Filling Memory Gaps: Enhancing Continual Semantic Parsing via SQL Syntax Variance-Guided LLMs Without Real Data Replay

Ruiheng Liu[1, 2], **Jinyu Zhang**[2], **Yanqi Song**[2], **Yu Zhang**[2*], **Bailong Yang**[1*]

[1]Xi'an Research Institute of High-Tech, Xi'an, China
[2]Harbin Institute of Technology, Harbin, China
{rhliu, jyzhang, yqsong, zhangyu}@ir.hit.edu.cn, xa_403@163.com

Abstract

Continual Semantic Parsing (CSP) aims to train parsers to convert natural language questions into SQL across tasks with limited annotated examples, adapting to dynamically updated databases in real-world scenarios. Previous studies mitigate this challenge by replaying historical data or employing parameter-efficient tuning (PET), but they often violate data privacy or rely on ideal continual learning settings. To address these issues, we propose a new **L**arge Language Model (LLM)-**E**nhanced **C**ontinuous **S**emantic **P**arsing method, named LECSP, which alleviates forgetting while encouraging generalization, without requiring real data replay or ideal settings. Specifically, it first analyzes the commonalities and differences between tasks from the SQL syntax perspective to guide LLMs in reconstructing key memories and improving memory accuracy through calibration. Then, it uses a task-aware dual-teacher distillation framework to promote the accumulation and transfer of knowledge during sequential training. Experimental results on two CSP benchmarks show that our method significantly outperforms existing methods, even those utilizing data replay or ideal settings. Additionally, we achieve generalization performance beyond upper limits, better adapting to unseen tasks.

Code — https://github.com/tom68-ll/LECSP
Extended version — https://arxiv.org/abs/2412.07246

Introduction

Semantic parsing offers a user-friendly interface for non-experts to query data and perform various analyses (Hu et al. 2023; Li et al. 2023b). While most previous studies focus on static data distributions (Li et al. 2023a; Wang et al. 2024a; Guo et al. 2024), the frequent updates in real-world databases have led researchers to shift their focus to continual semantic parsing (Li, Qu, and Haffari 2021; Lialin et al. 2021; Yadav et al. 2023; Chen et al. 2023a,b).

As illustrated in Figure 1(a), a semantic parser must undergo continual training across a series of tasks from different databases, while ensuring good performance on current and historical tasks. This task faces two primary challenges: (1) The scarcity of annotated data for each task can easily result in model overfitting (Qin and Joty 2022; Chen et al. 2023a). (2) Training sequentially leads to catastrophic forgetting (Chen et al. 2023a,b; Wang et al. 2024b; Qiao and Mahdavi 2024), where the model's performance on earlier tasks significantly worsens after learning new ones.

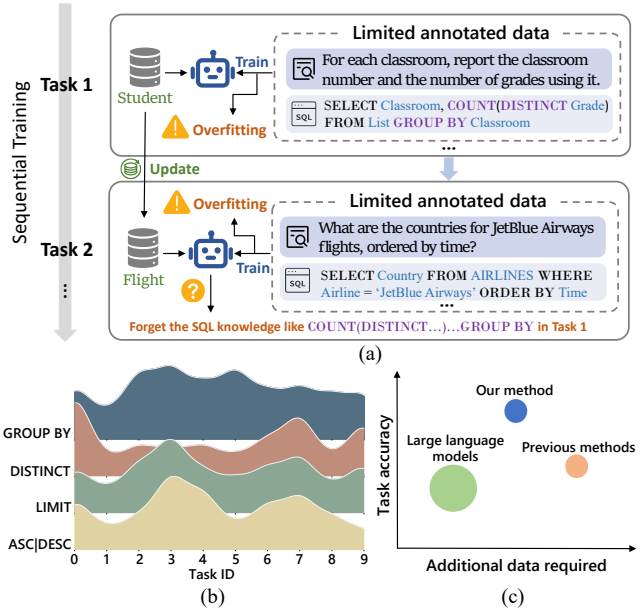

Figure 1: (a) Description of the continual semantic parsing process. (b) The average number of single SQL keywords at cluster centers of different tasks in Spider-stream-semi dataset (Chen et al. 2023a). (c) Comparison between our method and others, where *Additional data required* refers to extra historical data or unsupervised data.

To tackle these challenges, several studies have explored continual learning methods, which can be divided into two types from data and model dimensions: *Rehearsal-based* and *PET-based*. The former devises specific sampling strategies to replay historical task data to prevent forgetting (Li, Qu, and Haffari 2021; Wang et al. 2022; Chen et al. 2023a). The latter freezes the model backbone and uses small-scale parameters, such as prompt (Lester, Al-Rfou, and Constant 2021), to steer representation learning (Chen et al. 2023b;

*Corresponding Authors.

Razdaibiedina et al. 2023; Yadav et al. 2023). Although these methods have made initial progress, they still face limitations. **First**, the performance of *Rehearsal-based* methods relies heavily on the amount of historical data and may not be practical in privacy-sensitive or memory-constrained environments (Jung et al. 2023; Wang et al. 2023b, 2024b). **Second**, *PET-based* methods learn PET modules for each task and store them in a cache pool. When test samples arrive, most methods assume that the corresponding PET module for each sample is known. This ideal setting contradicts real-world scenarios and hinders generalization to unseen samples (Chen et al. 2023b; Razdaibiedina et al. 2023).

Considering the above, we believe the ideal approach should not rely on historical data or ideal continual learning settings. Instead, it should effectively bridge the gaps between tasks by identifying and analyzing the potential commonalities and critical differences in the required knowledge between incoming new tasks and previously learned ones, thereby mitigating forgetting and promoting generalization. As shown in Figure 1(b), limited annotated data can lead to uneven distribution of SQL syntax knowledge among tasks. Our motivation is to leverage these variances to guide LLMs in reconstructing relevant memories to fill these gaps.

