Discarding the Crutches: Adaptive Parameter-Efficient Expert Meta-Learning for Continual Semantic Parsing

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) enables parsers to generate SQL from natural language questions in task streams, using minimal annotated data to handle dynamically evolving databases in real-world scenarios. Previous works often rely on replaying historical data, which poses privacy concerns. Recently, replay-free continual learning methods based on Parameter-Efficient Tuning (PET) have gained widespread attention. However, they often rely on ideal settings and initial task data, sacrificing the model's generalization ability, which limits their applicability in real-world scenarios. To address this, we propose a novel Adaptive PET eXpert meta-learning (APEX) approach for CSP. First, SQL syntax guides the LLM to assist experts in adaptively warming up, ensuring better model initialization. Then, a dynamically expanding expert pool stores knowledge and explores the relationship between experts and instances. Finally, a selection/fusion inference strategy based on sample historical visibility promotes expert collaboration. Experiments on two CSP benchmarks show that our method achieves superior performance without data replay or ideal settings, effectively handling cold-start scenarios and generalizing to unseen tasks, even surpassing performance upper bounds[1].

1 Introduction

The rapid development of Large Language Models (LLMs) has greatly advanced semantic parsing, providing non-expert users with a convenient interface for querying and analyzing large-scale structured data (Yang et al., 2024; Zhang et al., 2024a; Qu et al., 2024; Li et al., 2024). However, frequent database updates require semantic parsers to adapt to evolving data environments dynamically.

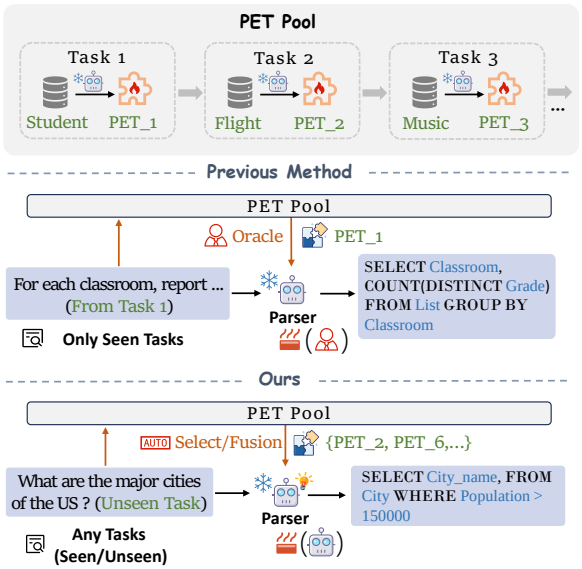

Figure 1: CSP process based on the PET method. Each task in the task stream learns a dedicated PET module and adds it to the pool. **Previous method**: ① Require manual data collection for warming up parsers. ② Assume known task identifier and load Oracle PET modules. ③ Handle only seen tasks, unable to generalize to new tasks. **Our method**: ① No need for manual warm up data. ② Model autonomously selects/fuses PET modules. ③ Handle both seen and unseen tasks.

Consequently, task stream-oriented continual semantic parsing has garnered significant attention from academia and industry, presenting three major challenges: (1) **Few-shot Challenge**: For new databases, obtaining sufficient text-to-SQL pairs in a short time is difficult, which may lead to overfitting of the parser (Chen et al., 2023b). (2) **Catastrophic Forgetting**: In continual learning, a model may forget previously learned knowledge while acquiring new information (Wang et al., 2024). (3) **Knowledge Transfer**: Accumulating and generalizing knowledge from task sequences to unseen data is challenging (Zhao et al., 2024).

To tackle these challenges, recent studies have explored PET methods to prevent forgetting caused

*Corresponding Authors.
[1]Codes are publicly available at https://github.com/tom68-ll/APEX

Proceedings of the 31st International Conference on Computational Linguistics, pages 3560–3578
January 19–24, 2025.

by extensive parameter updates (Wang et al., 2022; Chen et al., 2023c; Yadav et al., 2023). As shown in Figure 1, PET modules such as prompt (Lester et al., 2021) are trained to capture task-specific knowledge and stored in a PET pool. During testing, the appropriate PET module is selected from the pool. While these methods have made some progress in reducing training overhead and preventing catastrophic forgetting, they still have limitations: (1) **Ideal Setting**: Current methods perform well in preventing catastrophic forgetting but rely on a simplified scenario where task identifiers are known during testing, enabling the selection of the appropriate PET module. This assumption ignores the shared SQL syntax across tasks, limiting knowledge transfer and accumulation, and sacrificing the model's ability to generalize, making it less suitable for real-world applications (Chen et al., 2023c). (2) **Initial Task Data Dependency**: General language models are pre-trained on large-scale natural language corpora. Previous methods often depend on substantial labeled data for the initial task and supervised fine-tuning to ensure proper initialization of the model backbone, enabling adaptation to complex and dynamic semantic parsing task streams. Consequently, CSP performance heavily relies on this initial labeled data, which conflicts with the few-shot challenge (Chen et al., 2023b,c).

Inspired by the Mixture-of-Experts (MoE) theory (Jacobs et al., 1991), we propose a novel CSP framework, named APEX. The framework first builds a dynamically expanding shared expert pool and trains PET experts with domain-specific knowledge for each task. To enable the model to actively select the appropriate expert at the instance level, we assign a meta-representation as an index to each expert and propose a meta-learning approach for CSP to model the matching relationship between experts and instances. During inference, we propose a routing strategy based on the historical visibility of samples to facilitate expert collaboration. This approach effectively prevents catastrophic forgetting and improves generalization without task identifiers. Moreover, to reduce the dependency on initial task data, we propose an SQL syntax-guided adaptive warm-up method. It infers the SQL syntax features of current and future tasks through induction and evolution and then guides LLMs to synthesize pseudo sample pairs. Accuracy and fidelity are enhanced through correction and filtering strategies. Extensive experimental results demonstrate that our method achieves superior performance without replaying historical data, ideal settings, or manual data collection for warm-up, outperforming previous rehearsal-based methods by up to 15.8%. Compared to methods with ideal settings, our approach not only adapts well to cold-start scenarios but also exhibits strong forward transfer capabilities, generalizing to unseen tasks and even exceeding the upper performance bound by as much as 8.8%. The main contributions of this work are summarized as follows:

- We propose a novel CSP framework, APEX, which accumulates knowledge via a dynamically expanding expert pool, without replaying historical data, and supports adaptive warm-up without manual intervention.
- We introduce a multi-expert meta-learning method and an active selection/fusion strategy based on task visibility to guide expert collaboration for different instances in CSP.
- Experimental results show that our method surpasses baselines, exceeding performance limits, and handles cold-start scenarios without relying on data replay or ideal settings.

2 Related Work

2.1 Continual Semantic Parsing

Previous semantic parsing research primarily focuses on fine-tuning Pre-trained Language Models (PLMs) on large static datasets (Cai et al., 2022; Li et al., 2023a), but high-quality SQL annotations often require domain experts, making it costly. Recently, with the rapid advancement of LLMs (Anil et al., 2023; OpenAI et al., 2024), few-shot in-context learning has made significant progress (Li et al., 2023b; Mao et al., 2024). However, these methods often rely on powerful closed-source LLMs, posing risks of data privacy breaches and high inference costs (Xue et al., 2024; Li et al., 2024; Zhang et al., 2024b). Furthermore, they are still designed for static data, making it difficult to update and iterate models as environments change. Inspired by human knowledge accumulation and adaptive learning paradigms, continuous semantic parsing based on few-shot and dynamic data has become a promising direction (Yadav et al., 2023; Chen et al., 2023b,c). Earlier related works focus on *rehearsal-based* methods (Li et al., 2021; Chen et al., 2023b), which prevent forgetting by replaying real samples from past tasks during current

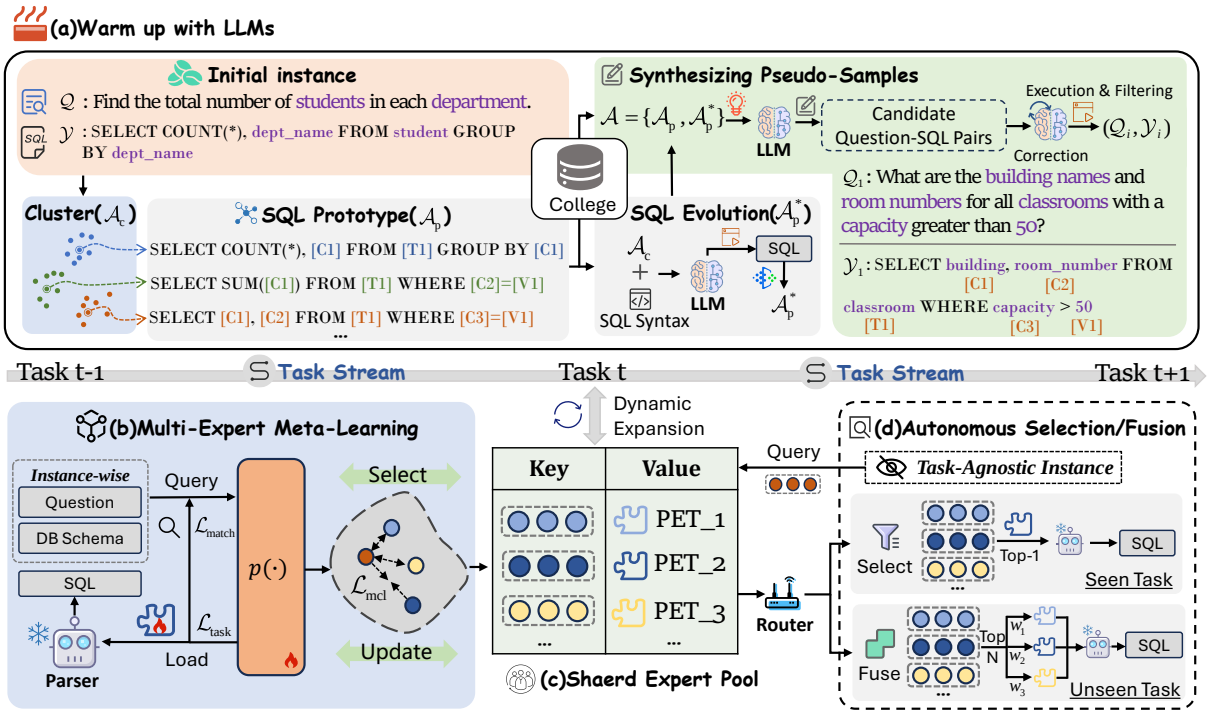

Figure 2: The overall architecture of APEX. [T], [C], and [V] represent table names, column names, and database values, respectively. For more details, see Section 4.1.

training. While simple and effective, these methods depend on the number of replayed samples and are unsuitable for scenarios with data privacy concerns or limited storage resources. Recently, *PET-based* methods that do not rely on replay have gained significant attention (Smith et al., 2023; Liu et al., 2024a; Zhao et al., 2024).

