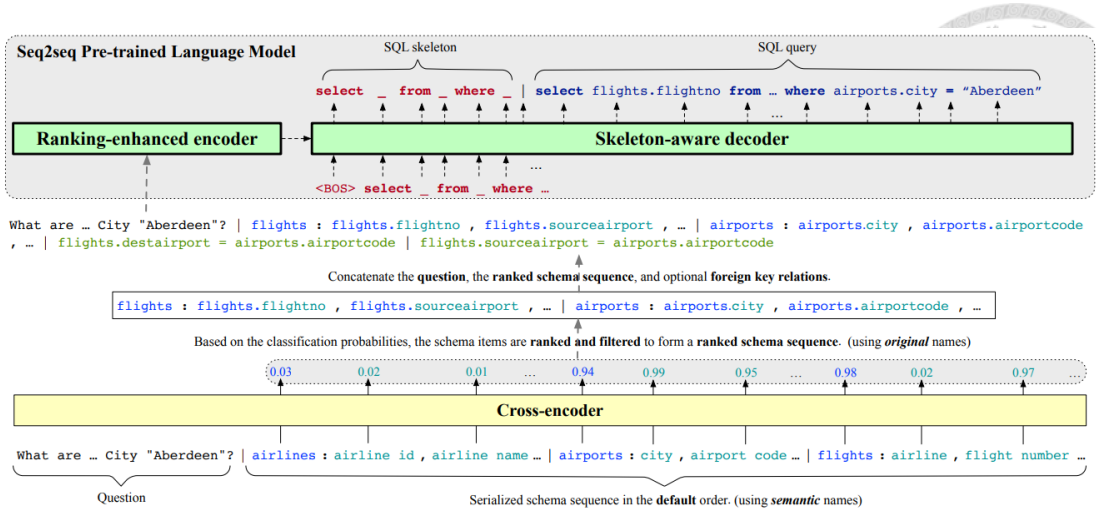

圖 4：RESDSQL 架構圖[9]。

由於 RESDSQL 在處理編碼器模式連結方面表現出色，能夠有效解決問題及資料庫模式項之間的引用，因此，本研究將參考交叉編碼器模式連結的結果，以改善 DIN-SQL[14]模型存在的不足之處。

doi:10.6342/NTU202303187

第三章　　問題定義與研究方法

本章節首先說明在 text-to-SQL 任務中的問題定義、以及在大型語言模型上執行該任務的操作流程。同時，也將探討目前在這個任務中所面臨的問題。接下來，會進行基線模型的錯誤分析，並根據分析結果提出優化的方向。根據這些優化方向改進提示策略。

3.1　問題定義

Text-to-SQL 任務為，提供一個自然語言問題Q以及相應的資料庫模式$S = < T, C, R >$，表格$T = \{t_1, t_2, \ldots, t_N\}$，$N$為表格總數，欄位$C = \{c_1^1, c_2^1, \ldots c_{n_1}^1, c_1^2, c_2^2, \ldots c_{n_2}^2, c_1^N, c_2^N, \ldots c_{n_N}^N\}$，上標為第$i$個表格，下標為第$i$個表格中第$j$個欄位。外鍵（foreign key）關係$R = \{(c_j^i, c_l^k), c_j^i, c_l^k \in C\}$，$(c_j^i, c_l^k)$是指欄位$c_j^i$與$c_l^k$有外鍵關係。最終我們的目的是求出一個 SQL 查詢Y，示意圖如圖 5。

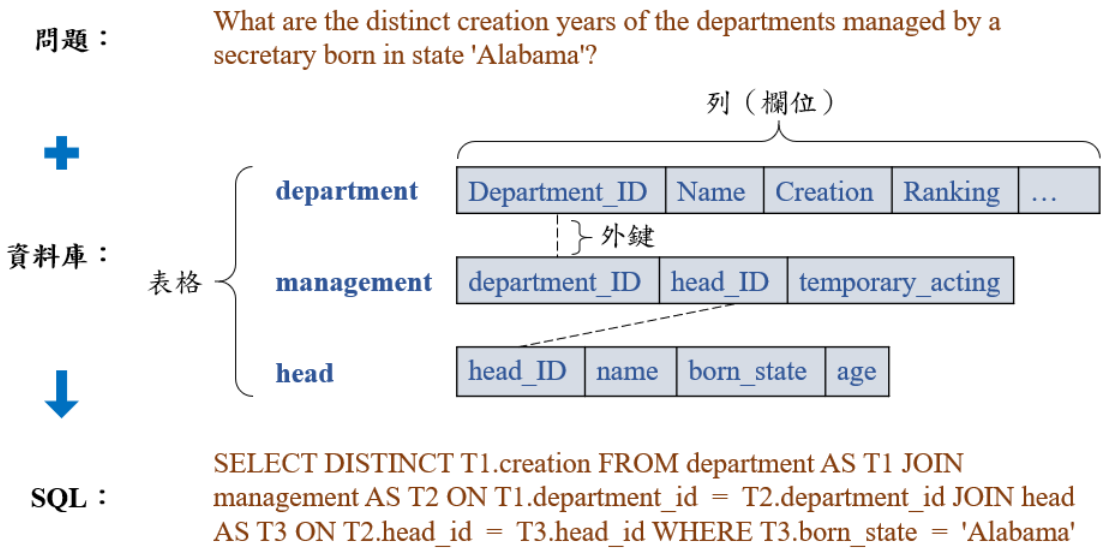

圖 5：Text-to-SQL 任務示意圖。

Text-to-SQL 任務在過去幾年已經有了相當的發展。早期的解決方案主要使用 seq2seq 模型，但近期的研究趨向於採用大型語言模型，並且結合提示工程（prompt engineering）的方式進行開發。提示工程是先前模型不曾見過的，由

doi:10.6342/NTU202303187

於思維鏈[21]技巧的發現，促進了研究者利用思維鏈技巧的提示來解決各領域的問題。

而在應用大型語言模型進行該任務時，會涉及到的因素有以下四個：自然語言問題Q、提示P、大型語言模型L和 SQL 生成結果Y。整個過程可以表示成$Y = f(Q|P, LLM)$，即在有提示P及大型語言模型LLM的情況下評估性能狀態的好壞，自然語言問題此因素會是固定不變的，而會改變的是提示的設計，以及大型語言模型的選擇，以及 SQL 生成結果的正確性。整個過程就是，自然語言問題透過提示的設計，經過大型語言模型的處理後，生成最終的 SQL 查詢，圖示如圖 6。