In this work, we propose a novel **L**LM-**E**nhanced **C**ontinual **S**emantic **P**arsing method (LECSP). First, it analyzes component biases from the perspective of SQL syntax between current and past tasks, without accessing historical data. Then, these biases are used to guide LLMs in generating relevant pseudo-samples as memory from intra-task and inter-task perspectives, reinforcing commonalities and filling in differences. Additionally, a calibration strategy is introduced to enhance the accuracy and fidelity of memory. To further improve memory utilization efficiency, we design a task-aware dual-teacher distillation framework. This framework not only facilitates the transfer and accumulation of historical task knowledge across the task stream but also enables the migration of external knowledge from LLMs to smaller models. A comparison of our approach with existing methods is illustrated in Figure 1(c). Extensive experiments on the CSP benchmarks Spider-stream-semi (Chen et al. 2023a) and Combined-stream (Chen et al. 2023b) demonstrate that LECSP significantly outperforms other baselines that use data replay or ideal continual learning settings, with gains up to 8.8%, and effectively adapts to more challenging cold-start scenarios. Additionally, we achieve performance beyond the theoretical upper bounds in knowledge forward transfer capability. The main contributions of this work are summarized as follows:

- We propose a novel CSP framework LECSP, including memory reconstruction with LLMs and task-aware dual-teacher distillation learning. These help to mitigate forgetting and promote generalization.
- A memory calibration strategy is introduced, consisting of iterative self-correction and SQL skeleton-based sampling, to further improve memory accuracy and fidelity.
- Extensive experiments on benchmark datasets show that LECSP achieves state-of-the-art (SOTA) performance without using historical data or ideal continual learning setups, and surpasses the upper bounds in knowledge transfer capabilities.

Related Work

Semantic Parsing with LLMs

The purpose of semantic parsing is to simplify data access in relational databases for users. Most previous research (Cai et al. 2022; Li et al. 2023a; Dou et al. 2023) relies on sequence-to-sequence architectures needing fine-tuning on large datasets. However, SQL annotation of natural language is costly and requires domain-specific experts. Recently, with the rise of LLMs like GPT-4 (OpenAI et al. 2024) and PaLM-2 (Anil et al. 2023), some methods have leveraged in-context learning to achieve SOTA performance and reduce labeled data needs (Pourreza and Rafiei 2023; Li et al. 2023b; Mao et al. 2024). However, relying on closed-source LLMs poses database security and privacy risks (Xue et al. 2024), incurs high inference costs, and hinders continual learning due to their black-box nature (Li et al. 2024). Unlike previous work, we leverage LLMs to guide smaller models in dynamically adapting to different tasks in CSP and facilitate capability transfer. Notably, to better meet security and cost requirements in real-world applications, we focus on open-source LLMs that can be deployed offline.

Continual Learning

A significant feature of human intelligence is the ability to learn new tasks while accumulating experience and preventing the forgetting of old knowledge, a concept known as continual learning (d'Autume et al. 2019; Wang et al. 2023a). There are two main challenges: (1) Preventing catastrophic forgetting. (2) Enabling forward transfer of knowledge from past to new tasks. Previous research on continual learning has primarily focused on classification tasks (Han et al. 2020; Wang et al. 2021a; Razdaibiedina et al. 2023). Recently, with advancements in Pre-trained Language Models (PLMs), it has also garnered widespread attention in complex tasks such as generative tasks (Chen et al. 2023a,b; Yadav et al. 2023; Zhao et al. 2024; Liang et al. 2024). SFNET (Chen et al. 2023a) adopts a semi-supervised learning approach, expanding data through self-training on additional unsupervised data. It also designs specific strategies for replaying historical instances to mitigate catastrophic forgetting. C3 (Chen et al. 2023b) freezes the core parameters of the PLM while training small-scale prompts and performing in-context tuning with relevant instances, ideally preventing forgetting issues. In this study, we focus on more realistic and challenging CSP scenarios, without accessing any historical data or relying on ideal task settings.

Preliminaries

Given a natural language question $\mathcal{Q}$ and a database schema $\mathcal{S} = (\mathcal{C}, \mathcal{T})$ consisting of columns $\mathcal{C} = \{c_1^{t_1}, c_2^{t_1}, ..., c_1^{t_2}, c_2^{t_2}, ...\}$ and tables $\mathcal{T} = \{t_i\}_{i=1}^{|\mathcal{T}|}$. The goal of the semantic parsing task is to generate the corresponding SQL query $\mathcal{Y} = \mathcal{F}_\theta(\mathcal{Q}, \mathcal{S})$, where $\mathcal{F}_\theta$ is the parser with

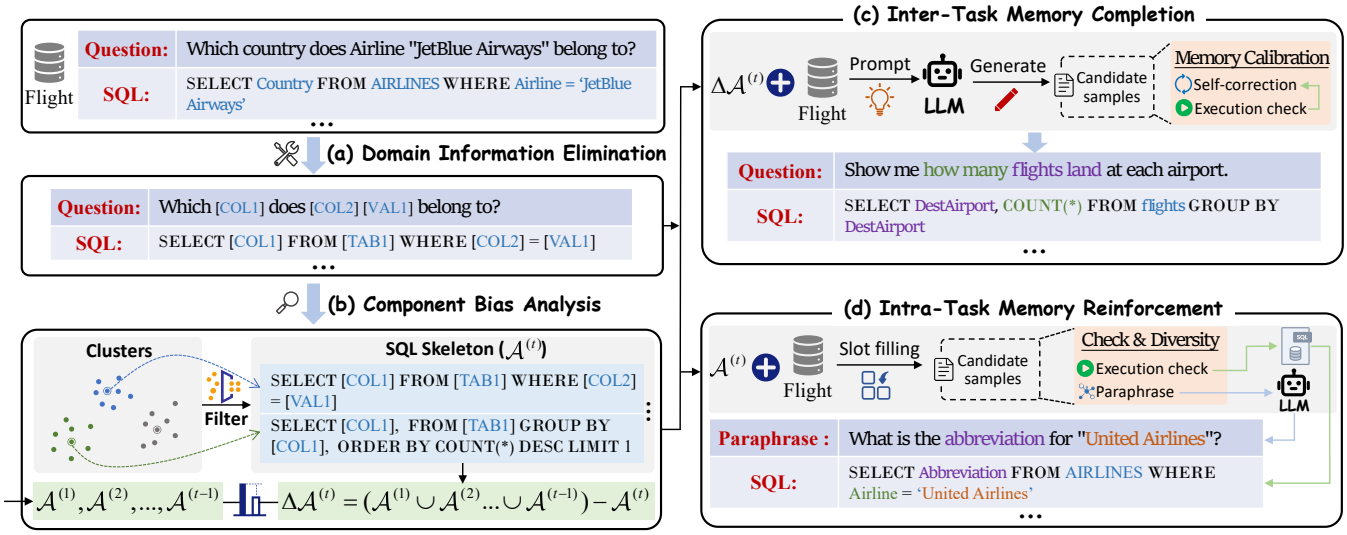

Figure 2: Memory reconstruction on task t via LLMs. (a) Domain-specific information in the questions and SQL pairs is firstly removed by replacing entity-linked results with placeholders like `[COL]` and `[VAL]`. (b) K-means clustering is then applied to derive the SQL skeleton set $\mathcal{A}^{(t)}$ for task t, capturing SQL syntax structure. The component bias $\Delta\mathcal{A}^{(t)}$ is computed by taking the difference with the saved sets from previous tasks $\mathcal{A}^{(1)}, ..., \mathcal{A}^{(t-1)}$. Note that the set $\mathcal{A}$ is only related to SQL syntax and does not involve any historical data. Finally, in (c) and (d), $\Delta\mathcal{A}^{(t)}$ and $\mathcal{A}^{(t)}$ are used to guide inter-task and intra-task memory reconstruction, respectively, filling differences between tasks and reinforcing commonalities.

