Mixture of LoRA Experts for Continual Information Extraction with LLMs

Zitao Wang[†] **Xinyi Wang**[†] **Wei Hu**[†,‡,*]
[†] State Key Laboratory for Novel Software Technology, Nanjing University, China
[‡] National Institute of Healthcare Data Science, Nanjing University, China
ztwang.nju@gmail.com, xywang.nju@gmail.com, whu@nju.edu.cn

Abstract

We study continual information extraction (IE), which aims to extract emerging information across diverse IE tasks incessantly while avoiding forgetting. Existing approaches are either task-specialized for a single IE task or suffer from catastrophic forgetting and insufficient knowledge transfer in continual IE. This paper proposes a new continual IE model using token-level mixture of LoRA experts with LLMs. We leverage a LoRA router to route each token to the most relevant LoRA experts, facilitating effective knowledge transfer among IE tasks. We guide task experts' selection by task keys to retain the IE task-specific knowledge and mitigate catastrophic forgetting. We design a gate reflection method based on knowledge distillation to address forgetting in the LoRA router and task keys. The experimental results show that our model achieves state-of-the-art performance, effectively mitigating catastrophic forgetting and enhancing knowledge transfer in continual IE.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities as a powerful expert tool across diverse applications (Azaria et al., 2024), and knowledge plays a crucial role in shaping their effectiveness and future potential (Chen, 2024). Among such knowledge-intensive applications, information extraction (IE) (Andersen et al., 1992) serves as a fundamental task, aims to identify and structure specific information from unstructured natural language text. It encompasses diverse tasks, e.g., named entity recognition (NER), relation extraction (RE), and event detection (ED), based on the different extraction targets (entity, relation, event, etc.). In practice, new types and tasks are emerging continually across diverse IE scenarios. Studying *continual learning* (Ring, 1994) within IE scenarios is crucial for building adaptive and scalable IE models.

Previous continual IE models (Wang et al., 2023c; Zhang et al., 2023a; Zhao et al., 2023; Le et al., 2024b, 2025) are task-specialized, resulting in separate application scenarios and isolated models for different IE tasks. Real-world applications demand a unified model that can continuously extract user-specified information across diverse IE tasks, rather than being limited to a single task. The task-specific design hinders model reuse and increases deployment complexity, limiting the scalability of continual IE. Moreover, the different IE tasks are not completely independent of each other. There is shared knowledge among different IE tasks that a model can leverage. The isolated models significantly limit the ability to share knowledge between related tasks and settings. Therefore, these task-specialized models hinder the development of IE in terms of cross-task adaptation and effective knowledge sharing. Toward this issue, we define a new problem called *continual IE* to address. Compared to task-specialized continual learning, continual IE expects the model to extract different information of different IE tasks from a continuous data sequence, not only extracting new information but remembering all learned information so far.

Existing continual learning methods for IE focus on alleviating catastrophic forgetting (Thrun and Mitchell, 1995; French, 1999) in their single tasks using knowledge transfer and experience replay. These task-specialized models fall short in knowledge transfer among IE tasks and cannot adapt to continual IE. Recent continual learning studies (Wang et al., 2022b; Razdaibiedina et al., 2023; Wang et al., 2023a; Zhao et al., 2024; Le et al., 2024a) for LLMs with parameter-efficient finetuning (PEFT) can be applied to continual IE by learning and selecting independent PEFT blocks such as soft prompt (Lester et al., 2021) or Low-Rank Adaptation (LoRA) (Hu et al., 2022) for each task.

* Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 13324–13339
November 4-9, 2025

However, they are still struggled with catastrophic forgetting and insufficient knowledge transfer. On one hand, the selection modules in these methods are crucial, as they need correct knowledge for prediction. However, these methods only use concatenation or attention weights at the sentence level for selecting the correct PEFT blocks. These sentence-level selection modules are insufficiently discriminative in continual IE, where the semantic features of sentences in different IE tasks are similar. Thus, these methods easily make incorrect choices and use wrong knowledge, leading to catastrophic forgetting. On the other hand, these methods only update the PEFT blocks for the current task during training, while freezing those that have already been learned. This prevents them from the backward transfer of useful knowledge from new tasks to old ones. Moreover, their limited use of previous knowledge in current training hinders the forward transfer from old tasks to new ones.