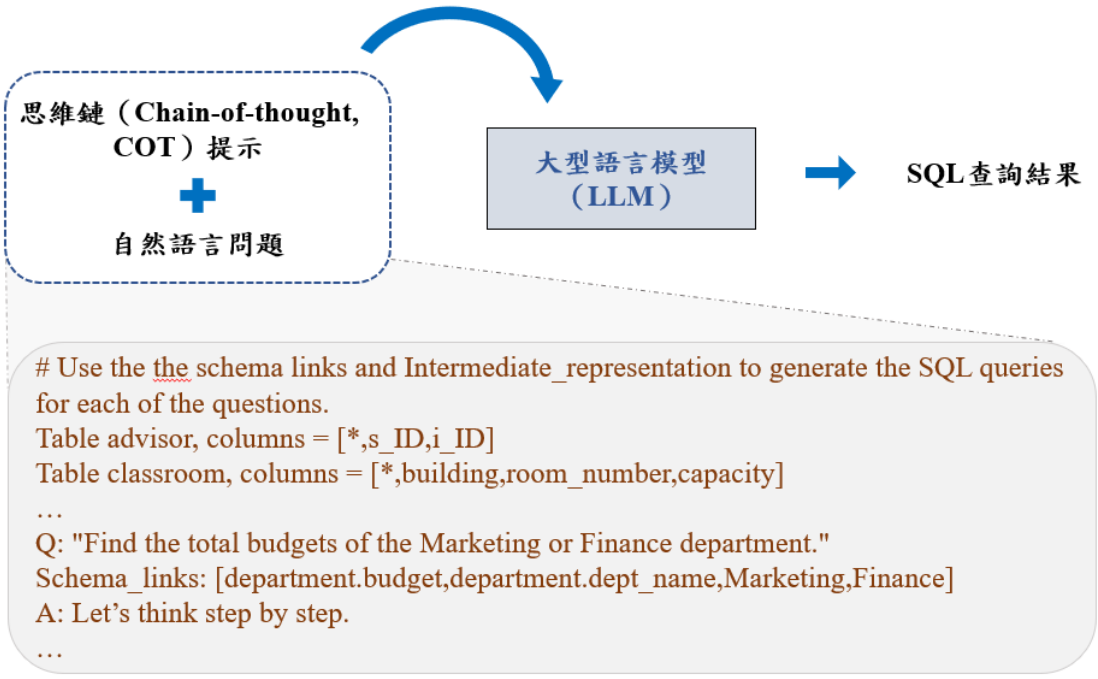

圖 6：運用大型語言模型解決 Text-to-SQL 任務示意圖。

這種結合大型語言模型與提示工程的方式，使得 Text-to-SQL 任務在處理自然語言問題與 SQL 之間的映射時，能夠獲得更好的性能和結果，而在此任務最典型的模型例子就是 DIN-SQL[14]。它在 Spider[24]資料集的執行準確率上成為最先進模型；然而即便如此，仍有進一步提升準確率的空間，在其論文中，作者通過錯誤分析比較了自己的模型和 zero-shot 提示方法的改進程度。本研究將

doi:10.6342/NTU202303187

此模型作為優化的基線（baseline）模型，同時分析其結果，進一步探討可以進行優化的方面。

3.2 最先進模型的錯誤分析

為了深入了解最先進模型的改善潛力，本研究進行了 DIN-SQL[14]的錯誤分析。接下來，在 3.2.1 節中，將比較 DIN-SQL 在 GPT-3.5 及 GPT-4[13]的性能表現，並解釋為何需要在這兩種大型語言模型上測試；在 3.2.2 節中，會針對其錯誤分析結果，說明本研究可從那些方面做優化。

3.2.1 GPT-4 與 GPT-3.5 錯誤分析比較

在 DIN-SQL[14]論文中，作者使用了 GPT-4[13]及 CodeX Davinci 測其性能，並比較了兩者不同規模的模型表現差異；然而，本研究選擇使用網頁版的 GPT-3.5 作為主要模型，GPT-4 只做部分實驗的測試結果。

本研究未使用 API 的原因有兩個：一、OpenAI[2]在提供 API 的使用是有限制的，取得方式是要先對 OpenAI Evals[3]有貢獻，才能優先取得 API 使用權限，若無，則會在等待列表。對於 OpenAI Evals 此部分尚無涉略，故無法迅速取得 API。二、使用 API 計費標準與網頁版不同，根據 DIN-SQL 的 github 議題中作者提及，完整運行 GPT-4 開發集需要花費 600 美元。考量到經濟因素，也暫時不考慮使用 API。故目前只將 ChatGPT 升級為 plus 版本。

即使 ChatGPT Plus 有 GPT-4 模型可用，但每 3 小時內只能執行 25 次，這限制了在短時間內完成所有實驗的可能性。其次，Plus 版本的 GPT-4 token 數僅接收 2048 個，而 GPT-3.5 可以達到 4096 個，大部分實驗（包含 DIN-SQL）的 token 長度超出 GPT-4 的最大限制。因此，本研究僅在 GPT-4 模型上執行在 GPT-3.5 上獲得最佳執行結果，且 token 數不超過 4096 的提示方法的實驗。

[2] https://openai.com/
[3] https://github.com/openai/evals

doi:10.6342/NTU202303187

由於本研究的主要目標是優化基於「DIN-SQL+GPT-4[14]」的最先進模型，故錯誤分析與改進策略仍以 GPT-4 為主。因為本研究只能實驗在 GPT-3.5 的緣故，所以會額外參考 GPT-3.5 的預測結果來進行比較。

分析流程如下：隨機抽樣 100 筆 GPT-4 錯誤預測結果樣本，錯誤分析同 DIN-SQL 論文中 Few-shot 分析的分類方式，以便進行相互比較，但在部分分析結果中我們無法將之用相同的分類方式分類，因此有新增了幾項與原論文不同的分類，詳細如附錄 A。由於 DIN-SQL 在論文中僅對每個分類提供了大致的解釋，但實際在分析時，可能存在一些認知上的差異，因此本研究所得出的結果可能會與原論文提供的數據略有不同，但整體趨勢是相似的，故仍具有參考價值。

GPT-4 錯誤分析結果如表 6 所示。根據分析，大部分的錯誤（37.74%）都是出在模式連結（schema linking）的問題，接下來排序下來錯誤類型包括（Other）、GroupBy、Join、Invalid 和 Nested。在模式連結的部分，最常見的是與實體（entity）有關的錯誤。可以推測錯誤的原因可能是，在生成 SQL 的過程中，模型並沒有看過資料庫內的資料，所以在根據題目描述進行查詢時，偶爾會發生實體不匹配的狀況。舉例來說，如果問題敘述是「How many dog pets are raised by female students?」，對照資料庫表裡面的格式，應該要找 sex 為「F」的實體，但題目只提到「female」，因此模式很有可能直接生成「sex = 'female'」的 SQL 查詢，導致無法找到正確的資訊。

doi:10.6342/NTU202303187

表 6：DIN-SQL[14]演算法中，大型語言模型使用 GPT-4[13]模型時的錯誤分析結果。

分類	子分類	子分類占比	分類占比
模式連結	錯誤的表格	3.70	37.04
	錯誤的欄位	15.74	
	錯誤的實體	17.59	
Join	錯誤的表格	7.41	8.33
	錯誤的方式	0.93	
GroupBy	錯誤的欄位	11.11	12.04
	未偵測到	0.93	
巢狀查詢	集合	4.63	6.48
	錯誤的巢狀查詢	1.85	
其他	Distinct	16.67	28.70
	冗贅的欄位	6.48	
	條件	5.56	
無效	條件	6.48	7.41
	集合	0.93	