parameters θ, and $\mathcal{C}$ and $\mathcal{T}$ are respectively the sets of column names and table names, with $c_i^{t_j}$ indicating the i-th column in table t_j. Following the problem definition of previous works (Li, Qu, and Haffari 2021; Lialin et al. 2021; Chen et al. 2023a,b), we assume that the semantic parser in CSP is not confined to a single specific train and test set. Instead, it is required to handle a series of semantic parsing tasks $\{\mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^M\}$, referred to as a task stream. Each task $\mathcal{D}^i$ originates from a different database domain, formally, for $\forall\mathcal{D}^i$ and $\forall\mathcal{D}^j$, if $i \neq j$, then $\mathcal{S}(\mathcal{D}^i) \cap \mathcal{S}(\mathcal{D}^j) = \emptyset$, where $\mathcal{S}(\mathcal{D}^i)$ represents the set of all database schemas $\mathcal{S}$ in task $\mathcal{D}^i$. Additionally, each task $\mathcal{D}^i$ has its own training, validation, and test sets. The primary objective of CSP is to develop a semantic parser that not only performs excellently on previous tasks but also effectively generalizes to future unseen tasks after being sequentially trained on all tasks.

Methodology

Our method consists of two main stages: memory reconstruction and dual distillation learning. In the first stage (Figure 2), when a new task arrives, domain-specific information is first masked to eliminate any irrelevant context, and component biases are computed to help the model connect with prior knowledge from an SQL syntax perspective, identifying the key knowledge features of the current task and its differences from previous tasks. Then, intra-task memory is reinforced, memory gaps between tasks are filled, and memory is calibrated. In the second stage (Figure 3), after acquiring key memories, a task-aware dual-teacher distillation learning framework is designed to facilitate the accumulation and transfer of memory in the continuous learning process.

Reconstruct Memory with LLMs

Domain Information Elimination To mitigate the impact of database domain differences between tasks, we remove domain-specific information from each sample, as detailed in Figure 2(a). This allows us to uncover the commonalities and differences between individual tasks from a syntax perspective. Specifically, following previous works (Yu et al. 2021; Li et al. 2023a), we first employ the string matching method to establish entity links between questions and SQL based on the database schema $\mathcal{S}$. Subsequently, we mask all linked entities in the question to obtain $\mathcal{Q}^{de}$, and retain only the original syntactic structure in SQL to get $\mathcal{Z}$, which we named SQL skeleton. Finally, we merge these two parts to form the joint input pair $(\mathcal{Q}^{de}, \mathcal{Z})$ for further analysis.

Component Bias Analysis After the above process, we analyze component bias in the current task $\mathcal{D}^t$ and all historical tasks $\cup_{i=1}^{t-1}\mathcal{D}^i$, identifying features that present in historical tasks but are absent in the current one, as shown in Figure 2(b). Specifically, given the program-like attributes of SQL, we use CodeT5 (Wang et al. 2021b) to encode $(\mathcal{Q}^{de}, \mathcal{Z})$ from all training samples of the current task and apply K-means clustering to the obtained representations. We then extract the SQL skeleton $\mathcal{Z}$ corresponding to the sample closest to each cluster center, forming the component feature set for task $\mathcal{D}^t$, denoted as $\mathcal{A}^{(t)} = \{\mathcal{Z}_1^{(t)}, \mathcal{Z}_2^{(t)}, ..., \mathcal{Z}_K^{(t)}\}$, where K is the number of cluster centers. Finally, we select individuals that are present in the stored component feature sets of all previous tasks $\cup_{i=1}^{t-1}\mathcal{D}^i$ but absent in the current task $\mathcal{D}^t$, forming the component bias $\Delta\mathcal{A}^{(t)}$. This bias is then utilized to guide subsequent memory reconstruction, mitigating the model's forgetting of

these crucial features. When $t > 1$, the formula is as follows:

$$\Delta\mathcal{A}^{(t)} = (\mathcal{A}^{(1)} \cup \mathcal{A}^{(2)} \cup ... \cup \mathcal{A}^{(t-1)}) - \mathcal{A}^{(t)} \quad (1)$$

Importantly, to get $\Delta\mathcal{A}^{(t)}$, we need to save each task's $\mathcal{A}^{(t)}$, which only contains the SQL skeleton related to syntax and does not involve any real historical database information or samples, ensuring minimal storage costs. This highlights our method's feature of not requiring data replay.

Inter-Task Memory Completion We employ the obtained component bias $\Delta\mathcal{A}^{(t)}$ to guide LLMs in generating pseudo-samples for the current task, as shown in Figure 2(c). This process aims to help the model fill in the memory gaps between the current and past tasks, while also uncovering knowledge relevant to the current domain from LLMs. Unlike previous strategies that generate pseudo-data through semi-supervised learning methods such as self-training (Chen et al. 2023a), these methods require collecting task-related questions in advance and then constructing pseudo-labels for them. In reality, obtaining accurate pseudo-labels for complex tasks like semantic parsing, relying solely on limited annotated data, is itself a challenge for most LLMs (Li et al. 2024). Therefore, we propose a soft pseudo-sample construction strategy that uses SQL skeletons to guide LLMs to simultaneously generate the natural language question $\hat{\mathcal{Q}}$ and corresponding SQL query $\hat{\mathcal{Y}}$ on the current task's database schema $\mathcal{S}^{(t)}$. The generation process can be summarized as follows:

$$(\hat{\mathcal{Q}}, \hat{\mathcal{Y}})^{(t)} = \begin{cases} \mathcal{F}_{\text{LLM}}(\mathcal{A}^{(t)}, \mathcal{S}^{(t)}), & \text{if } t = 1, \\ \mathcal{F}_{\text{LLM}}(\Delta\mathcal{A}^{(t)}, \mathcal{S}^{(t)}), & \text{if } t > 1. \end{cases} \quad (2)$$

Where $(\hat{\mathcal{Q}}, \hat{\mathcal{Y}})^{(t)}$ refers to the pseudo-samples generated by LLMs. On the first task, we use SQL skeletons $\mathcal{A}^{(t)}$ for better initialization. Each SQL skeleton generates N_{ske} data.