To address these issues in continual IE, we propose a novel continual IE model, **MoLE-CIE**, through the token-level mixture of LoRA experts with LLMs. Specifically, we route each token to the most relevant LoRA experts by a LoRA router to facilitate knowledge transfer among IE tasks. We allocate task experts and a shared IE expert to retain the task-specific knowledge of IE. We design a gate reflection method using knowledge distillation to mitigate the forgetting of the LoRA router and task keys. Our main contributions are threefold:

- Rather than task-specialized methods, we extend continual learning to universal IE tasks and design a new continual IE model MoLE-CIE.
- We propose token-level mixture of LoRA experts and gate reflection to deal with catastrophic forgetting and insufficient knowledge transfer in continual IE.
- We conduct extensive experiments on six benchmark datasets for three major IE tasks. Our results show that MoLE-CIE achieves state-of-the-art performance in continual IE. It also performs well in facilitating knowledge transfer and mitigating catastrophic forgetting.

2 Related Work

2.1 Information Extraction

Conventional IE models rely on a pre-trained encoder to capture the semantic representations in text and treat extraction as a classification task (Soares et al., 2019), a sequence labeling task (Huang et al., 2015), or a question-answer task (Du and Cardie, 2020). In recent years, the works (Li et al., 2023; Zhang et al., 2023b) use LLMs as data augmentation tools to improve extraction performance. The works (Wan et al., 2023; Heng et al., 2024; Zhu et al., 2024) utilize task-specific prompts to guide the model's extraction through chain-of-thoughts (Wei et al., 2022) or iterative self-improvement. The works (Guo et al., 2024; Sainz et al., 2024) establish IE schemata through code classes, thereby unifying IE tasks and facilitating IE via code generation. These models are not designed for continual IE, as they learn all types at once. When deployed in continual settings, they suffer from severe knowledge forgetting and poor generalization in continuous data sequences.

2.2 Continual Learning

There is a substantial body of work on continual learning in three main IE tasks: continual ED (Yu et al., 2021; Wang et al., 2023c; Le et al., 2024b); continual RE (Zhao et al., 2022, 2023; Le et al., 2025), and continual NER (Zheng et al., 2022; Zhang et al., 2023a; Yu et al., 2024, 2025). These methods alleviate catastrophic forgetting by knowledge transfer and experience replay. However, they are designed for single IE tasks and cannot generalize to other IE tasks, making them unsuitable for more comprehensive continual IE scenarios.

Recent continual learning studies for LLMs with PEFT learn and select PEFT blocks, such as soft prompt (Lester et al., 2021) or LoRA (Hu et al., 2022), for each task. These methods follow a pipeline that trains new PEFT blocks for new tasks and freezes the previously learned ones. For prompt-based methods, various prompt selection strategies have been explored: concatenating all prompts (Razdaibiedina et al., 2023), similarity in task feature distributions (Wang et al., 2023b), retrieving from a fixed prompt pool (Wang et al., 2022b), and non-linear residual gates (Le et al., 2024a). For LoRA-based methods, (Wang et al., 2023a) enforces orthogonality constraints during training, while (Zhao et al., 2024) leverages shared attention weights to guide LoRA selection. Although these methods are applicable to continual IE, they are limited in knowledge transfer due to parameter isolation and the limited use of previous knowledge. Furthermore, their reliance on the sentence-level PEFT block selection module leads

to incorrect choices due to the high semantic similarity in IE tasks, thereby exacerbating catastrophic forgetting. For full finetuning, the methods (Wang et al., 2024; He et al., 2024) equip LLMs with continual learning capabilities through replay and attention distillation. But they suffer from high resource consumption and low knowledge transferability. We address these problems and propose a novel continual IE model.