2.2 Parameter-Efficient Continual Learning

Techniques like prompt tuning (Lester et al., 2021), Adapter (Poth et al., 2023), and LoRA (Hu et al., 2022) in PET enable LLMs to adapt to low-resource downstream tasks by efficiently updating knowledge without large-scale parameter updates (Liao et al., 2024). These methods help mitigate distribution shifts and prevent catastrophic forgetting in continual learning without relying on historical data replay (Chen et al., 2023c; Zhao et al., 2024). Typically, PET modules are trained separately for each task to decouple task-specific knowledge. During inference, two settings are used: *Task-aware*, where the task identity of each sample is known, allowing the appropriate PET module to be loaded (Chen et al., 2023c). This ideal setting simplifies the continual learning scenario but struggles to generalize to unseen samples, making it difficult to apply in real-world practice. A more challenging and realistic scenario is *Task-agnostic*, where the model needs to autonomously organize PET modules (Wang et al., 2022; Razdaibiedina et al., 2023; Zhao et al., 2024; Menabue et al., 2024).

3 Problem Definition

The goal of semantic parsing is to generate an SQL query $\mathcal{Y} = F(\mathcal{Q}, \mathcal{D})$ from a natural language question $\mathcal{Q}$ and a database $\mathcal{D} = (\mathcal{L}, \mathcal{C})$, where $\mathcal{L}$ represents the set of table names, $\mathcal{C}$ is the set of corresponding column names, and F is the parser. CSP aims to learn from a sequence of tasks $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{N_T}\}$, where N_T denotes the number of tasks. Each task involves an independent semantic parsing task $\mathcal{T}_t = \{(\mathcal{X}_t^i, \mathcal{Y}_t^i)\}_{i=1}^{N_t}$, with $\mathcal{X} = (\mathcal{Q}, \mathcal{D})$, and N_t is the data size. The database information between tasks is non-overlapping. At time step t, the parser F is trained only on the data from the current task $\mathcal{T}_t$, without access to previous tasks' data. After training, the model is expected to perform well on all seen tasks and generalize effectively to unseen future tasks.

4 Methodology

4.1 Overview

The framework of APEX is shown in Figure 2, consisting of three stages: ①***Warm-up preparation*** (Figure 2(a)): To reduce reliance on large manu-

ally labeled data (Chen et al., 2023b,c), we propose a SQL syntax-guided adaptive warm-up method using LLMs. SQL prototypes are extracted from existing samples and evolved for potential future tasks, guiding LLMs to generate question-SQL pairs with accuracy and fidelity ensured through correction and filtering. ②***Training*** (Figure 2(b)): A dynamically expanding expert pool is first established (Figure 2(c)). We then propose a multi-expert meta-learning approach to capture instance-expert commonalities and differences, enabling the most suitable expert collaboration for each sample. ③***Prediction*** (Figure 2(d)): For task-agnostic samples, we infer task visibility and apply different decoding strategies. For seen tasks, the most relevant expert is selected, for unseen tasks, multiple experts are fused for collaborative inference.

4.2 Adaptive Warm-Up with LLMs

To enhance the model's initial semantic parsing performance, previous methods rely on large amounts of manually labeled data for supervised fine-tuning. However, in real-world settings, acquiring sufficient labeled data at the start of continual learning is challenging (Chen et al., 2023b). To overcome this issue, inspired by related work (Liu et al., 2024b), we propose a SQL syntax-guided adaptive warm-up method leveraging LLMs' generation and reasoning abilities, as shown in Figure 2(a). All prompt details are provided in Appendix A.

SQL Prototype Extraction The model's understanding of SQL syntax reflects its semantic parsing ability. While databases vary across CSP tasks, they share the same SQL syntax. Early in continual learning, limited labeled data hinders the model's grasp of diverse SQL. Thus, we extract SQL syntax features from two perspectives: reinforcing current knowledge and anticipating future syntax. Considering SQL's code-like properties, we use CodeT5 (Wang et al., 2021) as the encoder to jointly represent the question $\mathcal{Q}$, schemas $\mathcal{D}$, and SQL $\mathcal{Y}$ from existing samples, and then apply k-means clustering to decouple the complexity of semantic parsing. This allows each cluster to share similar sub-SQL knowledge. Next, database-specific information is removed from the SQL of cluster centers $\mathcal{A}_c$, to obtain syntax-focused SQL prototypes $\mathcal{A}_p$. However, limiting the model to existing syntax may restrict its scope. To address this, we use the extracted prototypes as seeds, combining them with common SQL keywords, and leverage LLMs to infer potential future SQL prototypes to obtain $\mathcal{A}_p^*$.

Pseudo-Sample Synthesis To tackle the scarcity of labeled data, previous works often rely on semi-supervised learning (Qin and Joty, 2022), such as self-training (Chen et al., 2023b), which requires extra data and yields low-quality labels due to its one-way generation. We propose a SQL syntax-guided soft synthesis method using LLMs, where LLMs are provided with database schemas and SQL prototypes $\mathcal{A} = \{\mathcal{A}_p, \mathcal{A}_p^*\}$ to generate natural language questions $\hat{\mathcal{Q}}$ and corresponding SQL $\hat{\mathcal{Y}}$ simultaneously. This bidirectional generation taps into LLMs' internal knowledge, avoiding one-way limitations and removing the need for additional semi-supervised data.

Correction and Filtering Although our method mitigates some challenges in pseudo-sample synthesis, LLMs' hallucination issues (Qu et al., 2024) can still produce SQL queries with execution errors or deviations from the intended syntax. To enhance accuracy and fidelity, inspired by prior code-related works (Huang et al., 2024; Chen et al., 2023a), we design a two-stage correction-filtering strategy. First, the LLM performs self-correction, retaining only executable samples. Then, we mask the database schema and select the Top-N_g samples with the smallest edit distance to the target syntax.

4.3 Multi-Expert Meta-Learning

After the warm-up phase, our method splits into two parts: multi-expert meta-learning and active selection or fusion, corresponding to the training and prediction stages in CSP, as shown in Figure 2(b)-(d). During training, the matching relationship between the meta-representations of different PET experts and instances is modeled and optimized. In inference, dynamic recognition of a sample's visibility guides the decoding strategy, mitigating forgetting and enhancing generalization.

Dynamic Expert Pool Expansion Unlike previous work that assigns an Oracle expert module to test samples (Chen et al., 2023c), this work focuses on real-world continual learning scenarios where the task identifier is unknown. Inspired by prior work (Wang et al., 2022), an expert pool is maintained to encourage the model to actively invite experts to collaborate, as shown in Figure 2(c). We observe that previous methods use a large, fixed expert pool, which makes early expert collaboration challenging (Yadav et al., 2023). To address

this, we propose a dynamically expanding expert pool that grows with tasks. For each task $\mathcal{T}_t$, we similarly use k-means to cluster the samples into K groups, train a PET module for each cluster as its expert, and assign a learnable meta-representation as a key. The expert pool then expands to include all key-expert paris up to task $\mathcal{T}_t$, forming the pool $\mathcal{E}_t$, represented as:

$$\mathcal{E}_t = \{(\mathbf{k}_1, P_1), (\mathbf{k}_2, P_2), \ldots, (\mathbf{k}_{N_p^t}, P_{N_p^t})\} \quad (1)$$

Here, $\mathbf{k} \in \mathbb{R}^{1\times d}$ represents different keys, initialized from the hidden state representation of each cluster center x_c, i.e., $\mathbf{k}^{\text{init}} = F_e(x_c)$. d is the encoding dimension. P_i represents the i-th PET expert module, with $i \in [1, N_p^t]$, and N_p^t denotes the total number of experts in the pool at task $\mathcal{T}_t$.

During training, the model actively queries key-value pairs to select the appropriate expert for each sample. We first use the parser's frozen encoder as a function F_e to extract query features of the sample: $q(x) = F_e(x)$, then apply a scoring function $\Phi(\cdot)$ to find the most relevant key $\mathbf{k}^*$ in $\mathcal{E}_t$:

$$\mathbf{k}^* = \arg\max_{\mathbf{k}_i \in \mathcal{E}_t} \Phi(q(x), \mathbf{k}_i) \quad (2)$$

Where $\Phi(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$ represents the similarity between $\mathbf{u}$ and $\mathbf{v}$. This design eliminates the need for real historical data during training or Oracle task identifiers during prediction.

Meta-Continual Learning In each training step t, the most relevant expert module is selected based on the above strategy, loaded into the frozen backbone model for training, and the match loss between the query vector and the meta-representation is then computed as follows:

$$\mathcal{L}_{\text{match}} = -\sum_{x_t \in \mathcal{T}_t} \Phi(q(x_t), \mathbf{k}^*) \quad (3)$$

Although the above calculation helps the model select expert modules by learning keys at the instance level, this approach treats different keys independently (Wang et al., 2022), overlooking the relationships between them. As a result, the model struggles to capture the similarities and differences among various experts effectively. To tackle this, we propose a Meta-Contrastive Loss (MCL) in CSP, aligning the current task with the relevant key while enhancing differentiation from other tasks, and improving expert distinguishability during inference.