Inter-Task Memory Calibration Although our proposed soft pseudo-sample construction strategy avoids the challenge of directly requiring the model to generate complex SQL queries and further leverages the generative capabilities of LLMs, it also faces the issue of hallucinations (Ji et al. 2023), which can lead to noise. To address this, we design a memory calibration strategy that includes two stages: iterative self-correction and SQL skeleton-based sampling. This aims to improve the accuracy of generated pseudo-samples and ensure fidelity to specific SQL skeletons.

The inspiration for iterative self-correction comes from previous works that used LLMs for code debugging and fixing (Pourreza and Rafiei 2023; Chen et al. 2024). We iteratively execute SQL queries and correct the generated pseudo-samples, ultimately retaining only the samples that LLMs consider correct and whose SQL queries can be executed successfully. Additionally, we sample pseudo-samples that best match the specified SQL skeleton by calculating the edit distance after removing domain information.

Intra-Task Memory Reinforcement Following previous work (Yu et al. 2021), we introduce a data synthesis method based on context-free grammar, utilizing the component features $\mathcal{A}^{(t)}$ of the current task $\mathcal{D}^t$ to enhance memory. Specifically, for each annotated sample, we synthesize N_{cfg} new instances $\hat{X}^{(t)}_{cfg}$ by synchronously replacing entities in the natural language question $\mathcal{Q}$ and SQL query $\mathcal{Y}$ with other database content based on entity linking results, as illustrated in Figure 2(d). To ensure quality, only SQL queries that pass execution tests are retained. Additionally, we use LLMs to rephrase questions to enhance data diversity.

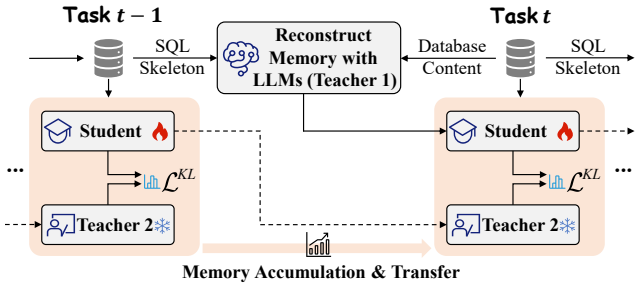

Figure 3: The task-aware dual-teacher distillation learning framework of LECSP.

Task-Aware Dual-Teacher Distillation Learning Framework

After reconstructing the relevant inter-task and intra-task memories, we design a task-aware dual-teacher distillation framework to promote the efficient utilization of these memories. The training framework is shown in Figure 3. Specifically, for each task, a *Student* model is trained to learn from two *Teacher* models: one from the LLM and the other from the previous student.

LLMs as Teacher 1 This process facilitates knowledge transfer from the LLMs (*Teacher 1*) to the smaller model (*Student*) via reconstructed memory, consisting of two types of pseudo-data: $\hat{X}_{ske}$ and $\hat{X}_{cfg}$. They respectively represent shared knowledge accumulated from historical tasks and specific domain knowledge related to the current task. We utilize these pseudo datasets for additional supervised training of the smaller student model. The loss function is defined as follows:

$$\mathcal{L}_{\text{cur/past}} = \frac{1}{N^{c/s}} \sum_{i=1}^{N^{c/s}} \ell(\mathcal{F}_\theta(\hat{x}_i^{c/s}), \hat{y}_i^{c/s}) \quad (3)$$

Where N^c, N^s respectively represent the number of data generated for types $\hat{X}_{cfg}$ and $\hat{X}_{ske}$. $\hat{x}_i$ and $\hat{y}_i$ with different superscripts denote the input and target output for the corresponding data types. ℓ means the cross-entropy loss.

Previous Student as Teacher 2 Although $\hat{X}_{ske}$ already covers knowledge from past tasks that is missing in the current task, this knowledge is primarily derived from indirect transfer by the teacher LLM, which may affect the stability and introduce inherent biases from the LLM. Therefore, we introduce another *Teacher 2* (the student model from the previous task) to directly constrain the learning of the current student. We encourage both models to maintain consistent representations of the shared knowledge in memory $\hat{X}_{ske}$.

Specifically, inspired by previous work (Qin and Joty 2022), after freezing the parameters of the student model from the previous task, we use KL divergence as the distillation loss to promote consistency in the output distributions of the two models. The specific process is as follows:

$$\mathcal{L}_{past}^{KL} = \frac{1}{N^s} \sum_{i=1}^{N^s} \sum_{j=1}^{l} D_{\text{KL}}(P_j(\hat{x}_i^s, \theta') || P_j(\hat{x}_i^s, \theta)) \quad (4)$$

Where l is the number of tokens in the output $\hat{y}_i^s$ corresponding to the input $\hat{x}_i^s$, and $P_j(\cdot)$ denotes the probability distribution over the vocabulary of model $\mathcal{F}(\cdot)$ when generating the j-th token. θ and θ' respectively represent the parameters of the current and previous task student models. Note that we do not calculate $\mathcal{L}_{past}^{KL}$ on the first task.

Total loss function The loss of the student model on the original annotated data follows a similar calculation process as previously described, and it can be represented as $\mathcal{L}_{task} = \frac{1}{N} \sum_{i=1}^{N} \ell(\mathcal{F}_\theta(x_i), y_i)$, N is the count of original annotated data. The final loss is as follows, where λ represents the weight factor:

$$\mathcal{L} = \mathcal{L}_{task} + \mathcal{L}_{cur} + \mathcal{L}_{past} + \lambda \mathcal{L}_{past}^{KL} \quad (5)$$

Experiments

Datasets We evaluate our method on the publicly available Spider-stream-semi (Chen et al. 2023a) and Combined-stream (Chen et al. 2023b) datasets. Spider-stream-semi is derived from Spider dataset and consists of 10 tasks involving complex SQL syntax such as `JOIN` and `GROUP BY`. Each task also provides unlabeled data intended for semi-supervised learning baselines, but our approach does not require these data. Most tasks have fewer than 500 annotated samples to evaluate the impact of database domain changes under limited resources. Combined-stream consists of Spider and WikiSQL datasets, with a total of 7 tasks. Unlike Spider-stream-semi, it introduces simple SQL syntax for single-table queries to assess the dual impact of table domain and SQL structure changes. In previous works, to ensure good initial performance, both Spider-stream-semi and Combined-stream set the task with the most annotated data as the first task, termed a *warm start*. Conversely, we also introduce a variant called *cold start*, which is more reflective of real-world scenarios. In this variant, tasks are randomly shuffled without assigning the first task the most annotated data. This helps us further study the effects of initial performance and task order on the model.