同樣地，本研究取出 100 筆錯誤預測樣本進行錯誤分析，結果如表 7 所示。在模式連結方面，GPT-3.5 的錯誤更加嚴重，占了 42.86%。在 join 分類上，錯誤頻率稍微高於 GPT-4（GPT-4 占 8.33%，GPT-3.5 占 9.89%）。然而整體而言，GPT-3.5 發生的問題都與 GPT-4 上的問題相似。因此，將 GPT-3.5 作為實驗的模型是相當合適的。

doi:10.6342/NTU202303187

表 7：DIN-SQL[14] 演算法中，大型語言模型使用 GPT-3.5 模型時的錯誤分析結果。

分類	子分類	子分類占比	分類占比
模式連結	錯誤的表格	3.70	39.81
	錯誤的欄位	16.67	
	錯誤的實體	19.44	
Join	錯誤的表格	5.56	9.26
	錯誤的方式	3.70	
	錯誤的欄位	1.85	
GroupBy	錯誤的欄位	5.56	5.56
	未偵測到	0.00	
巢狀查詢	集合	1.85	5.56
	錯誤的巢狀查詢	3.70	
其他	Distinct	13.89	24.07
	冗贅的欄位	5.56	
	條件	4.63	
無效	條件	4.63	13.89
	集合	0.00	
	Select	4.63	
	表格與欄位	4.63	

本研究更進一步將 DIN-SQL 分解與分類模組結果進行解析，以 Spider[24] 資料集提供之正解為主，來進行 SQL 查詢的分類，呈現如表 8。從數據中可以看出，整體的分類準確率僅 75.63%。在進行每一筆的錯誤分析時，有發現一個現象是，模型容易把原本簡單或只需要用到 join 的問題，都輸出成較複雜的巢狀形式。雖然有時結果是正確的，但如果能生成更直觀、更簡潔的 SQL 查詢，效果將更好。這種現象可以與表 8 中簡單類容易被分類成無巢狀類與有巢狀類相呼應。

doi:10.6342/NTU202303187

表 8：DIN-SQL[14]在 GPT-3.5 實驗中，分解與分類模組結果分類表格。中間數值為其中一次的分類分布狀況，最底下為平均 7 次的分類準確率結果。

正解 / 預測	簡單	無巢狀	有巢狀	
簡單	377	3	2	
無巢狀	123	308	60	
有巢狀	41	19	97	
無	3	1	0	
平均 7 次的分類準確率	**69.30%**	**93.05%**	**61.01%**	**整體準確率 = 75.63%**

3.2.2 本研究的改善策略

根據表 6 及表 7 分別在 GPT-4[13]及 GPT-3.5 上的分析結果，改善策略可以大致分為以下四個方向：

一、模式連結模組的優化：

在 DIN-SQL 模組架構中，第一個模組為模式連結模組。它的準確性對於後續模組的功能展現至關重要。因此，本研究旨在降低模式連結模組的錯誤發生率，以確保後續模組能夠產生最佳的預測結果。

二、分解與分類模組的改進：

在 DIN-SQL 的分解與分類模組中，分類準確率並不高，而這將主要影響到後續使用 join 及巢狀的方式。故本研究計畫調整該模組的提示架構，或是思考其他的分類方式，以提高分類的準確率。

三、SQL 生成模組的改善：

DIN-SQL 生成 SQL 模組是最直接影響 SQL 生成的結果的。因此，本研究欲改善提示的內容，以產生正確的結果。

四、自我修正模組與生成結果修正方式的改進：

從 DIN-SQL 論文的消融（Ablation）研究可以得知，自我修正模組能有效提升 SQL 輸出的品質，然而，仔細觀察結果發現，提示內容仍然與改善後的結果存在差距。此外，還有一些基本的格式錯誤可以進行

doi:10.6342/NTU202303187

微調。故本研究將改善提示內容，同時調整生成結果修正方式，以解決生成的 SQL 查詢不符合語法的問題。

綜合以上四個方向的改善策略，可以期待在 DIN-SQL 模組中提高分類準確率，生成更正確且符合語法的 SQL 查詢，取得更好的預測結果。這些改進將有助於提升 DIN-SQL 的整體性能可靠性，從而增強其在資料查詢方面的實用性和應用價值。

3.3 研究方法

本研究針對優化最先進模型——DIN-SQL[14]的性能嘗試了多種不同的提示策略，根據 3.2.2 節提及的四個方向可對應到以下小節：

一、模式連結模組的優化，改善的方法見 3.3.1 節和 3.3.2 節。

二、分解與分類模組的改進，改善的方法見 3.3.3 節。

三、自我修正模組與生成結果修正方式的改進，改善的方法見 3.3.4 節。

在改善 SQL 生成模組的策略中，由於本研究尚未找到合適的優化方式，故此改善策略暫不討論。接下來將詳述每種策略改善的方法。

3.3.1 Rev1.1——ReDINSQL：從 RESDSQL 交叉編碼器取得模式連結結果

本研究考慮 RESDSQL[9]的交叉編碼器作為改進的選擇，原因如下：

一、RESDSQL 是在 Spider 資料集[24]執行準確率排行榜中位居第三名，同時也是 seq2seq 模型中的第一名。

二、交叉編碼器生成的結果使用 ROC 曲線作為評估標準，能夠確保模式連結結果的品質。

三、交叉編碼器生成的結果是文字序列，而非向量，這使得它可以直接用作後續提示的一部分，更加方便。

doi:10.6342/NTU202303187

基於上述原因，本研究即用其產生模式連結結果，並遵照同個格式作輸入。

RESDSQL 交叉編碼器與 DIN-SQL 模組結合如圖 7，這裡稱之為 ReDINSQL。原先 DIN-SQL 是四個模組，這裡刪除了分解與分類模組，改由三個模組組成：模式連結模組、生成 SQL 模組和自我修正模組。