3 Task Definition

IE aims to extract structured information from unstructured natural language text. In continual IE, a sequence of K tasks $\{T_1, T_2, \ldots, T_K\}$ is presented to the model in a sequential manner. Each task is an independent IE sub-task with its own training set D_i^{train}, development set D_i^{dev}, test set D_i^{test}, and type set R_i. R_i of each task T_i is disjoint with other tasks.

At the i-th stage, a continual IE model is trained on D_i^{train} and evaluated on current and all previous test sets $\tilde{D}_i^{\text{test}} = \bigcup_{t=1}^{i} D_t^{\text{test}}$ (their corresponding training sets have been trained in the previous stage). In other words, the model must handle the emerging types within the same extraction task and shift to different extraction tasks over time. The model will not be informed of the specific extraction task during inference.

4 Methodology

Figure 1 shows the framework of our continual IE model MoLE-CIE. We add the **Mixture of LoRA Experts** (MoLE) module to the attention layer of the pre-trained transformer blocks. The attention from the pre-trained model is combined with the attention produced from the MoLE module to form a new attention, which is used in the subsequent layers. During training, we freeze the parameters of the pre-trained model and only update the parameters of the MoLE module.

In the MoLE module, the model first routes each token to the LoRA experts that are most relevant to it. Then, the model selects task experts by calculating the similarity between the task keys and the input tokens. Finally, the model fixes an IE expert and aggregates the output of each LoRA expert by the corresponding weight.

For continual training, we design the **Gate Reflection** module to distill past knowledge in the LoRA router and task keys. We pick and store a few instances of new types for the next task.

4.1 Base Model and Experience Replay

Base model. To handle various IE tasks, we model the IE task as a generation problem by finetuning LLMs. We use the pre-trained LLaMA-3.1-8b (Grattafiori et al., 2024) as the base model and finetune it by LoRA (Hu et al., 2022) to adapt to the continual learning tasks.

Experience replay. Inspired by previous methods (de Masson d'Autume et al., 2019; Wang et al., 2019; Cao et al., 2020) on continual IE, we replay a small number of data from previous tasks to alleviate catastrophic forgetting. At the i-th stage, we merge the replay data D_i^{replay} into the current training set D_i^{train} and train the model on it. We use the k-means algorithm to cluster the feature representations of each type's instances, where the number of clusters equals the memory size M. For details, we utilize the embedding layer of the backbone LLM to obtain features for clustering. We employ Euclidean distance (L2 norm) as the distance metric. We select the instances closest to the centroid of each cluster and store them as the replay data.

4.2 Mixture of LoRA Experts

In continual IE, the model needs to handle different extraction tasks and learn newly emerging types within the same task. This raises higher demands for knowledge retention and transferability of the model. We propose a new continual IE model that allocates the appropriate LoRA experts and task experts at the token level. It allows our model to fully integrate knowledge across different tasks and types and mitigate the forgetting of task-specific knowledge.

LoRA expert selection by LoRA router. With a LoRA router, we assign the most relevant LoRA experts to each token in the input. This implies that tokens from different types or different tasks can be assigned to the same LoRA experts based on their semantic similarity. This approach enables the model to effectively integrate semantic knowledge across diverse types or tasks, thereby enhancing its knowledge transferability.

Given a sentence x, we first use the embedding layer of the backbone LLM to obtain the hidden states h_{x_j} for each token x_j in x. Then, we feed h_{x_j} into a LoRA router to get the scores S_{x_j} between x_j and each LoRA expert as follows:

$$S_{x_j} = \mathbf{W_2}\left(\tanh\left(\mathbf{W_1} h_{x_j}\right)\right), \quad (1)$$