Evaluation Metrics Following previous works (Yu et al. 2018; Chen et al. 2023a,b), we define $a_{m,n}$ as the model's accuracy on the n-th task's test set after training on the m-th task, split into Exact-set-Match (EM) and EXecution (EX) accuracy. EM assesses the formal accuracy of SQL queries and EX measures their execution results. (1) *Average accuracy*: $\text{ACC}_\text{a} = \frac{1}{M} \sum_{i=1}^{M} a_{M,i}$. It reflects the model's overall performance across all historical tasks. (2) *Whole accuracy*: $\text{ACC}_\text{w} = a_{\mathcal{D}_{test}^{(1:M)}}$, where $\mathcal{D}_{test}^{(1:M)} = \cup_{i=1}^{M} \mathcal{D}_{test}^{i}$. It measures accuracy on all tasks' combined test sets after training on task M, reflecting historical performance changes. (3) *Backward transfer*: $\text{BWT} = \frac{1}{M-1} \sum_{i=1}^{M-1} (a_{M,i} - a_{i,i})$. It assesses the average impact of learning the M-th task on the performance of all previous tasks. (4) *Forward transfer*: $\text{FWT} = \frac{1}{M-1} \sum_{i=2}^{M} (a_{i-1,i} - \hat{a}_i)$, where $\hat{a}_i$ represents the test accuracy of a randomly initialized reference model on the i-th task, examining its generalization performance.

Implementation Details In our experiments, we utilize T5-base and T5-large as backbone models, applying the Adafactor (Shazeer and Stern 2018) optimizer. The maximum input and output lengths are configured to 512 and 256, respectively. In the component bias analysis, we choose CodeT5 as the encoder and set the number of cluster centers K to 80. In the data generation process, we set N_{ske} as 10 and N_{cfg} as 3. The weight hyperparameter λ in the training process undergoes grid search on the set {0.03, 0.05, 0.1, 0.2, 0.3}, and 0.1 is selected. For LLMs, We employ the publicly available Mixtral-8x7B-Instruct-v0.1, which is a sparse mixture of experts' language model. All experiments are performed on A100 GPUs with 80GB memory.

Baselines Our proposed method is compared with three types of baselines. (1) *Original method*: Sequential Fine-Tuning (**SFT**) (Yogatama et al. 2019). (2) *Rehearsal-based methods*: **EMAR** (Han et al. 2020), **SFNET** (Chen et al. 2023a). (3) *PET-based methods*: **PEFT** (Chen et al. 2023b), **C3** (Chen et al. 2023b), **ProgPrompt** (Razdaibiedina et al. 2023). Since our method uses LLMs as *Teacher 1* for zero-shot memory construction, we also report the zero-shot performance of a single LLM for comparison. The **ORACLE** setting trains the model incrementally on all seen tasks' data, representing the CSP performance upper bound.

Results and Analysis

Overall Results

Table 1 shows the performance of each method across various backbones and benchmarks, noting whether data replay strategies are used. We also list baselines under other configurations, such as ideal settings, LLM integration, and ORACLE performance. The results show that our method is competitive across all scenarios, requiring no external data or ideal settings, and better adapts to real-world applications.

Comparison with PET-based Methods LECSP outperforms PEFT by up tp 11.1% on ACC_a. PEFT uses an ideal continual learning setup that sacrifices model generalization, as it is challenging to select the appropriate PET module for unseen samples, making FWT calculation impossible. We also note that ProgPrompt does not require ideal settings, but its performs worse than SFT. This is because ProgPrompt is designed for tasks with simpler output structures, such as classification. However, for models pre-trained on natural language corpora, adapting to complex tasks like semantic parsing with limited-length prompts is challenging (Qin and Joty 2022). The progressive prompt design of ProgPrompt further compresses the prompt length for individual tasks, worsening the issue and making it nearly non-functional in *cold start* scenarios. Notably, PET-based methods with

Method	DR	Spider-stream-semi					Combined-stream				
		ACC_a	ACC_w	BWT	FWT	Δ	ACC_a	ACC_w	BWT	FWT	Δ
T5-Base(220M)											
SFT	✗	41.7/45.3	40.1/43.7	-13.6/-11.7	22.5/25.2	↓1.5	51.0/36.0	52.4/38.3	-11.1/-17.6	19.9/11.1	↓11.1
ProgPrompt	✗	14.9/16.5	14.3/16.4	-25.6/-26.3	14.7/15.8	↓14.9	21.3/19.2	27.2/26.8	-43.9/-41.2	6.1/3.3	↓21.1
EMAR	✓	44.1/48.0	43.0/47.0	-11.8/-9.7	23.7/25.2	↑2.3	60.6/58.9	55.7/55.3	-7.9/-7.1	28.7/25.4	↓5.6
SFNET♣	✗	39.5/43.9	38.1/42.7	-14.6/-12.4	22.7/25.7	↑0.4	56.2/49.4	51.8/48.4	-7.9/-14.4	21.0/14.6	↓21.7
SFNET♣	✓	45.8/48.1	44.2/47.7	-6.4/-6.4	23.9/26.2	↓3.4	60.7/57.7	55.2/53.9	-6.0/-6.7	28.7/26.0	↓6.7
LECSP(Ours)	✗	**47.4/53.7**	**46.5/53.3**	**-4.7/-4.6**	**30.1/32.8**	↑1.4	**63.4/61.7**	**60.5/59.8**	**-4.6/4.3**	**32.7/31.5**	↓4.8
Ideal Setting & Upper Bound											
PEFT♢	✗	40.6/43.8	44.6/48.4	0.0/0.0	-/-	↓32.5	63.8/-	66.2/-	0.0/0.0	-/-	↓38.8
ORACLE		60.3/61.3	62.8/63.9	4.6/3.3	25.7/27.9	↓1.1	70.2/68.2	71.1/70.8	5.6/4.7	29.2/26.6	↑0.3
T5-Large(770M)											
SFT	✗	46.4/50.4	44.3/48.9	-13.2/-11.5	29.0/31.5	↑1.8	57.8/48.9	58.9/53.3	-7.6/-16.4	28.3/25.4	↓8.3
ProgPrompt	✗	22.1/23.9	22.4/24.6	-27.0/-27.5	21.1/23.1	↓20.8	23.5/20.2	29.8/27.7	-46.4/-42.7	10.2/3.7	↓22.4
EMAR	✓	48.9/51.8	48.4/52.5	-8.5/-8.3	30.3/33.5	↑1.4	63.9/61.9	60.6/60.4	-6.6/-8.9	33.8/29.3	↓5.5
SFNET♣	✗	44.2/48.2	43.3/47.8	-12.3/-9.9	29.9/31.0	↑3.3	65.0/60.3	61.5/59.7	-7.5/-11.9	28.2/20.4	↓24.1
SFNET♣	✓	52.2/55.0	52.0/55.8	-6.5/**-4.2**	34.1/35.5	↓2.2	65.3/64.7	61.8/61.9	-5.7/-5.2	34.9/31.1	↓5.6
LECSP(Ours)	✗	**58.6/63.5**	**57.4/63.8**	**-5.8**/-5.0	**41.4/44.3**	↑0.8	**68.2/66.7**	**66.5/65.4**	**-2.9/-3.7**	**37.1/34.5**	↓5.1
Ideal Setting & LLMs Performance & Upper Bound											
PEFT♢	✗	49.6/52.4	53.4/56.4	0.0/0.0	-/-	↓24.1	67.3/-	70.0/-	0.0/0.0	-/-	↓23.4
C3♢	✗	-/-	-/-	-/-	-/-	-	69.0/-	71.2/-	0.0/0.0	-/-	-
C3♢†	✗	-/-	-/-	-/-	-/-	-	67.6/-	70.0/-	0.0/0.0	-/-	-
Mixtral-8x7B	✗	22.0/49.4	22.2/52.4	-/-	-/-	-	28.6/24.4	27.0/26.0	-/-	-/-	-
ORACLE		66.6/68.2	68.6/70.1	3.7/4.0	34.3/36.0	↑0.4	73.7/73.2	75.8/76.0	1.9/2.3	37.3/34.2	↓0.1