For the sample x_t, we treat its query feature $q(x_t)$ and the corresponding task's key $\mathbf{k}^+$ as a positive pair, while $q(x_t)$ and keys from other tasks $\mathbf{k}^- \in \mathcal{K}^-$ are negative pairs. We then apply a non-linear projection $p(\cdot)$ to map the query feature into the latent space and compute the following:

$$\mathbf{h}_t = p(q(x_t)) = \mathbf{W}_2 \text{ReLU}(\mathbf{W}_1 q(x_t)) \quad (4)$$

Here, $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable weight matrices. Next, we calculate the meta-contrastive loss:

$$\mathcal{L}_{\text{mcl}} = -\sum_{t}^{|\mathcal{T}_t|} \log \frac{\exp(\Phi_t^+/\tau)}{\exp(\Phi_t^+/\tau) + \sum_{\mathbf{k}^- \in \mathcal{K}^-} \exp(\Phi_t^-/\tau)} \quad (5)$$

Where $\Phi_t^+ = \Phi(\mathbf{h}_t, \mathbf{k}^+)$, and similarly for Φ_t^-. τ is the temperature factor. We define the training loss of PET parameters θ_{pet} as: $\mathcal{L}_{\text{task}} = -\sum_{(x_t, y_t)\in\mathcal{T}_t}^{\mathcal{T}_t} \log P(y_t|x_t; \theta_{\text{pet}})$. Thus, the total loss with weight factors α and β is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \alpha\mathcal{L}_{\text{match}} + \beta\mathcal{L}_{\text{mcl}} \quad (6)$$

Active Selection or Fusion During prediction, samples are classified as seen (from previous tasks) or unseen (from new tasks). For seen samples, the expert module from their original task is loaded, while for unseen samples, relevant expert modules are dynamically selected and combined based on SQL skills. We propose a simple but effective routing mechanism to predict sample historical visibility and choose the appropriate inference strategy. It calculates the similarity between the test sample and keys of all PET modules in the expert pool, using the z-score (standard score) (Abdi, 2007) to measure visibility. A high z-score indicates a significant deviation from the mean, suggesting a strong association with a specific expert, while a low z-score indicates a weak association with the current expert. The calculation is as follows:

$$Z_i = \frac{\Phi(q(x), \mathbf{k}_i) - \mu}{\sigma} \quad (7)$$

In this case, Z_i represents the z-score of sample x with the i-th expert module, where $i \in [1, N_p]$. μ is the mean of the similarity scores with all expert modules, and σ is the standard deviation. Finally, we select the largest Z_i among all experts as the final target: $Z = \max\left(Z_1, Z_2, \ldots, Z_{N_p}\right)$.

We also introduce a threshold γ to infer sample historical visibility by comparing it with Z, which guides the inference strategy. For seen tasks, the

Top-1 expert module is loaded. For unseen tasks, the Top-N experts are selected, and their scores are normalized to serve as weight proportions for fusing the parameters of these modules. The expert key set $\hat{\mathcal{K}} \in \mathcal{E}$ selection process is as follows:

$$\hat{\mathcal{K}} = \begin{cases} \arg\max \Phi(q(x), \mathbf{k}_i) & \text{if } Z \geq \gamma \\ \text{Top-}N(\Phi(q(x), \mathbf{k}_i)) & \text{else} \end{cases} \quad (8)$$

The calculation process of expert module parameter selection or fusion is as follows:

$$\hat{P} = \begin{cases} V(\hat{\mathcal{K}}) & \text{if } Z \geq \gamma \\ \sum_{i=1}^{N} w_i V(\hat{\mathcal{K}}) & \text{else} \end{cases} \quad (9)$$

Where $w_i = \frac{e^{\Phi(q(x), \mathbf{k}_i)}}{\sum_{j=1}^{N} e^{\Phi(q(x), \mathbf{k}_j)}}$. $V(\cdot)$ maps a key to expert module parameters.

5 Experiments

5.1 Datasets

We evaluate our method on two benchmarks: **Spider-stream** and **Combined-stream** (Chen et al., 2023c). Spider-stream, built from the Spider (Yu et al., 2018), has most tasks with fewer than 500 labeled samples. It assesses the impact of domain shifts in databases on CSP under limited resources. Combined-stream, constructed from Spider and WikiSQL (Zhong et al., 2018), introduces single-table queries from WikiSQL, evaluating performance across both domain shifts and SQL complexity. Notably, in the original setting, large amounts of labeled data are manually collected for the initial task to warm up the model. To better reflect real-world scenarios, we introduce a cold-start setting where the first task does not have the most labeled data, and tasks are shuffled randomly to evaluate the effects of initial performance and task order. More details are in Appendix B.1.

5.2 Evaluation Metrics

Following previous works (Yu et al., 2018; Chen et al., 2023b,c), we adopt four common continual learning metrics and report the official semantic parsing metrics: Exact-set-Match (EM) and EXecution (EX) accuracy. EM checks the correctness of the generated SQL, while EX verifies its execution on the database. (1) **Average Accuracy** (ACC_a): Average performance across all seen tasks. (2) **Whole Accuracy** (ACC_w): Performance on the combined test set of all seen tasks. (3) **Backward Transfer** (BWT): Impact on previous tasks after learning the last task. (4) **Forward Transfer** (FWT): Ability to generalize to unseen tasks. Notably, the previous method (Chen et al., 2023c) fails to generalize to new tasks due to its ideal settings. Metric calculations are detailed in Appendix B.2.

5.3 Implementation Details

Our method is independent of PLMs and PET structures, allowing seamless integration with Transformer-based LLMs. To ensure a fair comparison with prior works, we use the T5 series as the backbone model. For PET, we choose the popular Adapter (Poth et al., 2023) and LoRA (Hu et al., 2022), which are widely used for supervised fine-tuning of PLMs. During the warm-up phase, we utilize the publicly available Mixtral-8x7B-Instruct-v0.1, and generate 10 pairs of pseudo-samples per SQL prototype. The expert pool is expanded with $K = 5$ clusters per task. In training, the temperature factor τ is 0.07, and loss weights α and β are 0.3 and 0.1. The inference threshold γ is set to 1.6, and the Top-N for fusion is 3. All experiments are run on an A100 GPU with 80GB memory. Further details are provided in Appendix B.3.

5.4 Baselines

We conduct a comprehensive comparison with the following three types of baselines: (1) *Original method*: Sequential Fine-Tuning (**SFT**) (Yogatama et al., 2019) reflects the performance of a vanilla model. (2) *Rehearsal-based methods*: **EMAR** (Han et al., 2020), **SFNET** (Chen et al., 2023b). They require replaying real historical examples and using additional unlabeled data. (3) *PET-based methods*:**Fusion** (Pfeiffer et al., 2021), **PEFT** (Chen et al., 2023c), **C3** (Chen et al., 2023c), **ProgPrompt** (Razdaibiedina et al., 2023), **L2P** (Wang et al., 2022), **Soup** (Chronopoulou et al., 2023). Some of them rely on ideal settings. As our method uses LLMs to generate pseudo-samples (zero-shot) during the warm-up, we also report its standalone zero-shot performance. The **ORACLE** setting incrementally trains the model on all task data, representing the upper bound of CSP. More details are in Appendix B.4.

6 Results and Analysis

6.1 Overall Results

Table 1 shows the overall results of different methods. Our method demonstrates clear advantages across benchmarks, even exceeding theoretical per-

Method	DR	PET	Spider-stream					Combined-stream				
			ACC_a	ACC_w	BWT	FWT	❄ACC_w	ACC_a	ACC_w	BWT	FWT	❄ACC_w
T5-Base(220M)												
SFT	✗	Full	51.9/57.9	47.1/52.8	-19.6/-14.9	**29.6/33.2**	45.7	51.0/36.0	52.4/38.3	-11.1/-17.6	19.9/11.1	32.0
	✗	Adapter	40.8/43.5	38.9/41.7	-32.3/-31.3	23.9/27.4	35.6	38.8/33.9	43.0/40.9	-33.6/-37.8	11.4/5.4	18.5
	✗	LoRA	42.3/47.8	42.2/47.6	-28.4/-24.1	27.2/30.7	32.6	46.9/40.6	48.6/45.1	-21.0/-26.3	16.8/11.1	17.0
L2P	✗	Adapter	22.6/24.3	22.4/24.2	-42.2/-41.6	15.4/17.2	21.4	27.0/25.7	26.8/25.6	-38.1/-35.0	8.7/2.4	12.3
	✗	LoRA	30.1/32.7	34.8/38.2	-33.2/-31.2	23.0/25.9	26.0	27.1/22.0	26.0/23.2	-31.0/-35.8	8.2/3.7	16.7
Fusion	✗	Adapter	41.1/45.2	39.7/43.6	-32.4/-29.7	24.5/27.4	23.9	35.0/29.8	43.2/40.7	-33.8/-37.4	10.2/5.1	13.5
Soup	✗	Adapter	31.4/34.3	43.1/46.0	-1.7/-1.7	27.5/30.4	8.1	25.5/22.7	40.2/38.9	-5.2/-3.7	19.0/15.7	13.7
ProgPrompt	✗	Prompt	21.0/23.6	23.5/27.1	-40.1/-39.7	16.3/16.9	7.1	21.3/19.2	27.2/26.8	-43.9/-41.2	6.1/3.3	0.1
EMAR	✓	Full	55.6/59.4	50.6/55.1	-17.5/-15.0	28.5/31.1	53.2	61.2/59.4	57.6/57.9	-7.3/-6.8	28.4/25.9	50.1
SFNET♣	✗	Full	51.8/57.8	47.1/52.2	-18.9/-13.8	28.3/32.7	46.4	56.2/49.4	51.8/48.4	-7.9/-14.4	21.0/14.6	24.4
SFNET♣	✓	Full	60.4/62.5	54.6/57.4	-11.3/-10.7	28.8/31.8	51.3	61.3/60.9	57.5/56.7	-5.4/-5.2	**29.7/27.5**	55.9
APEX(Ours)	✗	Adapter	61.0/62.4	**63.4/64.8**	**1.3/1.4**	25.1/28.3	53.5	62.7/59.3	**64.9**/63.1	6.8/9.2	21.9/20.7	**57.7**
	✗	LoRA	**63.3/65.3**	62.1/63.9	0.4/0.5	24.4/27.5	**57.8**	**63.5/61.4**	64.3/**63.5**	**7.2/10.6**	29.2/25.8	56.1
Ideal Setting & Upper Bound												
PEFT♢	✗	Prompt	65.7/-	64.5/-	0.0/0.0	N/A	14.9	63.8/-	66.2/-	0.0/0.0	N/A	16.6
	✗	Adapter	69.9/71.1	70.7/72.0	0.0/0.0	N/A	70.1	66.7/65.2	67.6/67.5	0.0/0.0	N/A	64.3
	✗	LoRA	68.0/69.8	69.6/71.4	0.0/0.0	N/A	64.7	63.1/61.4	67.0/66.7	0.0/0.0	N/A	58.4
ORACLE	-	Full	75.6/77.0	76.4/78.4	4.2/3.3	29.7/32.0	-	70.2/68.2	71.1/70.8	5.6/4.7	29.2/26.6	-
T5-Large(770M)												
SFT	✗	Full	60.6/67.3	56.2/62.0	-20.0/-14.6	**39.7/44.4**	56.9	57.8/48.9	58.9/53.3	-7.6/-16.4	28.3/25.4	44.3
	✗	Adapter	47.1/51.8	48.0/52.8	-30.5/-27.3	34.3/37.8	44.2	47.3/39.6	52.0/47.5	-28.6/-35.0	16.4/11.0	22.6
	✗	LoRA	52.4/56.5	52.3/56.4	-25.3/-21.9	37.6/41.8	43.8	47.7/41.2	46.3/42.6	-28.7/-35.6	26.5/21.2	18.5
ProgPrompt	✗	Prompt	27.9/29.6	32.8/35.3	-37.0/-36.5	17.2/17.2	11.5	23.5/20.2	29.8/27.7	-46.4/-42.7	10.2/3.7	1.4
EMAR	✓	Full	63.9/68.0	59.2/62.7	-13.7/-12.0	36.5/39.5	59.0	58.1/55.8	55.2/54.3	-12.4/-12.8	28.1/22.1	53.0
SFNET♣	✗	Full	54.1/60.2	50.3/55.6	-21.0/-17.6	37.3/41.1	51.5	65.0/60.3	61.5/59.7	-7.5/-11.9	28.2/20.4	32.6
SFNET♣	✓	Full	63.1/66.6	56.5/59.9	-10.5/-8.6	34.3/38.7	58.6	66.2/**65.2**	61.4/61.3	-3.9/-3.2	**37.1/35.0**	59.8
APEX(Ours)	✗	Adapter	67.0/68.8	68.3/70.3	**0.0/4.4**	29.2/32.3	60.3	64.3/60.5	66.0/63.2	6.8/9.3	27.1/23.1	**62.9**
	✗	LoRA	**69.2/70.9**	**70.8/72.6**	-0.4/-0.4	33.3/37.2	**61.4**	**66.3**/63.0	**68.3/65.2**	**8.4/11.1**	32.8/30.3	58.5
Ideal Setting & LLMs Performance & Upper Bound												
PEFT♢	✗	Prompt	69.8/-	67.4/-	0.0/0.0	N/A	51.7	67.3/-	70.0/-	0.0/0.0	N/A	38.2
	✗	Adapter	73.3/75.5	73.8/76.0	0.0/0.0	N/A	74.7	70.4/68.1	71.3/70.4	0.0/0.0	N/A	70.1
	✗	LoRA	75.7/76.8	76.1/77.5	0.0/0.0	N/A	72.3	70.6/70.3	71.8/72.6	0.0/0.0	N/A	66.7
C3♢	✗	Prompt	70.7/-	68.9/-	0.0/0.0	N/A	-	69.0/-	71.2/-	0.0/0.0	N/A	-
C3♢†	✗	Prompt	71.3/-	69.6/-	0.0/0.0	N/A	-	67.6/-	70.0/-	0.0/0.0	N/A	-
Mixtral-8x7B	✗	-	23.9/29.7	23.1/28.5	-/-	-/-	-	28.6/24.4	27.0/26.0	-/-	-/-	-
ORACLE	-	Full	80.9/82.8	80.5/82.6	2.4/2.9	38.1/41.3	-	73.7/73.2	75.8/76.0	1.9/2.3	37.3/34.2	-