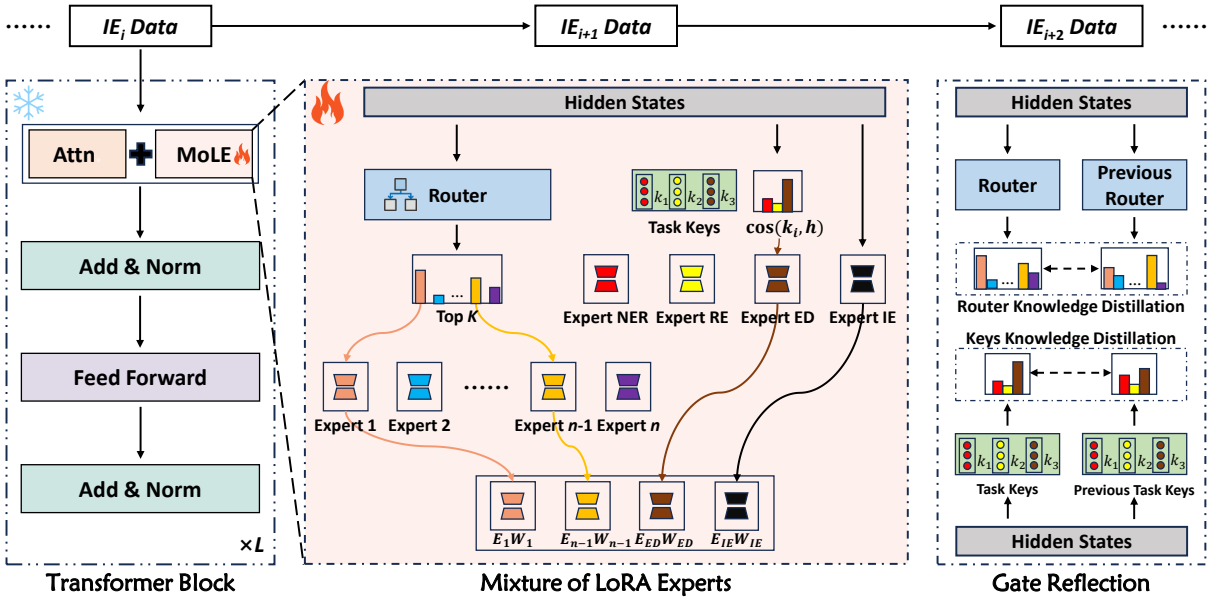

Figure 1: Framework of our proposed continual IE model MoLE-CIE.

where $\mathbf{W1} \in \mathbb{R}^{n \times h}$ and $\mathbf{W2} \in \mathbb{R}^{n \times n}$ are trainable parameters. h is the dimension of hidden states and n is the number of LoRA experts that can be routed. $\tanh(\cdot)$ is the hyperbolic tangent function.

We leverage the softmax activation function to model a probability distribution P over the experts:

$$P_j = \frac{\exp\left(S_{x_j,q}\right)}{\sum_{q=1}^{n} \exp\left(S_{x_j,q}\right)}, \tag{2}$$

where $S_{x_j,q}$ indicates the relevance score between x_j and LoRA expert E_q. We select the top-k suitable LoRA experts $E_{r_1}, \ldots, E_{r_k}$ and their corresponding weights $\theta_{r_1}, \ldots, \theta_{r_k}$.

Through this mechanism, our model can select the suitable LoRA experts at the token level, unlike previous methods (Wang et al., 2022b; Razdaibiedina et al., 2023; Wang et al., 2023a; Zhao et al., 2024; Le et al., 2024a) which operate at the sentence level. This enables the LoRA experts to develop varied capacities and efficiently handle diverse types and tasks.

Task expert selection by task keys. In the continual IE task sequence, after learning a newly emerging extraction task, the model is likely to forget the knowledge from the previously learned extraction tasks. To mitigate forgetting, we set task experts for each extraction task to preserve the corresponding task-specific knowledge. Based on the LoRA experts selected by the router, our model further selects the task experts that are most relevant to the input. Through these task experts, our model can retain the knowledge of previously learned tasks when learning new tasks. Additionally, our model updates the knowledge for the corresponding task when learning new types within the same extraction task. In contrast, previous methods (Razdaibiedina et al., 2023; Wang et al., 2023a; Zhao et al., 2024; Le et al., 2024a) freeze the LoRA parameters of the learned tasks, preventing effective updates to the knowledge of the same task.