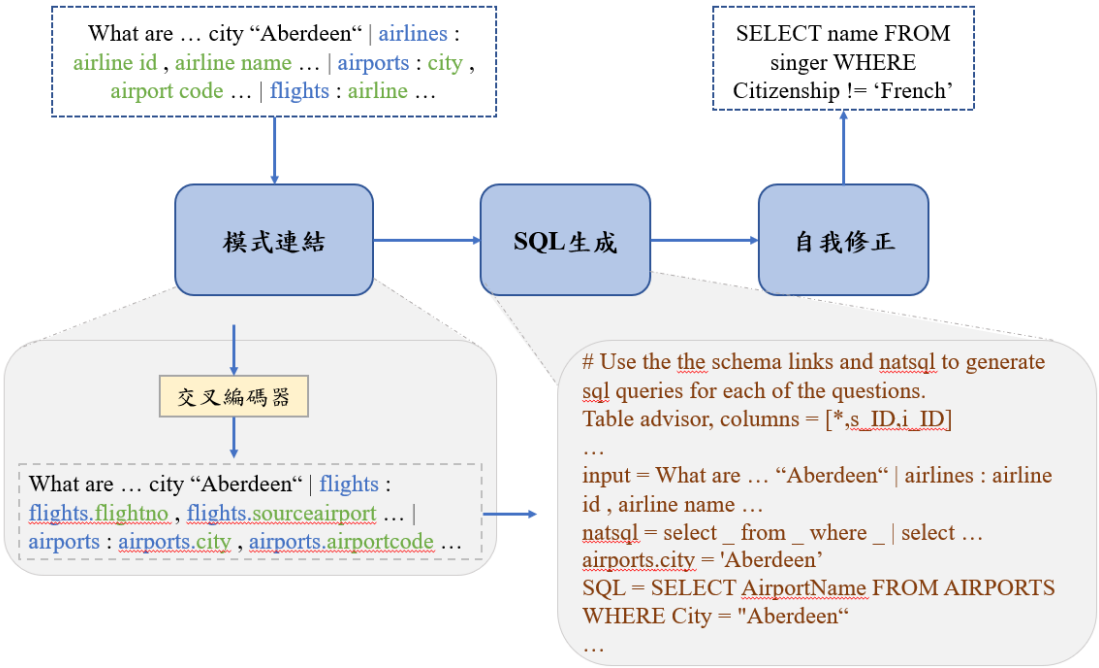

圖 7：ReDINSQL 架構圖。模式連結改為 RESDSQL[9]交叉編碼器。SQL 生成模組想法同 DIN-SQL[14]，但敘述方式以及提供的資訊稍微不同。自我修正模組則同 DIN-SQL。

原先 DIN-SQL 的模式連結模組，替換成 RESDSQL 架構中的交叉編碼器部分。問題Q與資料庫模式S，經過交叉編碼器後，得到排序及過濾好的資料庫模式序列X，表示如下：

$$X = Q \mid t_1 : c_1^1, c_2^1, \ldots \mid t_2 : c_1^2, c_2^2, \ldots$$

此一序列即為模式連結結果，將會在下個模組中使用。

在 SQL 生成模組中，使用了與 DIN-SQL 相似的想法，也就是提供類似的資訊，但敘述方式與使用的樣本與 DIN-SQL 不同。方法是，從訓練集抽取 9 個樣本作為提示，這些樣本平均是從四種不同難度的題目中隨機抽取的。每個樣本提供模式連結、中間表示以及正確解答。模式連結即是前個模組生成的結果。中間表示I使用兩種形式的 NatSQL[4]：一種是 SQL 骨架，保留關鍵字並將

doi:10.6342/NTU202303187

其餘部分替換為插槽（slot），目的是讓問題到生成 SQL 查詢時，能更輕易的找到對應的關鍵字；另一個則是將經過規範化後的值，填入插槽，以降低模型學習填值的難度。如此便能強化資料庫結構，引導模型生成更符合問題需求的查詢結果。最後，則是提供正確的 SQL 查詢。

自我修正模組則是 DIN-SQL 原始架構的第四個模組，由於它確實能提升性能，因此在此改善方法中，本研究仍然保留並使用它。

3.3.2 Rev1.2：新增實體修正模組

根據表 6 和表 7 錯誤分析結果，可以觀察到在實體（entity）在預測方面容易發生錯誤，原因是因為 DIN-SQL 在整個模組架構中並無參考資料庫單元值（cell value）的內容，若僅依據問題描述的資訊直接作為查詢條件，可能會導致格式不一致的問題，進而找不到預期的結果。因此，本研究開發了一個實體修正模組。

此模組的方法步驟如下：

一、找出 SQL 查詢中提及的實體部分，並向前回找該實體所在的欄位和表名。

二、從資料庫中，根據所獲得的表和欄位名，找出 10 組不重複的單元值。

三、在提示中請求模型確認，若有相似的值，則將其替換成資料庫中出現的字串；若否，則根據單元值規則（例如：以大寫字母開頭），修正目前的實體。

演算法表示，如表 9。

表 9：實體修正模組演算法。

1	if (SQL 查詢中存在實體):
2	表與欄位集合←實體所在的表與欄位名集合
3	利用表與欄位集合，下 SQL 查詢，得出不重複的 10 組單元值
4	if (在提示中請求模型確認有相似的值):
5	請求「替換字串」。

doi:10.6342/NTU202303187

6	else:
7	請求「根據單元值規則（例如：以大寫字母開頭），修正目前的
8	實體」
9	else:
10	continue

3.3.3 Rev2：調整分解與分類模組提示架構

根據表 8 的數據，可以看出 DIN-SQL 在分類處理方面還有進步的空間。因此，本研究針對此模組進行了兩種改進方式：第一種（Rev2.1）是用相似的提示架構，但改變敘述思維鏈[21]過程的方式；第二種（Rev2.2）是改變分類方式，原先分為三類，改成分成兩類。接下來將詳細說明本研究的調整方法。

對於 DIN-SQL 的分類準確率不高，我們推測原因如下：

一、提示內容存在冗餘詞語，導致內容過長，增添了模型記憶上下文的負擔，也可能受到那些無直接幫助的詞彙影響。

二、敘述方式不夠「程式化」，對話方式過於口語化，與人際間的交談相似，缺乏程式的專業表達方式。

因此，本研究對 DIN-SQL 中的提示語句進行了調整，以增強模型的學習能力。調整方法如下：

一、縮短描述問題時的用詞。

二、敘述方式越接近程式語言表達方式，應能使生成式模型產生出更接近程式語言形式的內容。因此使用程式邏輯的描述方式改寫句子，例如使用 if else 等。

三、提示工程有項技巧，是多使用區隔符號如：""、[]和<tag></tag>，有助於加深模型的記憶。因此，本研究嘗試在需要強調的部分添加區隔符號，例如：在問題敘述中提及的與表格相關的詞彙，增加中括號（[]），以讓模型知道這些詞彙與後續內容相互關聯。

綜合應用上述三種調整方法，以找出最適合作為提示的方式。調整方法一後面

doi:10.6342/NTU202303187

將簡稱為「Rev2.1 a.」，調整方法二加三簡稱為「Rev2.1 b+c.」。

另外一種思考方式為改變分類方式。目前 DIN-SQL 將問題分為三類：簡單類、無巢狀類和巢狀類，然而，實際上每個問題在撰寫 SQL 時有很多種方法，並不只有一種 SQL 查詢能解決問題。因此，我們認為可以調整問題的分類方式，將問題僅分為簡單類及複雜類。簡單類涉及單表格查詢的問題，其餘則屬於複雜類。

3.3.4 Rev3：重寫自我修正模組

根據 DIN-SQL 實驗結果顯示，自我修正模組在解決部分文法錯誤和檢查生成結果的可調整性方面已經有一定效果；然而，實際去觀察每筆修正後的 SQL 查詢，可以發現效果不夠顯著。在生成 group by 關鍵字時前面應伴隨著聚合函式（aggregation function）（例如 count、sum 和 max 等），但在 DIN-SQL 自我修正模組並未處理這一點。此外，DIN-SQL 程式碼在處理自我修正後的結果時，並未進行充分的檢查；具體來說，模型可能沒有按照提示要求的內容進行生成，但作者仍將模型返回的結果視為答案。因此，本研究將針對以上兩點做細部調整，改進措施如下：