Table 1: Results (EM/EX) on Spider-stream and Combined-stream datasets (%). APEX uses Mixtral-8x7B for warm-up. DR indicates whether historical data replay is used, with a memory size set to 15. PET refers to different training architectures, with FULL being full-parameter tuning. ❄ACC_w represents the ACC_w-EM in the cold start scenario, detailed results can be found in Appendix C.2. ♣ indicates the need for additional unsupervised data, and ♢ means a known task ID, but forward transfer is impossible (N/A). Some PEFT and C3 results are from their original papers or official code repositories. † 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.

formance limits on some metrics. Notably, it requires no historical data replay, additional unsupervised data, or idealized settings. More results and analysis are in Appendix C.

VS. Rehearsal-Based Methods These baselines rely on full parameter fine-tuning, data replay, and additional data, differing from ours and making comparisons less fair. Meanwhile, SFT results show that simply adding PET modules reduces overhead but lowers performance. Despite this, APEX achieves significant improvements without relying on data strategies, outperforming the rehearsal-based SFNET with a BWT increase of up to 15.8% on the Combined-stream. Notably, SFNET's performance drops below SFT when replay strategies are removed, underscoring its heavy reliance on replay, whereas APEX remains unaffected. More results are in Appendix C.1.

VS. PET-Based Methods They can be divided into with and without ideal settings. Baselines without ideal settings perform poorly due to task-level PET module combinations lacking fine-grained guidance. L2P, despite instance-level selection, is limited by a fixed prompt pool and single selection strategy, hindering knowledge accumulation and ignoring sample visibility (Yadav et al., 2023). Baselines relying on ideal settings use task identifiers, restricting knowledge sharing and making them unsuitable for new tasks. In contrast, APEX excels without ideal settings by flexibly accumulating knowledge and organizing expert collaboration. It mitigates forgetting, enhances general-

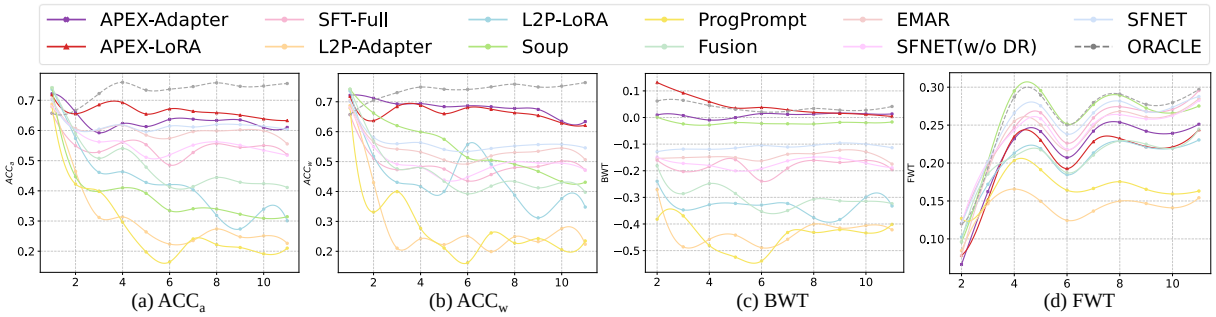

Figure 3: Comparison of results (EM) across all seen tasks on Spider-stream (T5-base).

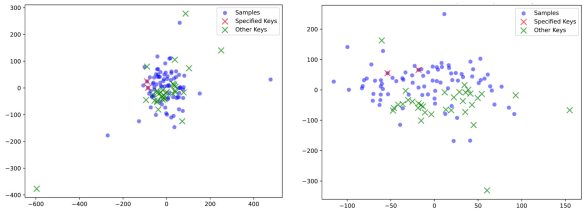

Figure 4: t-SNE visualization results without MCL (left) and with MCL (right). ● represent samples from the same task. ✗ denotes the specific key related to the current task, and ✗ indicates unrelated keys.

ization, and ensures efficient parameter utilization. APEX outperforms PET-based methods across most metrics, surpassing ORACLE in BWT by 8.8% on the Combined-stream, proving its effectiveness.

Cold Start Results Table 1 also shows the performance of different methods in the cold-start scenario, where most methods experience a performance drop, highlighting the challenges of this scenario. In particular, PET-based methods suffer a noticeable decline, emphasizing their reliance on manually collected warm-up data. ProgPrompt struggles on the Combined-stream dataset due to increased task diversity and its progressive prompt design, which limits input length. In contrast, our method uses LLMs for adaptive warm-up, adapts well to cold-start conditions, and performs best. See Appendix C.2 for more detailed results.

6.2 Detailed Results and Analysis

Results Till the Seen Tasks Figure 3 shows that our method maintains stable and significant advantages in all tasks, even surpassing ORACLE. Most PET-based methods suffer severe forgetting as tasks increase, while rehearsal-based methods perform slightly better. APEX with LoRA significantly boosts generalization, consistently surpassing ORACLE. More details are in Appendix C.3.

Impact of Warm-Up and MCL Table 2 shows that our LLM-based warm-up method significantly boosts both forgetting prevention and generalization, with gains up to 9.3%. Additionally, MCL consistently improves model performance. Further results in Figure 4 demonstrate that MCL effectively pulls samples closer to relevant keys and distances them from irrelevant ones, helping to predict the sample's historical visibility.

	ACC$_a$	ACC$_w$	BWT	FWT
APEX	62.7/59.3	64.9/63.1	6.8/9.2	21.9/20.7
w/o Warm up	57.7/50.1	60.5/57.4	-0.5/-0.1	15.8/14.2
w/o MCL	60.6/57.0	61.9/59.7	8.0/9.2	20.2/17.1
S/F(γ=1.4)	62.0/61.2	65.5/63.7	7.7/10.2	17.0/13.6
S/F(γ=1.5)	63.4/59.5	64.5/63.3	8.7/11.0	19.4/16.4
S/F(γ=1.7)	60.8/57.4	61.9/60.3	5.2/6.8	23.9/21.2
S/F(Top-2)	62.9/59.6	65.3/63.5	4.4/5.3	24.0/21.7
S/F(Top-4)	62.7/59.3	64.9/63.0	8.7/10.6	21.3/20.0
only S	60.6/57.2	60.8/59.4	-1.0/-1.0	18.2/15.7
only F(Top-2)	45.8/38.8	47.8/42.2	1.4/0.8	29.9/23.8
only F(Top-3)	33.6/26.9	39.9/34.3	0.1/0.8	23.5/18.1
only F(Top-4)	29.3/23.2	36.3/30.4	0.4/0.6	20.7/15.7

Table 2: Ablation study on Combined-stream (Adapter, T5-base). S/F indicates prediction using Select/Fuse.

Impact of Selection/Fusion We experiment with different Top-N, γ thresholds, and inference strategies, including selecting only the top expert (treating all samples as seen) and only fusing experts (treating all samples as unseen). γ is crucial in balancing the model's forgetting and generalization. From Table 2, we observe that the lower γ, the more likely a sample is to be classified as seen, improving forgetting-related metrics (ACC$_a$, ACC$_w$, BWT) but lowering generalization (FWT). Conversely, increasing γ enhances generalization but increases forgetting. Thus, we search for an optimal γ to balance forgetting and generalization. Using only selection or fusion strategies results in a significant overall performance drop, although the fusion-only strategy can improve the model's gen-

eralization. The number of Top-N experts in the fusion phase also impacts performance. In APEX, varying the number of experts consistently yields good results, while in the fusion-only strategy, increasing experts degrades performance.

7 Conclusion

This work presents a novel parameter-efficient approach for continual semantic parsing. It uses LLMs for adaptive warm-up without manual intervention and builds a dynamically expanding expert pool to accumulate task-specific knowledge. Through meta-continual learning and a routing module, the model actively selects appropriate experts for samples with unknown task identifiers. Importantly, it requires no real historical data or ideal settings, while delivering strong performance, making it well-suited for real-world applications.