Table 1: Results (EM/EX) on Spider-stream-semi and Combined-stream datasets (%). LECSP uses Mixtral-8x7B as *Teacher 1*. DR indicates whether historical data replay is used, with a memory size set to 10. Δ represents the performance change (ACC_a–EM) from the current *warm start* to the *cold start* scenario. ♣ indicates the need for additional unsupervised data, and ♢ means an ideal continuous learning setting is used, but forward transfer is impossible. The results of PEFT and C3 on the Combined-stream are from the original paper and its official code repository. † signifies the use of GPT-3.5 as the teacher. The best results are highlighted in bold, and the second-best results are underlined. Our results are the average of three random runs.

ideal settings, like PEFT, heavily rely on the initial task's labeled data. In *cold start* scenarios, their performance drops sharply, with a maximum decline of 38.8%.

Comparison with Rehearsal-based Methods Compared to the previous SOTA method SFNET, LECSP achieves up to an 8.8% improvement in FWT on Spider-stream-semi, despite SFNET using additional unlabeled data and data replay strategies. This improvement is attributed to LECSP's efficient use of knowledge from learned tasks and LLMs. Notably, SFNET (*w/o* DR) performs worse than SFT, likely due to the exacerbated noise from pseudo-labels in self-training, highlighting its reliance on data replay. In contrast, our method operates without relying on any external data.

Comparison with ORACLE and LLMs Impressively, our method outperforms most ORACLE baselines in FWT performance, with up to an 8.3% improvement, demonstrating effective knowledge transfer from LLMs to smaller models and enhancing their generalization ability. In contrast, the zero-shot performance of standalone LLMs is poor, even worse than the naive SFT baseline, and continuous training of LLMs is often challenging and resource-intensive, further highlighting the importance of our approach. By combining models of different scales, we effectively address this issue, carefully balancing cost and performance.

Detailed Results and Analysis

Results Till the Seen Tasks Figure 4 shows that as task numbers increase, LECSP (red) consistently outperforms other baselines in ACC_a and ACC_w, closely approaching the ORACLE (grey) and effectively mitigating forgetting. It also maintains stable BWT performance, especially in the early stages, thanks to its ability to quickly connect with previously learned knowledge. Notably, LECSP surpasses ORACLE in FWT, highlighting effective knowledge accumulation and transfer from LLM and past tasks.

Influence of Task Order and Cold Start Unlike previous works that start with data-rich tasks, we further explore the cold-start scenario with random task order and limited data. The results in Table 1 show a significant performance drop for most baselines, especially *PET-based* methods like PEFT and ProgPrompt, reflecting their strong dependence on labeled data of the initial task. In contrast, LECSP is less affected in this challenging scenario, demonstrating strong robustness and maintaining a significant advantage.

Influence of SQL Syntax Variance To further explore the role of SQL syntax variance in LECSP, we compare the impact on model performance between our method and randomly obtained component bias. Specifically, we swap the memories constructed under different orders of Spider-stream-semi and Spider-stream-semi (*cold start*), keep other

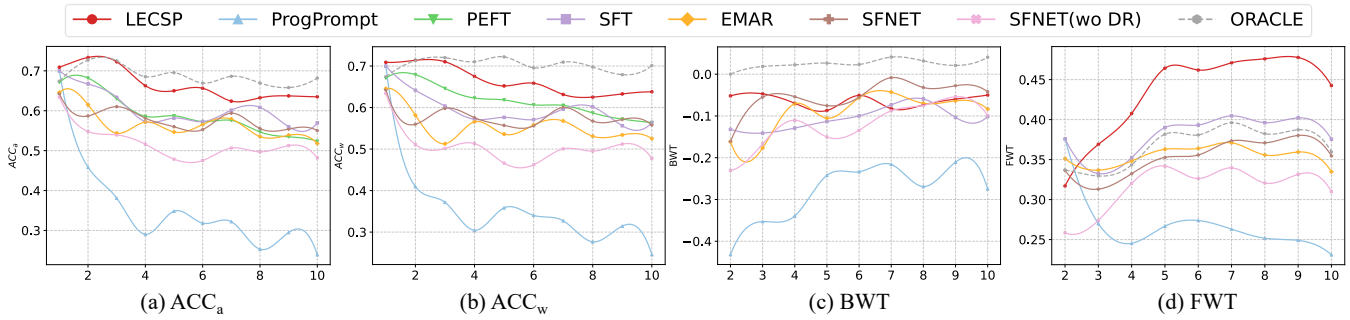

Figure 4: Results (EX) till the seen tasks based on Spider-stream-semi (T5-large).

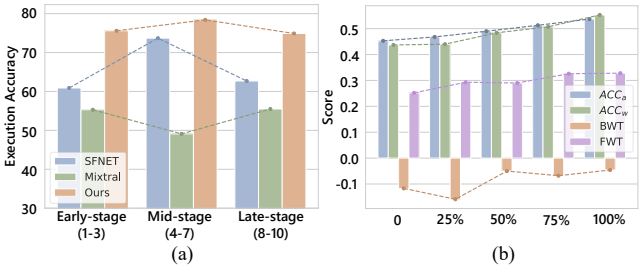

Figure 5: (a) Accuracy of synthetic data execution across different stages. (b) Impact of pseudo-sample quantity on different metrics (T5-base).