一、改善自我修正的提示，特別在處理 Group by 時加以考慮。

二、若模型回傳的結果不如預期，將再次要求返回結果，以獲得更合適的解答。

doi:10.6342/NTU202303187

第四章　研究結果與討論

4.1　實驗規劃

本研究透過公開的訓練和開發資料集進行實驗與評估。在提示的設計上會進行消融研究（ablation study）。在 RESDSQL[9]交叉編碼器的訓練上，目前是使用 RESDSQL github[4]上提供的模型 checkpoints。

4.2　相關設置

本研究實驗設置可分成大型語言模型和 RESDSQL[9]交叉編譯器兩部分做說明。

大型語言模型中，本研究選擇使用 GPT-3.5 作為實驗主要模型，原因在 3.2.1 節已經提及，在此章節就不重複說明。

RESDSQL 交叉編碼器參數同其論文提及的設置，設定如表 10 所示。

表 10：RESDSQL 交叉編碼器的參數配置表。

參數	配置
欄位增強層中的頭數	8
批次大小	32
學習率	1e-5
優化器	AdamW
欄位增強層 top-k_1	4
欄位增強層 top-k_2	5
損失函數	Focal loss
損失函數 Focusing 參數	2
損失函數加權因子 α	0.75

4.2　資料集

本研究使用了 Spider[24]資料庫，詳細說明可參考 2.1.1 節。在此資料集上，將利用訓練集的部分數據來設計提示，以實現 few-shot 的示例，再利用開

[4] https://github.com/RUCKBReasoning/RESDSQL

doi:10.6342/NTU202303187

發集評估方法的性能表現。

需要注意的是，由於測試集的不公開性質，我們無法直接在測試集上進行評估。若想要在測試集上進行測試，必須將相關程式碼提交給資料集的作者進行處理，並且有兩個月只能繳交一次的限制。因此，本研究目前尚未獲得測試集上的測試性能結果。

4.3 評估指標

Spider 資料集[24]主要有兩種評估指標，分別是精確匹配率（Exact Matching, EM）和執行準確率（Execution Accuracy, EX）。精確匹配率是比較預測及解答的 SQL 語句關鍵字（例如 SELECT、FROM 或 WHERE 等）與銜接的欄位是否一致，只有兩者完全匹配時，才視為正確。然而，此評估方式不會比較條件中的值的結果，例如：SELECT count(*) FROM pets WHERE weight > 10，並不會檢查 10 這個值是否預測正確。執行準確率則是比較兩者執行 SQL 語句後，資料庫回傳的結果是否相符。

另外還有一個可用來評估準確性的指標，稱作測試套件準確率（test suite accuracy, TS）[29]。方法是通過從隨機生成的數據庫中選取一個具有高代碼覆蓋率（code coverage）的小型測試套件，來近似語意準確率，能更有效的檢查語意準確率的上界。

在實際應用情境中，更重要的是檢查 SQL 查詢輸出的結果是否與預期結果相符，而不僅僅是強調語法是否一致。因此，本研究將專注於提升執行準確率（EX）。

執行準確率中每個範例分數計算方式如下：

$$score(\hat{V}, V) = \begin{cases} 1, & \hat{V} = V \\ 0, & \hat{V} \neq V \end{cases}$$

$\hat{V}$為預測的 SQL 查詢結果，V為解答的 SQL 查詢結果。評估計算方式如下：

$$EX = \frac{\sum_{n=1}^{N} score(\hat{Y}_n, Y_n)}{N}$$

doi:10.6342/NTU202303187

N為總組數。

在 RESDSQL 生成交叉編碼器的部分，使用的是 ROC 曲線作為評估模型的指標。表格跟欄位各自有自己的 AUC 分數，為了找到最佳的模型。會將表格 AUC 和欄位 AUC 的分數相加後，選擇總分最高的模型作為最佳模型。

4.4 實驗結果

本節將會呈現該研究的實驗結果，而因使用 ChatGPT 仍有每天的流量限制，故每次實驗執行次數並不多，以下結果皆會列出每次實驗的執行次數。

4.4.1 Rev1：模式連結優化方式比較

模式連結模組優化的方式有兩種，分別對應研究方法 3.3.1 節與 3.3.2 節。比較結果如表 11 所示。根據結果，在開發集中，ReDINSQL 的結果性能最好，提升了 3.82%。推測原因是因為，RESDSQL 交叉編碼器在生成模式連結的表現，優於 DIN-SQL 模式連結模組結果，這使得後續的生成 SQL 模組能夠更精準地輸出結果。

另一個發現是關於 Rev1.2 的實體修正模組。在 DIN-SQL 添加實體修正模組後，性能略為提升 1%；然而，在 ReDINSQL 的結果中，加上實體修正模組後，性能反而下降了 3.24%。證明說 Rev1.2 僅對 DIN-SQL 有效，而對於 Rev1.2 是無效的。推測無效原因是，在模式連結模組時已參考了資料庫單元值內容，在預測時已無太多的實體錯誤，故多加一層實體修正模組，並不會有太大的改進。

表 11：改善一的結果比較表。

提示架構	執行準確率（EX）（%）	執行次數
DIN-SQL	70.28	7
ReDINSQL	74.10	6
DIN-SQL + Rev1.2	71.28	4
ReDINSQL + Rev1.2	70.86	9

doi:10.6342/NTU202303187

4.4.2 Rev2：分解與分類模組改進方式比較

分解與分類模組的改進方法在 3.3.3 節提及，本研究嘗試了兩種調整方式。

對於改變敘述思維鏈[21]方式，我們試了兩種改進策略，第一種是純粹縮短敘述（Rev2.1 a.），第二種是讓敘述更加程式化並加上標註符號（Rev2.1 b+c.）。分類結果如表 12 和表 13 所示。

表 12：Rev2.1 a.的分類結果。

預測 \ 正解	簡單	無巢狀	有巢狀	
簡單	355	9	1	
無巢狀	128	271	34	
有巢狀	58	48	124	
無	3	3	0	
分類準確率	**65.26%**	**81.87%**	**77.99%**	**整體正確率 = 72.53%**

表 13：Rev2.1 b+c.的分類表結果。

預測 \ 正解	簡單	無巢狀	有巢狀	
簡單	426	74	18	
無巢狀	50	188	31	
有巢狀	60	63	110	
無	8	6	0	
分類準確率	**78.31%**	**56.80%**	**69.18%**	**整體正確率 = 70.02%**

然而，與表 8 中 DIN-SQL 的結果相比，這兩種改進方式皆未提升分類能力。根據表 12 縮短文字敘述的分類結果顯示，僅有巢狀類準確率上升，其餘分類準確率皆下降。依照表 13 將提示改成程式化並加上標註符號的結果，在簡單類與巢狀類的預測上，準確率皆有提升；然而，在無巢狀類卻出現嚴重的錯誤。