Limitations

This study has several limitations that warrant further exploration in future work: (1) Optimization of the expert pool: although our PET-based method effectively reduces training costs, the growing number of tasks in the task stream leads to an accumulation of PET modules in the expert pool. This results in knowledge redundancy and increased storage overhead. Streamlining the expert pool through pruning and refinement will be essential for managing longer task streams in practical applications. (2) Balancing past and future: our research shows that combining sample historical visibility with flexible selection/fusion inference strategies can significantly improve model performance. However, this improvement largely depends on the accuracy of predicting historical visibility. Exploring better approaches to distinguish between past and future tasks remains a promising direction. (3) Scaling to more LLMs and tasks: while this work primarily focuses on CSP tasks, our approach can also be applied to continual learning in other domains, such as code generation and complex reasoning. Moreover, it has the potential to integrate with more open-source LLMs and PET technologies, broadening its applicability.

Ethics Statement

In this paper, we propose a novel continual semantic parsing framework called APEX. We conduct experiments on two publicly available benchmark datasets, Spider-stream and Combined-stream, ensuring no ethical concerns.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work is supported by the National Natural Science Foundation of China (No. 62476066 and No. 62277002).

References

Hervé Abdi. 2007. Z-scores. *Encyclopedia of measurement and statistics*, 3:1055–1058.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, and Eric Chu et al. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

Zefeng Cai, Xiangyu Li, Binyuan Hui, Min Yang, Bowen Li, Binhua Li, Zheng Cao, Weijie Li, Fei Huang, Luo Si, and Yongbin Li. 2022. STAR: SQL guided pre-training for context-dependent text-to-SQL parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1235–1247, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. 2023a. Codet: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations*.

Yongrui Chen, Xinnan Guo, Tongtong Wu, Guilin Qi, Yang Li, and Yang Dong. 2023b. 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.

Yongrui Chen, Shenyu Zhang, Guilin Qi, and Xinnan Guo. 2023c. Parameterizing context: Unleashing the power of parameter-efficient fine-tuning and in-context tuning for continual table semantic parsing. In *Advances in Neural Information Processing Systems*, volume 36, pages 17795–17810. Curran Associates, Inc.

Alexandra Chronopoulou, Matthew Peters, Alexander Fraser, and Jesse Dodge. 2023. AdapterSoup: Weight averaging to improve generalization of pretrained language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2054–2063, Dubrovnik, Croatia. Association for Computational Linguistics.

Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6429–6440, Online. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Baizhou Huang, Shuai Lu, Xiaojun Wan, and Nan Duan. 2024. Enhancing large language models in coding through multi-perspective self-consistency. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1429–1450, Bangkok, Thailand. Association for Computational Linguistics.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 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.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. Codes: Towards building open-source language models for text-to-sql. *Proc. ACM Manag. Data*, 2(3).

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Ma Chenhao, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems*, volume 36, pages 42330–42357. Curran Associates, Inc.

Zhuang Li, Lizhen Qu, and Gholamreza Haffari. 2021. Total recall: a customized continual learning method for neural semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3816–3831, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Huanxuan Liao, Shizhu He, Yao Xu, Yuanzhe Zhang, Yanchao Hao, Shengping Liu, Kang Liu, and Jun Zhao. 2024. From instance training to instruction learning: Task adapters generation from instructions. *Preprint*, arXiv:2406.12382.

Jialin Liu, Jianhua Wu, Jie Liu, and Yutai Duan. 2024a. Learning attentional mixture of loras for language model continual learning. *Preprint*, arXiv:2409.19611.

Ruiheng Liu, Jinyu Zhang, Yanqi Song, Yu Zhang, and Bailong Yang. 2024b. Filling memory gaps: Enhancing continual semantic parsing via sql syntax variance-guided llms without real data replay. *Preprint*, arXiv:2412.07246.

Wenxin Mao, Ruiqi Wang, Jiyu Guo, Jichuan Zeng, Cuiyun Gao, Peiyi Han, and Chuanyi Liu. 2024. Enhancing text-to-SQL parsing through question rewriting and execution-guided refinement. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2009–2024, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Martin Menabue, Emanuele Frascaroli, Matteo Boschini, Enver Sangineto, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. 2024. Semantic residual prompts for continual learning. In *Computer Vision – ECCV 2024*, pages 1–18, Cham. Springer Nature Switzerland.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and et al. Shyamal Anadkat. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan

Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. 2023. Adapters: A unified library for parameter-efficient and modular transfer learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 149–160, Singapore. Association for Computational Linguistics.

Chengwei Qin and Shafiq Joty. 2022. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *International Conference on Learning Representations*.

Ge Qu, Jinyang Li, Bowen Li, Bowen Qin, Nan Huo, Chenhao Ma, and Reynold Cheng. 2024. Before generation, align it! a novel and effective strategy for mitigating hallucinations in text-to-SQL generation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 5456–5471, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. 2023. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11909–11919.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A comprehensive survey of continual learning: Theory, method and application. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(8):5362–5383.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 139–149.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Siqiao Xue, Caigao Jiang, Wenhui Shi, Fangyin Cheng, Keting Chen, Hongjun Yang, Zhiping Zhang, Jianshan He, Hongyang Zhang, Ganglin Wei, Wang Zhao, Fan Zhou, Danrui Qi, Hong Yi, Shaodong Liu, and Faqiang Chen. 2024. Db-gpt: Empowering database interactions with private large language models. *Preprint*, arXiv:2312.17449.

Prateek Yadav, Qing Sun, Hantian Ding, Xiaopeng Li, Dejiao Zhang, Ming Tan, Parminder Bhatia, Xiaofei Ma, Ramesh Nallapati, Murali Krishna Ramanathan, Mohit Bansal, and Bing Xiang. 2023. Exploring continual learning for code generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 782–792, Toronto, Canada. Association for Computational Linguistics.

Jiaxi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024. Synthesizing text-to-SQL data from weak and strong LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7864–7875, Bangkok, Thailand. Association for Computational Linguistics.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. *Preprint*, arXiv:1901.11373.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, bailin wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, richard socher, and Caiming Xiong. 2021. Grappa: Grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Bin Zhang, Yuxiao Ye, Guoqing Du, Xiaoru Hu, Zhishuai Li, Sun Yang, Chi Harold Liu, Rui Zhao, Ziyue Li, and Hangyu Mao. 2024a. Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation. *Preprint*, arXiv:2403.02951.

Xiaokang Zhang, Jing Zhang, Zeyao Ma, Yang Li, Bohan Zhang, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, Daniel Zhang-Li, Jifan Yu, Shu Zhao,

Juanzi Li, and Jie Tang. 2024b. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *Preprint*, arXiv:2403.19318.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. SAPT: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11641–11661, Bangkok, Thailand. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Seq2SQL: Generating structured queries from natural language using reinforcement learning.

A Method Details

A.1 SQL Prototype Extraction

To prevent the model from being constrained by existing SQL syntax prototypes, we first utilize LLM to evolve and expand them, preparing for potential future tasks. The prompt flow is illustrated in Figure 5, where {database_schema} represents the SQL-formatted database schema information, {sql_keywords} are the given SQL syntax keywords, such as [SELECT, FROM, JOIN, ...]. This allows our method to generalize to semantic parsing tasks that use different types of SQL syntax. {existing_sql_queries} refers to the set of SQL queries obtained from clustering, denoted as $\mathcal{A}_c$. In practice, we first randomly sample a subset of SQL queries from $\mathcal{A}_c$ as input. Then, using the prompt shown in Figure 5, we generate N_s new SQL queries. Next, we retain only the executable SQL queries. This process is repeated N_e times, and finally, all the newly generated SQL queries are clustered and stripped of database schema information to obtain a new set of SQL prototypes $\mathcal{A}_p^*$. We set N_s to 20 and N_e to 5, and cluster centers are set to 80 for warm start and 60 for cold start.

A.2 Pseudo-Sample Synthesis

We use the existing SQL prototype $\mathcal{A}$ to guide the LLM in generating appropriate pseudo-sample pairs, with the prompt design shown in Figure 6. Specifically, the LLM is provided with an SQL prototype and database schema and instructed to generate corresponding question-SQL pairs. It is important to note that we do not directly use SQL prototypes with placeholders such as [T], [C], and [V], as we found that most open-source LLMs struggle to generate sample pairs that strictly adhere to the prototype, while some closed-source LLMs perform better. To accommodate different LLMs and reduce the complexity of the task, we follow previous work (Liu et al., 2024b) and simplify the prototype into a set of SQL keywords. To reduce noise in generation and improve fidelity to the SQL prototype, we introduce a reflection phase for sample correction, with the prompt shown in Figure 7. Finally, we replace the database schema in the generated SQL with placeholders, compute the edit distance from the SQL prototype, and select the Top-N_g most similar pseudo-sample. A total of 10 pseudo-samples are generated for each SQL prototype, and N_g is set to 1.

B Experimental Details

B.1 Datasets

Spider-stream consists of 11 tasks, while Combined-stream contains 7 tasks, with statistics for both datasets shown in Figure 8. In the cold-start scenario, the original task order IDs we used for the two datasets are [5, 2, 9, 0, 7, 4, 10, 1, 8, 3, 6] (Spider-stream) and [3, 2, 4, 0, 6, 5, 1] (Combined-stream). It is important to note that SFNET requires additional semi-supervised data, so we followed the approach from prior work (Chen et al., 2023c) by using unused data from the corresponding databases in Spider and WikiSQL as semi-supervised data. Figure 8 also shows the amount of such data, while the other methods do not require this data. During the warm-up with LLMs phase, Table 3 lists the amount of synthesized data for our method's initial tasks in Spider-stream and Combined-stream.

	Spider-stream	**Combined-stream**
Warm	3,393	3,430
Cold	1,445	1,828

Table 3: The number of pseudo samples synthesized on the initial task. *Warm* and *Cold* represent the original task order and the cold-start task order, respectively.

B.2 Evaluation Metrics

We define $a_{i,j}$, where $i, j \in [1, |\mathcal{T}|]$, as the test accuracy (including EM and EX) of the parser on task $\mathcal{T}_j$ after training on task $\mathcal{T}_i$. $|\mathcal{T}|$ denotes the total number of tasks. The four metrics are calculated as follows:

- $\text{ACC}_\text{a} = \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} a_{|\mathcal{T}|,i}$

You are provided with a database schema, a list of SQL keywords, and several sample SQL queries. Your task is to generate a diverse set of new SQL queries that reflect different structures, SQL patterns, and use cases. Ensure that each generated SQL query incorporates different SQL keywords, refers to the provided schema, and follows SQL best practices.