We set a trainable task key K_m for each IE task T_m to record the task features of T_m. Due to the variation in sentence lengths across different inputs, we first perform an averaging operation on the hidden states h_x along the sentence dimension, denoted by $\text{avg}\left(h_x\right)$. We assign the corresponding task experts to the hidden states of the current input by calculating the similarity between the task keys and the hidden states as follows:

$$C_{x,m} = \cos\left(\text{avg}\left(h_x\right), K_m\right), \tag{3}$$

where $\cos(\cdot)$ is the cosine similarity between two features. We choose the task expert E_t with the highest score θ_t in normalized C_x and assign it to the input. To learn the shared knowledge across all extraction tasks, we additionally assign a fixed task expert E_{IE} to each input. The weight of θ_{IE} is $\theta_{\text{IE}} = 1 - \theta_t$.

Aggregation of selected LoRA experts. After selecting LoRA experts $E_{r_1}, \ldots, E_{r_k}$ and task experts E_t, E_{IE} by LoRA router and task keys, respectively, we obtain the MoLE attention A_{MoLE,x_j} on

token x_j:

$$A_{\text{MoLE},x_j} = \alpha \sum_{i=1}^{k} \theta_{r_i} E_{r_i}\left(h_{x_j}\right) + \beta\left(\theta_t E_t\left(h_{x_j}\right) + \theta_{\text{IE}} E_{\text{IE}}\left(h_{x_j}\right)\right), \quad (4)$$

where $E(h_{x_j})$ denotes the output of expert E given h_{x_j} as input. α and β are two hyperparameters. We add A_{MoLE,x_j} and the LLM's original attention as the final output of the attention layer to participate in the subsequent model training.

The training loss of the current task T_i is defined as follows:

$$\mathcal{L}_{\text{task}} = -\sum_{(x,y)\in D_i^{\text{train}}} \log P\left(y \mid x; \mathcal{M}_{\text{LLM}}, \mathcal{M}_{\text{MoLE},i}\right), \quad (5)$$

where $\mathcal{M}_{\text{LLM}}$ is the pre-trained LLM. $\mathcal{M}_{\text{MoLE},i}$ is the mixture of LoRA experts trained on T_i.

4.3 Gate Reflection

During training, the gates to experts (i.e. LoRA router and task keys) are updated to adapt to new types and extraction tasks. We need to reflect the learned knowledge for the gates to prevent them from forgetting too much. So, we propose the gate reflection module to address this issue for the gates with knowledge distillation (Hinton et al., 2015).

Router knowledge distillation. For the LoRA router, we enforce that the probability P_{i,x_j} over the experts predicted on token x_j by the current LoRA router does not deviate from the probability P_{i-1,x_j} predicted by the previous LoRA router. For each token x_j in the sentence x, we preserve the knowledge of the previous LoRA router by the router knowledge distillation loss:

$$\mathcal{L}_{\text{rkd}} = \sum_{x\in D_i^{\text{train}}} \sum_{x_j \in x} \text{KL}\left(P_{i-1,x_j} \parallel P_{i,x_j}\right), \quad (6)$$

where $\text{KL}(\cdot)$ is the KL divergence loss.