在改變分類方式（Rev2.2）中，分類結果如表 14。整體的準確率略贏

doi:10.6342/NTU202303187

DIN-SQL 約 1.83%，在簡單類的分類上準確率就高了 6.3%。

表 14：Rev2.2 的分類結果。

預測 \ 正解	簡單	複雜	
簡單	428	113	
複雜	116	377	
分類準確率	**76.68%**	**76.94%**	**整體正確率 = 77.85%**

上述 Rev2 的三種方式，儘管分類準確率尚未達到一定的水準，但我們仍試著執行整個完整的實驗，結果如表 15 所示，可以觀察在 Rev2.1 b_c.的提示下，性能提升 0.42%，Rev2.1 a.和 Rev2.2 則提升 1.96%；然而，這些結果並不符合我們的預期，原因是在分類準確率下降的情況下，理論上最後的執行準確率也應該是下降的。

表 15：Rev2 的結果比較表。

提示架構	執行準確率（EX）（%）	執行次數
DIN-SQL	70.28	7
DIN-SQL + Rev2.1 a.	72.24	1
DIN-SQL + Rev2.1 b+c.	70.70	1
DIN-SQL + Rev2.2	72.24	1

為了進一步探討這個現象，我們移除了最後一層的自我修正模組，結果如表 16 所示。通過這次的實驗，可以更清楚地看出分類準確率的高低確實是會影響 SQL 生成準確率的。具體來說，在分類準確率較低的 Rev2.1 a.與 Rev2.1 b+c.中，分類正確且 SQL 生成的正確率皆低於 DIN-SQL 基線模型，分別降低了 16.92%和 12.86%，執行準確率結果分別是 44.39%和 50.97%；而分類準確率較高的 Rev2.2，正確率則優於基線模型，提升 0.58%，執行準確率結果為 64.89%。

doi:10.6342/NTU202303187

表 16：DIN-SQL 與 Rev2 SQL 生成模組後生成的結果比較表。

	DIN-SQL		DIN-SQL + Rev2.1 a.		DIN-SQL + Rev2.1 b+c.		DIN-SQL + Rev2.2	
分類 / 預測結果	正確	錯誤	正確	錯誤	正確	錯誤	正確	錯誤
正確個數	559	142	384	75	426	101	565	106
錯誤個數	223	110	366	209	298	209	240	123
分類正確且 SQL 的生成準確率	54.06%		37.14%		41.20%		54.64%	
執行準確率	67.12%		44.39%		50.97%		64.89%	

4.4.3 Rev3：自我修正模組與生成結果修正方式改進比較

在 3.3.4 節中介紹了對自我修正模組的改進方式，其結果如表 17 所示。從數據可以觀察到，將 DIN-SQL 的自我修正模組改成 Rev3 後， DIN-SQL 的性能些微提升了 0.59%，在 RESDSQL 交叉編碼器的模組上，提升了 0.72%。然而，這些性能提升幅度並不大，但確實是個可參考的模組修正方式。因此，在自我修正模組的改善方面可進一步延伸做討論。

表 17：Rev3 的結果比較表。

提示架構	執行準確率（EX）（%）	執行次數
DIN-SQL	70.28	7
DIN-SQL+ Rev3	70.87	4
ReDINSQL	74.10	6
ReDINSQL + Rev3	74.82	8

4.4.4 最佳性能提示架構與基線提示架構錯誤分析比較

根據前面的實驗結果觀察，ReDINSQL 在性能上取得了最顯著的提升，因此，我們將其與 DIN-SQL 基線模型錯誤結果進行比較，以評估性能的提升程度。比較如圖 8 所示。根據結果可以看出，在模式連結方面改善效果顯著，錯誤率降低了 8.79%。同時也觀察到，在無效的 SQL 查詢處理方面有所改善，猜測原因可能如下：

doi:10.6342/NTU202303187

一、在 DIN-SQL 的簡單類中並無提供中間表示，故可能在簡單類時產生了無效的指令；而 ReDINSQL 則全部皆有提供。

二、DIN-SQL 僅提供了 NatSQL 已將值填充完框架中的插槽的中間表示，而 ReDINSQL 則是提供了填充前與填充後的字串，使得模型可以更好地學習到值填充的過程，從而改善了無效查詢的處理。

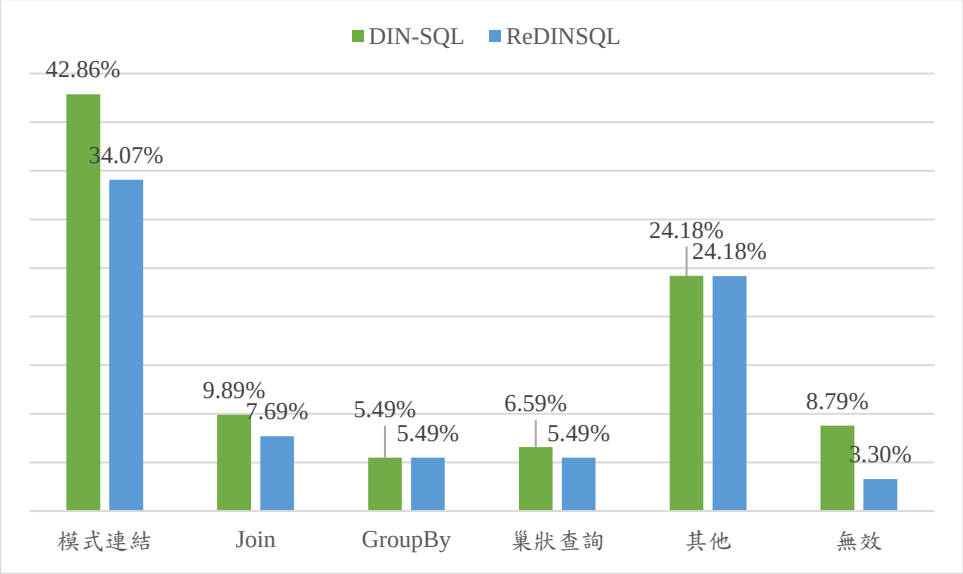

圖 8：ReDINSQL 與 DIN-SQL 模型錯誤分析比較直條圖。

4.4.5 最佳性能提示架構在 GPT-4 上的結果討論

由於 GPT-4 有使用流量限制的關係，本研究僅將在 GPT3.5 中選擇了性能表現最優異的提示架構在 GPT-4 上實驗。因此，我們選擇了 ReDINSQL 的結果。除此之外，我們還對最後一個自我修正模組進行了消融研究，包括保留自我修正模組、移除自我修正模組和替換為本研究所設計的自我修正模組。表 18 展示了相關的數據結果。

doi:10.6342/NTU202303187

表 18：GPT-4 模型上，本研究最佳性能模組與 DIN-SQL 模組的結果比較表。

提示架構	執行準確率（EX）
DIN-SQL	81.72
ReDINSQL	79.79
ReDINSQL + Rev3	79.49
ReDINSQL（w/t 自我修正模組）	80.86