Database Schema: {database_schema}

SQL Keywords: {sql_keywords}

Sample SQL Queries: {sample_sql_queries $\mathcal{A}_c$}

Instructions:
-- Generate 20 new SQL queries.
-- Each query must use SQL keywords from the provided list.
-- Vary the structure and style of the queries, such as by using different SQL clauses, joins, subqueries, aggregation functions, and conditions.

Generated SQL Queries:

Figure 5: The prompt of SQL evolution.

You are provided with a set of SQL keywords and a database schema. Your task is to generate 10 pairs of natural language questions and corresponding SQL queries. Each SQL query should incorporate as many of the provided SQL keywords as possible while being valid and executable based on the given database schema. Ensure that the questions and SQL queries are varied and meaningful, reflecting realistic use cases.

Database Schema: {database_schema}

SQL Keywords: {sql_prototype_keywords}

Instructions:
-- Generate a natural language question that accurately reflects the SQL query.
-- Ensure that the final SQL query includes as many of the SQL keywords as possible.
-- The question-SQL pair should make sense in the context of the given database schema.

Generated Question—SQL Pairs:

Figure 6: The prompt of pseudo-sample synthesis.

You are provided with a pair consisting of a natural language question and a corresponding SQL query, along with the database schema. Your task is to reflect on this pair and verify if the SQL query correctly answers the question based on the given schema.

Sample pairs: {sample_pairs}

Database Schema: {database_schema}

Instructions:
--If the pair matches, respond with "Yes", and don't give unnecessary explanations.
--If the pair does not match, respond with "No", provide corrections to the question or SQL query.

Generated Response:

Figure 7: The prompt of self-correction.

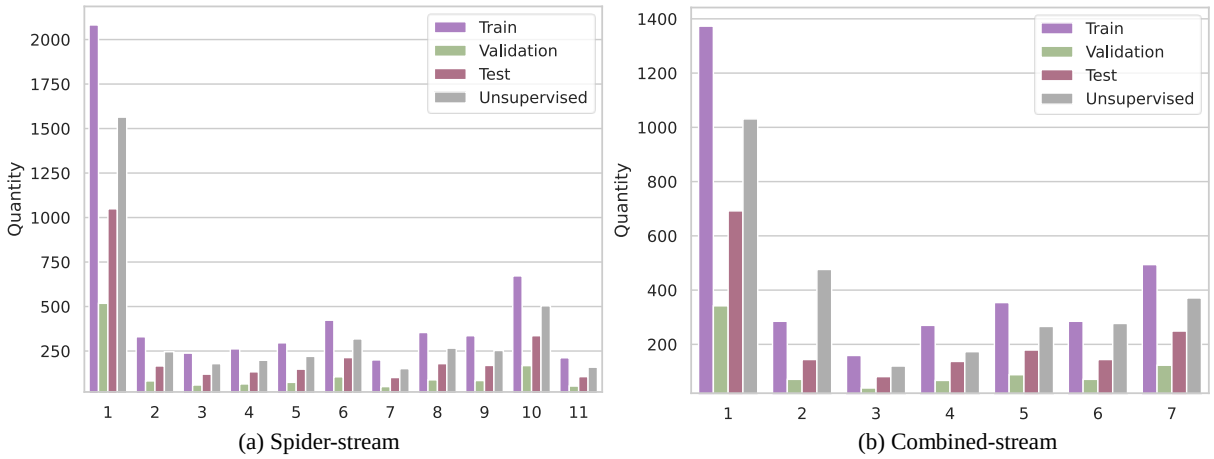

Figure 8: Number statistics of Spider-stream and Combined-stream.

- $\text{ACC}_\text{w} = a_{\mathcal{T}_\text{test}^{(1:|\mathcal{T}|)}}$, $\mathcal{T}_\text{test}^{(1:|\mathcal{T}|)} = \cup_{i=1}^{|\mathcal{T}|}\mathcal{T}_\text{test}^{i}$ represents the combined test set from task $\mathcal{T}_1$ to $\mathcal{T}_{|\mathcal{T}|}$.
- $\text{BWT} = \frac{1}{|\mathcal{T}|-1}\sum_{i=1}^{|\mathcal{T}|-1}(a_{|\mathcal{T}|,i} - a_{i,i})$
- $\text{FWT} = \frac{1}{|\mathcal{T}|-1}\sum_{i=2}^{|\mathcal{T}|}(a_{i-1,i} - \hat{a}_i)$, where $\hat{a}_i$ represents the accuracy of a randomly initialized model on task $\mathcal{T}_i$.

B.3 Implementation Details

Our experiments are conducted using the Pytorch framework (Paszke et al., 2019) with the Adam optimizer (Kingma and Ba, 2017). Following previous work (Chen et al., 2023c), we use T5-base[2] and T5-large[3] as backbone models. For full fine-tuning, batch sizes are set to 12 and 8, with a learning rate of 1e-4. The maximum training epoch is 150, and the evaluation delay is set to 50 epochs. The input and output length limits are set to 512 and 256, respectively. In parameter-efficient fine-tuning, we experiment with two representative methods: Adapter and LoRA, both widely used for updating LLM parameters, with a learning rate of 1e-3 and an evaluation delay of 50 epochs. Our method is implemented using the Transformer (Wolf et al., 2020) library and AdapterHub[4] repository.

Adapter (Poth et al., 2023) adapts to new tasks by inserting additional small networks (usually feed-forward neural networks) between the layers of the model, without modifying the existing weights. Each Adapter layer typically consists of a dimensionality reduction and restoration through linear transformations, along with a nonlinear activation function. We use Bottleneck Adapters, which introduce bottleneck feed-forward layers in each layer of the transformer model. The forward computation process is as follows:

$$\mathbf{h} = \mathbf{W}_\text{up} \cdot f(\mathbf{W}_\text{down} \cdot \mathbf{x}) + \mathbf{r} \tag{10}$$

Where $\mathbf{W}_\text{up}$ and $\mathbf{W}_\text{down}$ represent the projection matrices. $\mathbf{r}$ denotes the residual connection. We set the reduction factor to 16, and the nonlinear activation function is ReLU.

LoRA (Low-Rank Adaptation) (Hu et al., 2022) is an efficient fine-tuning technique that adjusts the weights of certain layers in a pre-trained model by introducing low-rank matrices. These low-rank matrices serve as modifiers to the pre-trained weights rather than replacements. For the output of a linear layer $\mathbf{h} = \mathbf{W}_0\mathbf{x}$, the reparameterization process is as follows:

$$\mathbf{h} = \mathbf{W}_0\mathbf{x} + \frac{\alpha}{r}\mathbf{B}\mathbf{A}\mathbf{x} \tag{11}$$

Here, $\mathbf{A} \in \mathbb{R}^{r\times k}$ and $\mathbf{B} \in \mathbb{R}^{d\times r}$. $\mathbf{W}_0$ remains fixed during training. We set the low rank to 8. In summary, LoRA modifies existing weights, while Adapter adds new layers without altering the original weights.

During the warm-up phase, we utilize different types and sizes of LLMs, including Mistral-7B-Instruct-v0.3[5], Meta-Llama-3-8B-Instruct[6], and Mixtral-8x7B-Instruct-v0.1[7]. All models were run

[2] https://huggingface.co/google/t5-base-lm-adapt
[3] https://huggingface.co/google/t5-large-lm-adapt
[4] https://adapterhub.ml
[5] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
[6] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[7] https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

using the vLLM[8] inference framework with the following settings: temperature of 0.6, top_p of 0.95, and max_token set to 256.

B.4 Baselines

Following the setting of previous works (Chen et al., 2023b,c; Liu et al., 2024b), we compare our approach with three types of baseline methods. In addition to citing results from the original papers, certain baselines using different backbone models and evaluation approaches are reimplemented using their official code. Four key metrics are reported: ACC_a, ACC_w, BWT, and FWT, to provide deeper insights into the performance of existing methods on CSP. The details of each baseline are as follows:

(1) Sequential Fine-Tuning (SFT) (Yogatama et al., 2019) is a naive method of sequentially fine-tuning tasks in a stream, typically suffering from severe catastrophic forgetting and limited generalization, representing the lower bound of continual learning performance. We use the same backbone parameter settings as our method.

(2) Rehearsal-based Methods mitigate forgetting by repeatedly replaying real data from previous tasks during the current task. They allocate memory for each task to store past data, which is replayed in every training batch to directly enhance performance in continual learning tasks. Some approaches, such as SFNET, also incorporate manually collected semi-supervised data. While comparisons with these methods may not be entirely fair, as our approach does not require replaying historical data or using external manual data, it still shows strong competitive performance.

EMAR (Han et al., 2020) enhances memory consolidation by combining prototype replay of both new and old relational data, allowing the model to retain prior knowledge while learning new relations, thus preventing catastrophic forgetting through memory reconsolidation. We do not directly reference the results from Chen et al. (2023c) because the original paper does not specify the task sequence order, so we reproduce the results using the official code.

SFNET (Chen et al., 2023b) is a semi-supervised continual learning method that integrates self-training (for pseudo-label generation) and episodic memory replay (to retain information from previous tasks), leveraging both supervised and unsupervised data. It employs a teacher-student framework where the teacher optimizes the current task through self-training, and the student learns from pseudo-labeled data across all tasks to ensure overall task performance. Notably, the original SFNET uses GRAPPA (Yu et al., 2021) (encoder-only), which is pre-trained on semantic parsing, as its backbone model and evaluates a simplified custom intermediate representation. However, GRAPPA utilizes database information from the Spider dataset during pre-training, which raises concerns about potential data leakage. Moreover, its generated intermediate representation cannot be directly executed on the database, and converting it to SQL results in significant performance loss. To ensure fair comparisons with other methods, we follow previous work (Chen et al., 2023c) and use the T5 series models based on a seq2seq architecture. We carefully replace the backbone model while keeping all other parameters from the original code unchanged and report the updated results.