Keys knowledge distillation. The task keys are used to retain the feature knowledge for each task. If the task keys forget their knowledge in the continual task sequence, our model cannot select specific task experts correctly, which leads to catastrophic forgetting. For each sentence x, we expect the task similarity $C_{i,x}$ calculated between the current task keys and x to be similar to $C_{i-1,x}$ by the previous task keys, making the current task keys preserve more knowledge of the previous tasks. We implement this strategy by

$$\mathcal{L}_{\text{kkd}} = \sum_{x\in D_i^{\text{train}}} \text{KL}\left(C_{i-1,x} \parallel C_{i,x}\right). \quad (7)$$

We optimize the task loss $\mathcal{L}_{\text{task}}$ and distillation losses $\mathcal{L}_{\text{rkd}}$ and $\mathcal{L}_{\text{kkd}}$ with multi-task learning. The final training loss is

$$\mathcal{L} = \left(1 - \frac{|\tilde{R}_{i-1}|}{|\tilde{R}_i|}\right)\mathcal{L}_{\text{task}} + \frac{|\tilde{R}_{i-1}|}{|\tilde{R}_i|}\left(\gamma\mathcal{L}_{\text{rkd}} + \delta\mathcal{L}_{\text{kkd}}\right), \quad (8)$$

where $\tilde{R}_i = \bigcup_{t=1}^{i} R_t$ accumulates current and all previous types at T_i. γ and δ are hyperparameters.

5 Experiments and Results

In this section, we conduct experiments to evaluate our model and report the results. The source code is attached as supplementary materials. Please refer to the appendix for more experimental analyses. The source code is accessible online.[1]

5.1 Experiment Setup

Datasets. We carry out our experiments on three mainstream IE tasks. Appendix A gives details.

- **ED benchmarks.** (1) *ACE05-EN+* (Doddington et al., 2004) is a classic ED dataset with 33 event types. We follow (Lin et al., 2020) to pre-process it. (2) *MAVEN* (Wang et al., 2020) is a large-scale ED dataset containing 168 event types. We re-split the dataset due to the original dataset does not offer the annotations of the test set.

- **RE benchmarks.** (1) *Few-Rel* (Han et al., 2018) is a widely-used dataset for RE. Following (Cui et al., 2021; Zhao et al., 2022), we use 80 relations and each relation is assigned 700 samples. (2) *TACRED* (Zhang et al., 2017) is another popular RE dataset, which contains 42 relations and 106,264 samples.

- **NER benchmarks.** (1) *OntoNotes-5.0* (Hovy et al., 2006) is a popular dataset for NER with 18 entity types and 77k samples. (2) *i2b2* (Murphy et al., 2010) is a dataset with 16 entity types in the medical domain, bringing more diversity compared to other datasets.

Following (Yu et al., 2021; Cui et al., 2021), we partition each dataset into four subsets and merge them based on the task type. We standardize the input format for all extraction tasks to prevent task leakage. Please refer to Appendix B for details. The types and instances in each subset are disjoint. Thus, we generate four disjoint splits for each IE

[1] https://github.com/nju-websoft/MOLE-CIE

Sequence 1	ED_1	RE_1	NER_1	ED_2	RE_2	NER_2	ED_3	RE_3	NER_3	ED_4	RE_4	NER_4
SeqLoRA	88.74	36.27	27.44	28.04	26.81	25.80	21.55	14.81	22.64	23.60	25.73	16.33
Joint-training	88.74	93.65	94.11	91.33	91.23	90.72	89.32	88.26	90.62	85.94	97.92	87.13
L2P$^{\triangle}$	88.43	67.31	78.48	60.59	60.65	62.72	55.27	53.54	56.64	52.97	55.33	49.61
ProgPrompt$^{\triangle}$	88.74	63.05	75.90	66.79	61.25	64.55	51.97	54.34	63.26	51.36	50.43	50.23
NoRGa$^{\triangle}$	89.38	65.18	72.91	66.81	72.53	66.06	61.15	51.96	60.04	51.08	58.21	51.78
O-LoRA	88.74	65.94	80.85	68.54	70.26	65.64	42.45	50.09	58.66	53.86	57.41	50.16
SAPT	88.52	71.46	80.13	47.27	59.93	65.19	42.04	53.94	52.55	42.2	49.96	43.77
GPT-4o	30.65	36.16	50.07	44.29	41.28	41.69	37.59	35.37	41.97	38.63	37.06	36.95
MoLE-CIE	**90.35**	**85.45**	**84.49**	**70.68**	**73.23**	**69.91**	**63.41**	**72.53**	**67.85**	**65.92**	**73.75**	**66.10**