從數據中觀察到兩個結果：首先，整體而言，本研究的最佳性能提示架構與 DIN-SQL 的性能相當，然而，在 GPT-3.5 模型上，我們的最佳性能提示架構明顯優於基線模型。這顯示出本研究的最佳性能提示架構在 GPT-4 上無法展現其強大性能。其次，我們觀察到，移除自我修正模組的提示架構性能比保留該模組性能更好；也就是說，在 GPT-4 模型上，自我修正模組對於 RESDSQL 交叉編碼器的結果並沒有發揮作用。根據錯誤分析結果，如表 19，可以觀察到在其他分類的條件子分類中，錯誤的比例是最高的，自我修正模組會額外添加一些多餘的條件限制，使得原本正確的結果變成錯誤的。另外，在模式連結中的錯誤的欄位也佔了很大比例，也注意到此模組會錯改實體名稱，導致原先已經過單元值審查的實體被改為題目所寫的實體。

表 19：ReDINSQL 保留自我修正模組的錯誤分析結果。

分類	子分類	數量	子分類占比	分類占比
模式連結	錯誤的欄位	9	20.00	24.44
	錯誤的實體	2	4.44	
Join	錯誤的方式	1	2.22	2.22
GroupBy	錯誤的欄位	7	15.56	15.56
其他	Distinct	2	4.44	44.44
	冗贅的欄位	2	4.44	
	條件	12	26.67	
	彙總函數	4	8.89	
無效	格式	6	13.33	13.33

doi:10.6342/NTU202303187

4.5 問題討論

4.5.1 使用提示於線上的大型語言模型產生回應的穩定性討論

在整個實驗過程中，發現了兩個發生在線上的 GPT-3.5 和 GPT-4 模型（也就是 ChatGPT）中的現象：

一、在每次提示輸入後，系統回覆有時不太穩定。曾經我們在 GPT-3.5 模型下進行 DIN-SQL 實驗時，執行準確率（EX）僅 63.64%（相較於其他實驗平均準確率為 70.92%）；另一次經驗是，在 GPT-4 模型下運行本研究最佳性能之提示架構時，執行準確率（EX）僅 69.73%（但在重新運行後提高至 79.79%）。

二、為了驗證分解任務的作法是否確實優於 zero-shot 方法，本研究運行了多次的 zero-shot[12]實驗。過程中觀察到一個現象，即根據作者提供的結果與我們後續執行的結果進行比較，性能存在一些差異。實驗結果如表 20，可以發現本研究執行結果性能，比原始結果提升了約 1.8%。

由第一點的現象，可以得知，對於如何穩定系統輸出結果，以及如何有效地使用簡單的提示方式來解決任務，需要進一步討論。第二點的現象，可以表明，ChatGPT 模型一直有在持續悄悄地更新，並且有著越來越好的發展趨勢。

表 20：zero-shot[12]作者提供的結果與本研究執行結果的比較表。

提示架構	執行準確率（EX）（%）	執行準確率平均（%）
zero-shot（作者提供的結果）	70.21	70.21
zero-shot（本研究重新執行的結果）	71.95	72.03
	71.95	
	72.74	
	71.47	

doi:10.6342/NTU202303187

4.5.2 DIN-SQL 與 zero-shot 性能討論

根據 DIN-SQL[14]論文記載，該提示架構在 GPT-4 或 CodeX Davinci 模型的環境下表現優於運行在 ChatGPT 上的 zero-shot [12]結果。然而，我們卻無法重現論文中的數據。例如，理論上，如果 DIN-SQL 運行在 GPT-3.5 上，其性能應該介於 GPT-4 和 CodeX Davinci 之間，並應該超過在 ChatGPT 上的 zero-shot 結果。然而，實際運行結果卻顯示，zero-shot 性能卻能與之並駕齊驅，在本研究執行的 zero-shot，性能甚至稍微優於 DIN-SQL。具體數據如表 21 所示。我們認為是論文中的數據存在錯誤。為了確認這一點，我們已聯繫了論文的作者，詢問了他們使用的評估指標，並他們表示將在確認後向大家提供更多資訊（詳見附錄 B）。

若 DIN-SQL 的分解方法在性能上與 zero-shot 相似，那麼在 Text-to-SQL 上使用分解方法是否是確實有效的，這件事值得做進一步的討論。

表 21：DIN-SQL 與 zero-shot 在 GPT-3.5 模型的性能比較表。

提示架構	執行準確率（EX）
DIN-SQL	70.28
zero-shot （作者提供的結果）	70.21
zero-shot （本研究重新執行的結果）	72.03

doi:10.6342/NTU202303187

第五章　　結論與未來展望

5.1　結論

在 text-to-SQL 任務中，應用提示於大型語言模型已成為近期的趨勢，尤其是思維鏈[21]想法釋出後，更是加速了研究人員在利用思維鏈解決此任務的速度。而目前，最具代表性的最先進模型即是 DIN-SQL[14]，因此，我們以 DIN-SQL 做為基線模型，與後續實驗方法做比較。

本研究在 3.2 節，先遵從 DIN-SQL 論文中提及的錯誤分析依據，對運行在 GPT-3.5 的 DIN-SQL 進行了錯誤分析，並在 3.2.2 小節中，提出可改善的方向。因此，在 3.3 節，同樣採用思維鏈[21]策略，將 text-to-SQL 任務分解成多個子任務，並嘗試了三種方法來改善每個子任務。觀察 4.4 實驗結果，我們可以得出以下結論：

一、第一種改善方式——模式連結優化，ReDINSQL 可以提升約 3.82%的性能。

二、第二種改善方式——分解與分類模組改進，是在縮短描述以及將分類改為兩種，在兩種方法下，可提升 1.96%的性能。

三、第三種改善方式——自我修正模組，在單獨將其添加到 DIN-SQL 上時，以及與改善方法一之一結合使用時，性能可以提升 0.59%及 0.72%。

這些觀察結果提供了本研究在不同改善方法下的性能提升情況，並為我們提供了有價值的洞察。

然而，在 GPT-4 上，最佳提示架構（ReDINSQL）性能卻僅能與基線模型（DIN-SQL）匹敵，無法超越它，性能約 79.79%。另外，觀察到，若使用自我修正模組，反而降低了性能表現。