	ACC$_a$	ACC$_w$	BWT	FWT
S$_S$	47.4/53.7	46.5/53.3	-4.7/-4.6	30.1/32.8
S$_C$	45.0/49.1	43.7/48.0	-4.5/-5.7	28.1/31.7
C$_C$	48.8/52.6	49.9/54.2	-1.1/-0.9	24.8/27.3
C$_S$	43.6/48.1	44.2/49.2	-4.4/-5.6	27.1/30.2

Table 2: The impact of SQL syntax variance on model performance (%). S$_S$ denotes using component bias from Spider-stream-semi itself, as does C$_C$. S$_C$ indicates using component bias from Spider-stream-semi (Cold start) to construct memory for Spider-stream-semi, and vice versa.

experimental settings unchanged, and retrain the models. The results in Table 2 show that, after the exchange, model performance decreases on most metrics but improves in FWT on C$_S$. This improvement may be due to the introduction of unseen SQL syntax, enhancing generalization.

Quality and Quantity of Pseudo-Samples To evaluate the quality of generated memory (pseudo-samples), we compare our method with the self-training method used in SFNET (T5-large) and the zero-shot results of Mixtral-8x7B, both using additional unsupervised data. We divide the original Spider-stream-semi into early, mid, and late stages, randomly selecting 70 pseudo-samples from each stage to compare SQL execution accuracy. For our method, we use manual evaluation, whereas for the other two methods, we rely on their original labels. The results in Figure 5(a) show that our method significantly improves the quality of pseudo-samples, while the self-training method in SFNET is less stable. Figure 5(b) shows that as the proportion of pseudo-samples increases, all metrics generally improve, but several metrics stabilize in the later stages.

	ACC$_a$	ACC$_w$	BWT	FWT
LECSP	47.4/53.7	46.5/53.3	-4.7/-4.6	30.1/32.8
- $\hat{X}_{cfg}$	45.6/50.3	44.3/49.1	-5.6/-6.6	27.7/30.7
- MC	20.3/22.9	19.4/21.9	0.4/-0.6	15.8/17.5
- T2	46.6/52.8	45.5/51.8	-6.6/-5.9	30.4/33.3
- $\hat{X}_{ske}$	41.7/45.3	40.1/43.7	-13.6/-11.7	22.5/25.2

Table 3: Ablation study results on Spider-stream-semi.

Ablation Study We decompose LECSP (T5-base) and sequentially remove each component to assess its impact on performance. Table 3 shows that the full LECSP achieves the best overall performance. Omitting intra-task memory $\hat{X}_{cfg}$ or inter-task memory $\hat{X}_{ske}$ reduces performance, with the latter significantly improving forgetting mitigation, with gains of 5.7%-9.6%, as it retains SQL knowledge from previous tasks. Removing the Memory Calibration (MC) strategy leads to a sharp performance drop of up to 31.4%, highlighting the importance of MC in mitigating hallucinations in LLM-generated memories (Ji et al. 2023). Ignoring *Teacher 2* (T2) leads to an overall performance decline (0.8%-1.9%) despite minor gains on FWT (0.3%-0.5%).

Conclusion

In this paper, we introduce LECSP, a novel CSP method without real data replay or ideal settings. To handle incoming new tasks, LECSP extracts key SQL syntax features and analyzes the commonalities and differences with historical task knowledge, guiding LLMs to generate pseudo-samples to reconstruct key memories. We also design a task-aware dual-teacher distillation framework to transfer knowledge from the LLM and previous tasks to a smaller model. Experimental results show that, even with auxiliary strategies in the baseline, LECSP achieves competitive performance and overcomes certain performance limits, demonstrating strong adaptability to challenging real-world scenarios.

Acknowledgments

We sincerely thank the anonymous reviewers for their insightful comments and suggestions. This work was supported by the National Natural Science Foundation of China (No. 62476066 and No. 62277002).

References

Anil, R.; Dai, A. M.; Firat, O.; Johnson, M.; Lepikhin, D.; Passos, A.; Shakeri, S.; Taropa, E.; Bailey, P.; Chen, Z.; and et al., E. C. 2023. PaLM 2 Technical Report. arXiv:2305.10403.

Cai, Z.; Li, X.; Hui, B.; Yang, M.; Li, B.; Li, B.; Cao, Z.; Li, W.; Huang, F.; Si, L.; and Li, Y. 2022. STAR: SQL Guided Pre-Training for Context-dependent Text-to-SQL Parsing. In Goldberg, Y.; Kozareva, Z.; and Zhang, Y., eds., *Findings of the Association for Computational Linguistics: EMNLP 2022*, 1235–1247. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

Chen, X.; Lin, M.; Schärli, N.; and Zhou, D. 2024. Teaching Large Language Models to Self-Debug. In *The Twelfth International Conference on Learning Representations*.

Chen, Y.; Guo, X.; Wu, T.; Qi, G.; Li, Y.; and Dong, Y. 2023a. Learn from Yesterday: a semi-supervised continual learning method for supervision-limited text-to-SQL task streams. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press. ISBN 978-1-57735-880-0.

Chen, Y.; Zhang, S.; Qi, G.; and Guo, X. 2023b. Parameterizing Context: Unleashing the Power of Parameter-Efficient Fine-Tuning and In-Context Tuning for Continual Table Semantic Parsing. In Oh, A.; Neumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 17795–17810. Curran Associates, Inc.

d'Autume, C. d. M.; Ruder, S.; Kong, L.; and Yogatama, D. 2019. Episodic memory in lifelong language learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.

Dou, L.; Gao, Y.; Pan, M.; Wang, D.; Che, W.; Lou, J.-G.; and Zhan, D. 2023. UniSAr: a unified structure-aware autoregressive language model for text-to-SQL semantic parsing. *International Journal of Machine Learning and Cybernetics*, 14(12): 4361–4376.

Guo, C.; Tian, Z.; Tang, J.; Li, S.; Wen, Z.; Wang, K.; and Wang, T. 2024. Retrieval-Augmented GPT-3.5-Based Text-to-SQL Framework with Sample-Aware Prompting and Dynamic Revision Chain. In Luo, B.; Cheng, L.; Wu, Z.-G.; Li, H.; and Li, C., eds., *Neural Information Processing*, 341–356. Singapore: Springer Nature Singapore. ISBN 978-981-99-8076-5.

Han, X.; Dai, Y.; Gao, T.; Lin, Y.; Liu, Z.; Li, P.; Sun, M.; and Zhou, J. 2020. Continual Relation Learning via Episodic Memory Activation and Reconsolidation. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6429–6440. Online: Association for Computational Linguistics.