(3) PET-based Methods typically require training on supervised data, and then the model's backbone parameters are frozen. Efficient parameter tuning is then applied to learn a PET for each task in the task stream. During inference, these methods generally operate under two configurations:

① **Task-aware**: This is an ideal setting that focuses solely on the CSP training phase, bypassing the task selection process during inference, thereby simplifying continual learning. Specifically, the task identifier for each sample is known during inference, so only the corresponding PET module needs to be loaded. However, this setting limits the model's generalization ability as it cannot handle samples from unseen tasks. Such methods include:

- **PEFT** (Chen et al., 2023c) is a vanilla method based on the ideal setting, where it learns a specific PET module for each task and loads it during inference. This approach makes the results independent of task order, so we report the results from the original paper.
- **C3** (Chen et al., 2023c) builds upon PEFT by introducing a teacher-student architecture and using in-context tuning to strengthen contextual information. Since the original paper does not specify the task stream order, we report the results provided in the official code repository.

② **Task-agnostic**: This challenging and more re-

[8] https://docs.vllm.ai/en/latest/

alistic setting requires the model to infer without knowing the task identifier during inference. The model must autonomously select the appropriate PET module, allowing it to naturally generalize to unseen tasks. This paper focuses on this scenario.

- **Fusion** (Pfeiffer et al., 2021) improves multi-task learning by separating knowledge extraction (Adapter training) from knowledge combination (fusion), preventing catastrophic forgetting and task interference, while enabling effective cross-task information transfer. Specifically, it trains an additional layer on each task to fuse all previously learned Adapter modules.

- **ProgPrompt** (Razdaibiedina et al., 2023) progressively accumulates learned prompts over the task stream, so the prompt length grows with the number of tasks. We set the prompt length to 50 and prompt learning rate to 0.3.

- **L2P** (Wang et al., 2022) introduces a prompt-based continual learning method that predefines a subset of learnable prompt parameters in a prompt pool, allowing dynamic prompt selection and task-specific learning without modifying the large model parameters. Since the original method is designed for classification tasks, we adapt it to sequence generation based on prior work (Yadav et al., 2023). The initial prompt pool size is set to 5 $\times$ the number of tasks, and the Top-1 PET module is selected for each instance.

- **Soup** (Chronopoulou et al., 2023) is a simple yet effective method that improves domain adaptation by performing weight-space averaging of domain-specific adapters, achieving better performance on new domains without additional training.

C More Results and Analysis

C.1 Impact of Memory Size and Ideal Settings

Table 4 presents the impact of different memory sizes and ideal settings on baseline performance. It is evident that rehearsal-based methods heavily depend on memory size. When memory size is reduced to 1, EMAR's performance on Spider-stream even falls below that of SFT. SFNET, which is supported by replay strategies and additional unlabeled data, achieves better results. However, our method consistently outperforms without relying on any replay or external data.

For baselines based on ideal settings, we examine their performance in real-world scenarios with unknown task identifiers by employing random loading of pre-trained PET modules in the core PEFT approach. The results show a significant decline in ACC_a and ACC_w, indicating that PEFT methods experience severe catastrophic forgetting when not relying on ideal settings. Interestingly, random loading reduces task interference, resulting in better BWT and FWT performance on Spider-stream. However, on Combined-stream, the diversity of task types leads to a drop in BWT. In contrast, our method remains stable and unaffected by these external conditions.

C.2 Cold Start Results

Table 5 shows the performance of different methods in the cold-start scenario, where only a small amount of labeled data is available for the first task. Nearly all methods experience performance degradation, highlighting the challenge of the cold-start scenario. Our findings include:

Dependence on Initialization for PET-based Methods. From the results of L2P, Fusion, Soup, ProgPrompt, and PEFT, we observe a sharp performance drop, even below that of SFT. This is due to the requirement for strong initial performance in these methods.

Differences among PET Methods. In ideal settings, prompt-based methods perform worse than others, indicating greater reliance on supervised data in the initial task. This may be due to the fixed prefix design in prompt methods, which, while minimally invasive to the model, struggles with complex sequence generation tasks (Qin and Joty, 2022). The prefix also consumes input space, truncating long inputs. This issue is exacerbated in ProgPrompt, where the growing prefix severely impacts performance on later tasks.

Impact of Replay and External Data. Full-parameter fine-tuning methods like EMAR and SFNET, which utilize historical data replay and incorporate additional semi-supervised data, perform well in preventing catastrophic forgetting and enhancing generalization. As seen in Appendix C.1, their performance is highly dependent on the memory size allocated for replay data (the more data replayed, the slower the forgetting). In contrast, PET-based methods, like ours, which do not rely on data replay, show limited effectiveness in gen-

Method	Memory Size	Spider-stream				Combined-stream			
		ACC_a	ACC_w	BWT	FWT	ACC_a	ACC_w	BWT	FWT
Original Method									
SFT	0	51.9/57.9	47.1/52.8	-19.6/-14.9	**29.6/33.2**	51.0/36.0	52.4/38.3	-11.1/-17.6	19.9/11.1
Rehearsal-Based Method									
EMAR	1	46.7/51.6	43.6/47.8	-26.7/-23.5	26.4/30.7	53.7/51.2	50.8/49.4	-15.8/-16.5	23.1/19.0
	10	55.0/57.4	50.0/52.5	-17.8/-17.1	28.4/32.1	60.6/58.9	55.7/55.3	-7.9/-7.1	28.7/25.4
	15	55.6/59.4	50.6/55.1	-17.5/-15.0	28.5/31.1	61.2/59.4	57.6/57.9	-7.3/-6.8	28.4/25.9
Rehearsal-Based Method Requiring Unsupervised Data									
SFNET	0	51.8/57.8	47.1/52.2	-18.9/-13.8	28.3/32.7	56.2/49.4	51.8/48.4	-7.9/-14.4	21.0/14.6
	1	50.9/54.2	50.3/54.1	-20.2/-19.1	24.8/27.9	59.1/54.1	53.3/50.8	-6.3/-9.7	25.6/19.9
	10	58.1/61.7	52.7/56.8	-12.4/-9.9	28.7/32.2	60.7/57.7	55.2/53.9	-6.0/-6.7	28.7/26.0
	15	60.4/62.5	54.6/57.4	-11.3/-10.7	28.8/31.8	61.3/60.9	57.5/56.7	-5.4/-5.2	29.7/27.5
Random PET									
$Random_{Adapter}$	0	37.8/40.1	42.6/45.5	**4.8/4.2**	25.5/28.2	21.8/17.1	28.1/26.3	-17.4/-20.3	25.2/21.0
$Random_{LoRA}$	0	36.9/40.2	42.3/45.8	2.6/2.6	25.9/29.1	30.5/25.0	35.1/32.4	-8.0/-10.8	**32.8/30.6**
Without Rehearsal and Unsupervised Data									
$APEX_{Adapter}$	0	61.0/62.4	**63.4/64.8**	1.3/1.4	25.1/28.3	62.7/59.3	**64.9**/63.1	6.8/9.2	21.9/20.7
$APEX_{LoRA}$	0	**63.3/65.3**	62.1/63.9	0.4/0.5	24.4/27.5	**63.5/61.4**	64.3/**63.5**	**7.2/10.6**	29.2/25.8
Upper Bound									
ORACLE		75.6/77.0	76.4/78.4	4.2/3.3	29.7/32.0	70.2/68.2	71.1/70.8	5.6/4.7	29.2/26.6

Table 4: Effect of memory size and ideal settings on baseline performance (T5-base).

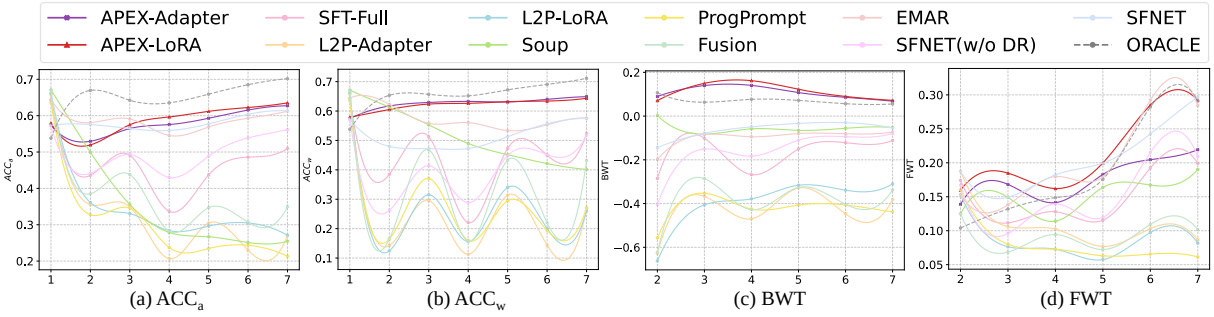

Figure 9: Comparison of results (EM) across all seen tasks on Combined-stream (T5-base).

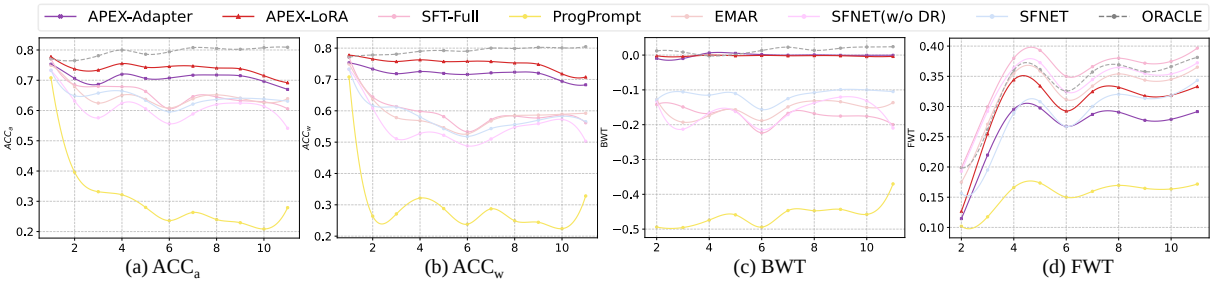

Figure 10: Comparison of results (EM) across all seen tasks on Spider-stream (T5-large).

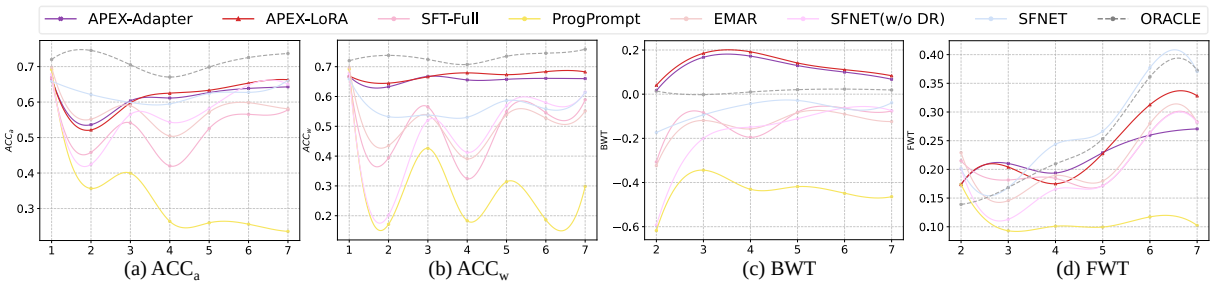

Figure 11: Comparison of results (EM) across all seen tasks on Combined-stream (T5-large).