在 4.5.1 節中，使用提示於線上的大型語言模型也遇到了一些問題值得探討，可以得知說，因為系統更新造成變動，可能導致輸出結果品質不固定，但

doi:10.6342/NTU202303187

從整體數據上來看的話，模型的性能是有緩緩提升的。

5.2 未來展望

本研究在 GPT-3.5 模型上實驗結果顯示，相較於基線模型，性能確實有提升；然而在 GPT-4 模型下，性能沒有取得更大的突破。因此我們希望能夠在 GPT-4 模型上進行更多的嘗試。但是，目前仍受到每三小時只能發送 25 次請求的限制，本研究計畫取得 GPT-4 模型的 API，以取得更多的資源，這將有助於我們進一步探索在不同提示下性能的提升程度。此外，計劃將本研究的模型提交到 Spider 排行榜，以測試其在測試集上的性能。這將提供一個客觀評估我們模型在實際應用中的表現。

doi:10.6342/NTU202303187

參考文獻

1. Brown, T.B., et al., *Language models are few-shot learners*, in *Proceedings of the 34th International Conference on Neural Information Processing Systems*. 2020, Curran Associates Inc.: Vancouver, BC, Canada. p. Article 159.
2. Cao, R., et al. *LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations*. 2021. Online: Association for Computational Linguistics.
3. Chen, Z., et al. *ShadowGNN: Graph Projection Neural Network for Text-to-SQL Parser*. 2021. Online: Association for Computational Linguistics.
4. Gan, Y., et al. *Natural SQL: Making SQL Easier to Infer from Natural Language Specifications*. 2021. Punta Cana, Dominican Republic: Association for Computational Linguistics.
5. Guo, J., et al. *Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation*. 2019. Florence, Italy: Association for Computational Linguistics.
6. Kojima, T., et al., *Large Language Models are Zero-Shot Reasoners.* ArXiv, 2022. **abs/2205.11916**.
7. Lee, J.-O. and D.-K. Baik. *SemQL: a semantic query language for multidatabase systems*. in *International Conference on Information and Knowledge Management*. 1999.
8. Lei, W., et al. *Re-examining the Role of Schema Linking in Text-to-SQL*. 2020. Online: Association for Computational Linguistics.
9. Li, H., et al., *Decoupling the Skeleton Parsing and Schema Linking for Text-to-SQL.* arXiv preprint arXiv:2302.05965, 2023.
10. Li, J., et al., *Graphix-T5: Mixing Pre-Trained Transformers with Graph-Aware Layers for Text-to-SQL Parsing.* arXiv preprint arXiv:2301.07507, 2023.
11. Lin, X.V., R. Socher, and C. Xiong. *Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing*. 2020. Online: Association for Computational Linguistics.
12. Liu, A., et al., *A comprehensive evaluation of ChatGPT's zero-shot Text-to-SQL capability.* arXiv preprint arXiv:2303.13547, 2023.
13. OpenAI, *GPT-4 Technical Report.* ArXiv, 2023. **abs/2303.08774**.
14. Pourreza, M. and D. Rafiei, *DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction.* arXiv preprint arXiv:2304.11015, 2023.
15. Qi, J., et al. *RASAT: Integrating Relational Structures into Pretrained Seq2Seq Model for Text-to-SQL*. 2022. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

doi:10.6342/NTU202303187

16. Raffel, C., et al., *Exploring the limits of transfer learning with a unified text-to-text transformer.* J. Mach. Learn. Res., 2020. **21**(1): p. Article 140.
17. Rajkumar, N., R. Li, and D. Bahdanau, *Evaluating the text-to-sql capabilities of large language models.* arXiv preprint arXiv:2204.00498, 2022.
18. Scholak, T., N. Schucher, and D. Bahdanau. *PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models*. 2021. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.
19. Tang, L.R. and R.J. Mooney, *Using multiple clause constructors in inductive logic programming for semantic parsing*, in *Proceedings of the 12th European Conference on Machine Learning*. 2001, Springer-Verlag: Freiburg, Germany. p. 466–477.
20. Wang, B., et al. *RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers*. 2020. Online: Association for Computational Linguistics.
21. Wei, J., et al., *Chain of Thought Prompting Elicits Reasoning in Large Language Models.* ArXiv, 2022. **abs/2201.11903**.
22. Xu, P., et al. *Optimizing Deeper Transformers on Small Datasets*. 2021. Online: Association for Computational Linguistics.
23. Yu, T., et al. *CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases*. 2019. Hong Kong, China: Association for Computational Linguistics.
24. Yu, T., et al. *Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task*. 2018. Brussels, Belgium: Association for Computational Linguistics.
25. Yu, T., et al. *SParC: Cross-Domain Semantic Parsing in Context*. 2019. Florence, Italy: Association for Computational Linguistics.
26. Zeng, L., S.H.K. Parthasarathi, and D. Hakkani-Tur. *N-Best Hypotheses Reranking for Text-to-SQL Systems*. in *2022 IEEE Spoken Language Technology Workshop (SLT)*. 2023.
27. Zhang, Z., et al., *Automatic chain of thought prompting in large language models.* arXiv preprint arXiv:2210.03493, 2022.
28. Zhao, Y., et al., *Importance of synthesizing high-quality data for text-to-SQL parsing.* arXiv preprint arXiv:2212.08785, 2022.
29. Zhong, R., T. Yu, and D. Klein. *Semantic Evaluation for Text-to-SQL with Distilled Test Suites*. 2020. Online: Association for Computational Linguistics.
30. Zhong, V., C. Xiong, and R. Socher, *Seq2sql: Generating structured queries from natural language using reinforcement learning.* arXiv preprint

doi:10.6342/NTU202303187

arXiv:1709.00103, 2017.

31. Zhuang, L., et al. *A Robustly Optimized BERT Pre-training Approach with Post-training*. 2021. Huhhot, China: Chinese Information Processing Society of China.

doi:10.6342/NTU202303187

附錄 A——錯誤分析分類說明

本論文在 3.2.2 節與 4.4.5 節的錯誤分析中，分類的準則以基線模型論文中提及的分類方式為主。由於部分錯誤我們在分類時無法為他們歸類在適當的分類中，因此我們多新增了幾項子分類，以下為詳細說明：

一、Join－錯誤的欄位：在 Join 時互為外鍵的欄位未被正確取用。

二、無效－Select：在輸出結果時重複出現 select 關鍵字。

三、無效－表格與欄位：因為表格或欄位的問題，例如：相同的欄位名未指定表名，造成執行時無法辨認，SQL 指令無法執行。

doi:10.6342/NTU202303187

附錄 B——DIN-SQL 作者對於數據異議的解釋

根據 DIN-SQL GitHub 議題[5]中的討論，提到有關論文內的 Exec Acc 分數如何計算的問題，作者在該議題中給出了一些答覆，但仍有部分資訊還待商榷，相關討論的截圖如圖 9 所示。

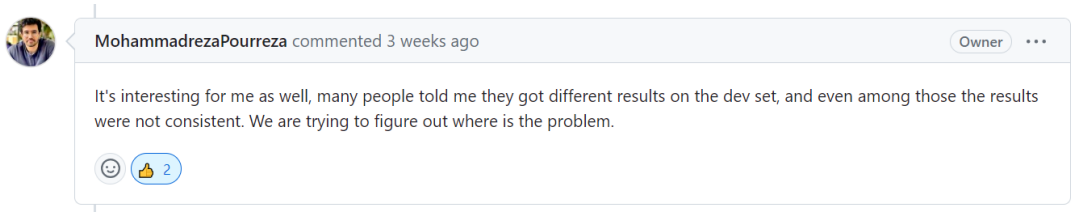

圖 9：作者對於大家對 Exec Acc 分數疑問後的回覆。

[5] https://github.com/MohammadrezaPourreza/Few-shot-NL2SQL-with-prompting/issues/7

doi:10.6342/NTU202303187