Hu, C.; Fu, J.; Du, C.; Luo, S.; Zhao, J.; and Zhao, H. 2023. ChatDB: Augmenting LLMs with Databases as Their Symbolic Memory. arXiv:2306.03901.

Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y. J.; Madotto, A.; and Fung, P. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.*, 55(12).

Jung, D.; Han, D.; Bang, J.; and Song, H. 2023. Generating Instance-level Prompts for Rehearsal-free Continual Learning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 11813–11823.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3045–3059. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Li, H.; Zhang, J.; Li, C.; and Chen, H. 2023a. RESDSQL: decoupling schema linking and skeleton parsing for text-to-SQL. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press. ISBN 978-1-57735-880-0.

Li, H.; Zhang, J.; Liu, H.; Fan, J.; Zhang, X.; Zhu, J.; Wei, R.; Pan, H.; Li, C.; and Chen, H. 2024. CodeS: Towards Building Open-source Language Models for Text-to-SQL. *Proc. ACM Manag. Data*, 2(3).

Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; Zhou, X.; Chenhao, M.; Li, G.; Chang, K.; Huang, F.; Cheng, R.; and Li, Y. 2023b. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. In Oh, A.; Neumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 42330–42357. Curran Associates, Inc.

Li, Z.; Qu, L.; and Haffari, G. 2021. Total Recall: a Customized Continual Learning Method for Neural Semantic Parsers. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3816–3831. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Lialin, V.; Goel, R.; Simanovsky, A.; Rumshisky, A.; and Shah, R. 2021. Update Frequently, Update Fast: Retraining Semantic Parsing Systems in a Fraction of Time. arXiv:2010.07865.

Liang, Y.; Meng, F.; Wang, J.; Xu, J.; Chen, Y.; and Zhou, J. 2024. Continual Learning with Semi-supervised Contrastive Distillation for Incremental Neural Machine Translation. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings*

of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 10914–10928. Bangkok, Thailand: Association for Computational Linguistics.

Mao, W.; Wang, R.; Guo, J.; Zeng, J.; Gao, C.; Han, P.; and Liu, C. 2024. Enhancing Text-to-SQL Parsing through Question Rewriting and Execution-Guided Refinement. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics ACL 2024*, 2009–2024. Bangkok, Thailand and virtual meeting: Association for Computational Linguistics.

OpenAI; :; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; and Shyamal Anadkat, e. a. 2024. GPT-4 Technical Report. arXiv:2303.08774.

Pourreza, M.; and Rafiei, D. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In Oh, A.; Neumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 36339–36348. Curran Associates, Inc.

Qiao, F.; and Mahdavi, M. 2024. Learn more, but bother less: parameter efficient continual learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Qin, C.; and Joty, S. 2022. LFPT5: A Unified Framework for Lifelong Few-shot Language Learning Based on Prompt Tuning of T5. In *International Conference on Learning Representations*.

Razdaibiedina, A.; Mao, Y.; Hou, R.; Khabsa, M.; Lewis, M.; and Almahairi, A. 2023. Progressive Prompts: Continual Learning for Language Models. In *The Eleventh International Conference on Learning Representations*.

Shazeer, N.; and Stern, M. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4596–4604. PMLR.

Wang, D.; Dou, L.; Zhang, X.; Zhu, Q.; and Che, W. 2024a. Improving Demonstration Diversity by Human-Free Fusing for Text-to-SQL. arXiv:2402.10663.

Wang, L.; Zhang, X.; Li, Q.; Zhang, M.; Su, H.; Zhu, J.; and Zhong, Y. 2023a. Incorporating neuro-inspired adaptability for continual learning in artificial intelligence. *Nature Machine Intelligence*, 5(12): 1356–1368.

Wang, L.; Zhang, X.; Su, H.; and Zhu, J. 2024b. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–20.

Wang, L.; Zhang, X.; Yang, K.; Yu, L.; Li, C.; HONG, L.; Zhang, S.; Li, Z.; Zhong, Y.; and Zhu, J. 2022. Memory Replay with Data Compression for Continual Learning. In *International Conference on Learning Representations*.

Wang, X.; Zhang, S.; Qing, Z.; Shao, Y.; Gao, C.; and Sang, N. 2021a. Self-Supervised Learning for Semi-Supervised Temporal Action Proposal. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1905–1914.

Wang, Y.; Wang, W.; Joty, S.; and Hoi, S. C. 2021b. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 8696–8708. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Wang, Z.; Liu, Y.; Ji, T.; Wang, X.; Wu, Y.; Jiang, C.; Chao, Y.; Han, Z.; Wang, L.; Shao, X.; and Zeng, W. 2023b. Rehearsal-free Continual Language Learning via Efficient Parameter Isolation. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 10933–10946. Toronto, Canada: Association for Computational Linguistics.

Xue, S.; Jiang, C.; Shi, W.; Cheng, F.; Chen, K.; Yang, H.; Zhang, Z.; He, J.; Zhang, H.; Wei, G.; Zhao, W.; Zhou, F.; Qi, D.; Yi, H.; Liu, S.; and Chen, F. 2024. DB-GPT: Empowering Database Interactions with Private Large Language Models. arXiv:2312.17449.

Yadav, P.; Sun, Q.; Ding, H.; Li, X.; Zhang, D.; Tan, M.; Bhatia, P.; Ma, X.; Nallapati, R.; Ramanathan, M. K.; Bansal, M.; and Xiang, B. 2023. Exploring Continual Learning for Code Generation Models. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 782–792. Toronto, Canada: Association for Computational Linguistics.

Yogatama, D.; de Masson d'Autume, C.; Connor, J.; Kocisky, T.; Chrzanowski, M.; Kong, L.; Lazaridou, A.; Ling, W.; Yu, L.; Dyer, C.; and Blunsom, P. 2019. Learning and Evaluating General Linguistic Intelligence. arXiv:1901.11373.

Yu, T.; Wu, C.-S.; Lin, X. V.; bailin wang; Tan, Y. C.; Yang, X.; Radev, D.; richard socher; and Xiong, C. 2021. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. In *International Conference on Learning Representations*.

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; Zhang, Z.; and Radev, D. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In Riloff, E.; Chiang, D.; Hockenmaier, J.; and Tsujii, J., eds., *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3911–3921. Brussels, Belgium: Association for Computational Linguistics.

Zhao, W.; Wang, S.; Hu, Y.; Zhao, Y.; Qin, B.; Zhang, X.; Yang, Q.; Xu, D.; and Che, W. 2024. SAPT: A Shared Attention Framework for Parameter-Efficient Continual Learning of Large Language Models. arXiv:2401.08295.