Method	DR	PET	Spider-stream (*cold start*)				Combined-stream (*cold start*)			
			ACC_a	ACC_w	BWT	FWT	ACC_a	ACC_w	BWT	FWT
T5-Base(220M)										
SFT	✗	Full	49.6/55.4	45.7/52.0	-20.8/-16.3	26.7/27.8	39.9/36.4	32.0/29.4	-25.9/-28.5	24.5/20.5
	✗	Adapter	40.4/44.9	35.6/39.8	-33.2/-29.4	23.0/23.6	28.9/25.6	18.5/16.4	-40.1/-43.2	23.4/19.4
	✗	LoRA	36.4/39.6	32.6/36.0	-30.2/-28.8	21.4/22.9	27.7/25.7	17.0/15.8	-35.6/-36.3	20.6/17.4
L2P	✗	Adapter	25.7/28.4	21.4/23.7	-41.0/-39.9	15.9/16.4	20.1/16.5	12.3/10.1	-47.5/-49.6	15.0/12.8
	✗	LoRA	29.6/33.5	26.0/30.5	-39.2/-37.4	18.8/19.7	27.1/24.0	16.7/14.8	-40.7/-42.6	20.7/16.8
Fusion	✗	Adapter	25.7/30.1	23.9/28.3	-14.1/-13.3	23.4/24.7	22.0/20.3	13.5/12.4	-28.1/-29.4	18.1/15.3
Soup	✗	Adapter	9.0/10.2	8.1/9.7	-2.5/-2.5	4.5/4.8	22.5/19.8	13.7/12.0	-0.3/0.3	17.9/15.6
ProgPrompt	✗	Prompt	8.7/11.0	7.1/9.4	-34.7/-35.3	8.4/8.5	0.2/0.6	0.1/0.3	-10.2/-11.2	0.1/0.0
EMAR	✓	Full	**57.2/61.4**	53.2/58.1	-15.6/-12.2	**27.4/28.0**	54.4/51.5	50.1/50.0	-10.0/-12.1	17.8/15.9
SFNET♣	✗	Full	50.1/55.3	46.4/51.9	-21.0/-17.9	25.9/27.5	34.5/31.9	24.4/23.3	-33.7/-34.6	22.5/19.2
SFNET♣	✓	Full	55.4/59.3	51.3/55.5	-13.4/-11.4	24.7/25.6	**58.4/56.7**	55.9/56.3	-6.8/-7.0	24.2/20.0
APEX(Ours)	✗	Adapter	54.1/55.6	53.5/55.5	2.9/2.7	9.1/9.4	58.2/55.5	**57.7/57.1**	**7.6/9.3**	26.3/**24.3**
	✗	LoRA	**57.2**/59.3	**57.8/60.1**	**3.5/3.7**	13.0/13.3	57.3/55.3	56.1/55.7	2.5/2.2	**27.6**/23.4
Ideal Setting										
PEFT♢	✗	Prompt	11.5/14.5	14.9/18.6	0.0/0.0	N/A	25.0/23.1	16.6/18.2	0.0/0.0	N/A
	✗	Adapter	69.6/71.0	70.1/72.1	0.0/0.0	N/A	61.7/62.0	64.3/65.7	0.0/0.0	N/A
	✗	LoRA	63.6/65.8	64.7/67.4	0.0/0.0	N/A	55.3/55.0	58.4/59.0	0.0/0.0	N/A
T5-Large(770M)										
SFT	✗	Full	60.9/**67.7**	56.9/63.5	-16.8/-11.0	**34.6/36.3**	49.5/47.2	44.3/41.7	-22.1/-23.0	32.6/27.0
	✗	Adapter	48.2/53.2	44.2/48.9	-28.7/-24.9	30.5/31.4	32.5/30.7	22.6/21.4	-40.4/-42.0	29.4/23.9
	✗	LoRA	48.0/53.7	43.8/49.0	-27.3/-22.4	32.1/33.5	30.1/27.8	18.5/17.1	-41.2/-43.4	28.3/23.4
ProgPrompt	✗	Prompt	13.9/16.5	11.5/13.8	-38.3/-37.1	10.2/10.7	1.1/1.5	1.4/1.5	-13.5/-14.1	0.3/0.2
EMAR	✓	Full	61.2/65.3	59.0/63.2	-14.1/-11.7	32.6/33.3	55.0/51.5	53.0/50.4	-11.9/-14.4	25.2/19.6
SFNET♣	✗	Full	56.4/61.4	51.5/56.0	-18.3/-14.7	31.2/32.1	40.9/37.0	32.6/29.7	-29.9/-33.5	28.8/25.0
SFNET♣	✓	Full	**63.4**/67.3	58.6/62.6	-10.6/-8.8	32.8/34.4	**62.2/61.9**	59.8/60.6	-6.5/-5.1	31.3/27.2
APEX(Ours)	✗	Adapter	58.9/59.7	60.3/61.9	**4.7/5.0**	16.6/17.8	61.9/57.4	**62.9/61.1**	-0.1/-0.1	27.6/25.3
	✗	LoRA	61.2/62.9	**61.4/63.6**	2.5/2.5	18.1/18.3	61.4/60.8	58.5/58.9	**4.6/4.4**	**35.2/32.7**
Ideal Setting										
PEFT♢	✗	Prompt	42.6/46.2	51.7/55.1	0.0/0.0	N/A	43.9/45.5	38.2/40.8	0.0/0.0	N/A
	✗	Adapter	74.1/75.6	74.7/76.6	0.0/0.0	N/A	65.4/65.9	70.1/71.2	0.0/0.0	N/A
	✗	LoRA	72.1/73.4	72.3/74.2	0.0/0.0	N/A	63.3/63.8	66.7/68.5	0.0/0.0	N/A

Table 5: Results on Spider-stream and Combined-stream datasets under *cold start* settings (%). DR indicates whether historical data replay is used, with a memory size set to 15. PET refers to different training architectures, with FULL being full-parameter tuning. ♣ indicates the use of additional unsupervised data, and ♢ means a known task ID, but forward transfer is impossible. The best results are highlighted in bold, and the second-best results are underlined. Our results are from three random runs.

eralization. For example, in Spider-stream, even though Soup is designed for task generalization and uses a fusion of Adapters similar to our approach, its FWT performance is only 4.5/4.8.

Robustness and Effectiveness of Our Method. APEX achieves consistent performance without requiring historical data replay or external semi-supervised data. It significantly outperforms PET-based baselines and rivals or surpasses replay-based methods, with a 16.9% improvement over SFNET in BWT, further proving its effectiveness.

C.3 Results Till the Seen Tasks

Figures 9 to 11 provide further comparison results on previously seen tasks. It can be observed that our method shows significant advantages across most metrics, even surpassing ORACLE performance in some cases. We also notice that methods based on full-parameter fine-tuning, such as SFT-Full, EMAR, and SFNET, generally perform better on FWT compared to PET-based methods, indicating that a larger parameter update range helps the model generalize to new tasks more effectively.

C.4 Impact of Threshold γ

To further explore the impact of threshold γ, we evaluate sample visibility prediction accuracy using known task identifiers in the test set across different threshold values, as shown in Figure 12. The results indicate that as the threshold increases, the accuracy for seen and unseen samples becomes inversely related. The threshold of 1.6, used in Section 6.2, is located near the intersection of the two curves on the x-axis, which further demonstrates that balancing the prediction accuracy for seen and unseen samples is key to balancing the model's

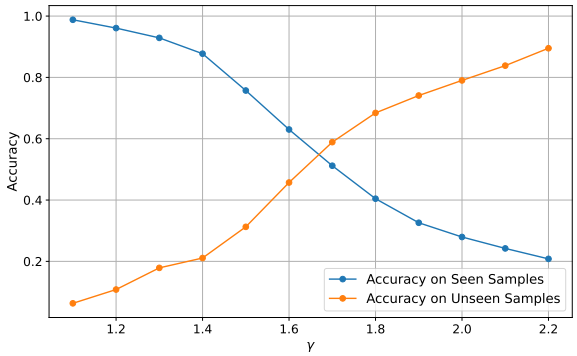

Figure 12: Prediction accuracy of sample history visibility (seen/unseen) under different γ settings.

resistance to forgetting and its generalization.

C.5 Warm Up with Different LLMs

To further explore the impact of different types and scales of LLM warm-up on model performance, in addition to the previously mentioned Mixtral-8x7B-Instruct-v0.1, we also used Mistral-7B-Instruct-v0.3 and Meta-Llama-3-8B-Instruct. Table 6 presents the results of our method under different LLM warm-up settings, as well as the performance of LLMs in zero-shot inference. It is evident that the zero-shot results of LLMs alone are not satisfactory, further highlighting the importance of continual learning. Notably, our method, when combined with various types and scales of LLMs, continues to demonstrate strong performance, especially in the BWT metric, consistently exceeding the upper performance bound, which further validates the effectiveness of our approach.

Model	**ACC$_a$**	**ACC$_w$**	**BWT**	**FWT**
SFT-Full	51.0/36.0	52.4/38.3	-11.1/-17.6	19.9/11.1
SFT-LoRA	46.9/40.6	48.6/45.1	-21.0/-26.3	16.8/11.1
Mistral-7B	27.3/22.1	24.9/23.2	-/-	-/-
+APEX	64.3/62.8	66.7/66.9	8.1/9.6	23.4/22.1
Llama-3-8B	33.5/26.2	27.7/24.9	-/-	-/-
+APEX	64.9/63.1	66.9/65.9	7.8/9.5	24.0/21.5
Mixtral-8x7B	28.6/24.4	27.0/26.0	-/-	-/-
+APEX	63.5/61.4	64.3/63.5	7.2/10.6	29.2/25.8
Ideal Setting & ORACLE				
PEFT-LoRA	63.1/61.4	67.0/66.7	0.0/0.0	N/A
ORACLE	70.2/68.2	71.1/70.8	5.6/4.7	29.2/26.6

Table 6: Comparison of LLM zero-shot results and the performance of different LLMs warming up APEX$_{\text{LoRA}}$ on Combined-stream (T5-base).