Sequence 2	ED_1	ED_2	ED_3	ED_4	RE_1	RE_2	RE_3	RE_4	NER_1	NER_2	NER_3	NER_4
SeqLoRA	88.74	48.86	34.87	23.72	15.34	15.70	15.70	14.85	10.46	14.73	19.49	17.44
Joint-training	88.74	85.57	82.51	77.76	81.45	81.8	82.19	82.73	83.41	85.92	85.85	86.43
L2P$^{\triangle}$	88.43	53.18	45.28	40.79	53.62	48.39	48.09	49.75	54.05	53.19	55.32	53.99
ProgPrompt$^{\triangle}$	88.74	51.09	48.59	40.03	41.70	47.04	46.55	46.95	56.60	54.21	56.12	54.43
NoRGa$^{\triangle}$	89.38	52.05	46.06	40.89	55.62	59.24	57.29	57.94	63.29	61.33	60.48	57.37
O-LoRA	88.74	50.94	45.43	40.10	41.13	44.05	44.44	44.93	54.41	51.64	56.34	54.23
SAPT	88.52	55.80	44.89	36.31	50.19	51.68	46.24	47.31	41.80	50.52	47.81	47.66
GPT-4o	30.65	30.51	26.40	22.20	25.28	25.29	24.06	23.01	28.65	30.59	37.06	36.95
MoLE-CIE	**90.35**	**59.77**	**51.06**	**50.40**	**62.72**	**63.28**	**65.22**	**68.72**	**70.18**	**68.91**	**67.44**	**67.29**

Table 1: Accuracy comparison between MoLE-CIE and competitors on all current and previous splits. The best and second-best scores except for joint-training are marked in **bold** and with underline, respectively. $^{\triangle}$ denotes the methods replaced with LoRA.

task, denoted by $ED_{1,...,4}, RE_{1,...,4}, NER_{1,...,4}$. As the majority of types have more than 10 training instances, we set the memory size to 10. To simulate various continual IE scenarios, we design the following two experiment sequences as defined in Section 3. The sequence of extraction tasks is arranged in order of difficulty, from hardest to easiest.

- **Sequence 1.** In this sequence, different types of tasks arrive in a polling order. In each cycle, the model should sequentially learn all IE tasks. Thus, the task sequence is $\{ED_1, RE_1, NER_1\} \rightarrow \cdots \rightarrow \{ED_4, RE_4, NER_4\}$.

- **Sequence 2.** In this sequence, the model should learn all splits of the current IE task before learning the next one. Therefore, the task sequence is $\{ED\}_{i=1}^{4} \rightarrow \{RE\}_{i=1}^{4} \rightarrow \{NER\}_{i=1}^{4}$.

Competing methods. We compare our MoLE-CIE with eight competitors: (1) **SeqLoRA** simply trains a LoRA on the current training set without any memory. (2) **Joint-training** trains the model on the current and all previous training data for each new task. It represents the behavior of retraining and can be viewed as the upper bound in continual IE. (3) **L2P** (Wang et al., 2022b) adapts the model to sequential tasks by dynamically selecting and learning prompts from a fixed prompt pool. (4) **ProgPrompt** (Razdaibiedina et al., 2023) learns a new soft prompt for each task and sequentially concatenates it with the previously learned prompts. (5) **NoRGa** (Le et al., 2024a) proposes non-linear residual gates to gate the prefix prompt in continual learning. (6) **O-LoRA** (Wang et al., 2023a) learns tasks in different LoRA subspaces that are kept orthogonal to each other. (7) **SAPT** (Zhao et al., 2024) aligns the LoRA learning and selection by shared attention weights. (8) **GPT-4o** (Hurst et al., 2024) is employed as a powerful zero-shot IE model. The test prompt templates for GPT-4o are listed in Appendix C.

Hyperparameter setting and environment. We run all experiments on an X86 server with two Intel Xeon Gold 6326 CPUs, 512 GB memory, four NVIDIA RTX A6000 GPU cards, and Ubuntu 20.04 LTS. Due to our limited computational resources, we cannot afford the high cost of full parameter tuning. Therefore, we do not compare with continual learning methods (Wang et al., 2024; He et al., 2024), which are based on full parameter tuning in our experiments.

For our MoLE-CIE and all competitors, we use AdamW optimizer to train the model with the learn-

Hyperparameters	Values
$\alpha, \beta, \gamma, \delta$	$0.75, 0.25, 1, 1$
M (memory size)	10
Num. of all experts	12
Num. of LoRA experts	8
Num. of activated LoRA experts	2
Num. of task experts for each IE task	1
Num. of activated task experts	1

Table 2: Hyperparameter setting in our model.

ing rate of 5e-5 for LLaMA-3.1-8B (Grattafiori et al., 2024). The batch size is set to 4 during training, while 32 during prediction. r and α are set to 4 and 32, respectively. The number of training epochs and the gradient accumulation steps are set to 2 and 4, respectively. For the unique hyperparameters of all competitors, we follow their original papers. For the unique hyperparameters of MoLE-CIE, we list them in Table 2.

For a fair comparison, we replay the data for each continual learning model and use the same backbone model LLaMA-3.1 (Grattafiori et al., 2024). We also replace prompt tuning in L2P, ProgPrompt, and NoRGa to LoRA to compare them fairly with other methods.

Evaluation metrics. (1) Following previous works (Wang et al., 2023a; Zhao et al., 2024), we use **accuracy** to measure the proportion of correctly predicted samples out of the total number of samples. (2) **Backward transfer** (BWT) and **forward transfer** (FWT) (Lopez-Paz and Ranzato, 2017) are two widely-used metrics to measure the knowledge transferability and how well the model alleviates catastrophic forgetting. BWT measures the influence of learning on previous splits, while FWT measures the influence of learning on future splits. BWT scores are defined as $\frac{1}{K-1}\sum_{i=1}^{K-1}(a_{K,i} - a_{i,i})$, and FWT scores are defined as $\frac{1}{K-1}\sum_{i=2}^{K}(a_{i-1,i} - a_{0,i})$, where K is the number of splits. $a_{i,j}$ measures the test accuracy of the split j after training the split i. Note that BWT scores are negative due to catastrophic forgetting. (3) **F.Ra** (Chaudhry et al., 2018) measures the forgetting rate of previous splits, which is defined as $\frac{1}{K-1}\sum_{i=1}^{K-1}\left(\max_{t=i}^{k-1} a_{t,i} - a_{K,i}\right)$. A lower F.Ra score indicates a better performance.

5.2 Results and Analyses

Main results. Table 1 presents the main results of MoLE-CIE and the competitors in the two sequences. Note that the results on ED_1 are based on each model itself without any continual learning.

Our model MoLE-CIE is highly effective in addressing the challenge of catastrophic forgetting and knowledge transfer in continual IE. After training on all splits, compared with the best competitor NoRGa, MoLE-CIE gains 14.32% and 9.92% improvement of accuracy in Sequence 1 and Sequence 2, respectively. The significant gap demonstrates that the token-level mixture of LoRA experts has a strong capacity to retain task-specific knowledge across different training sequences. We also conclude that our model can be effectively applied to the datasets from different domains of multiple IE tasks. In Sequence 1, MoLE-CIE maintains excellent and stable performance in each cycle, which shows our model's superior knowledge transferability with the LoRA experts across multiple IE tasks. In Sequence 2, MoLE-CIE performs smoothly after adding different splits for the same IE task, especially on RE and NER tasks, validating that MoLE-CIE is also effective in mitigating forgetting of task-specific knowledge by task experts. Furthermore, the results of MoLE-CIE are close to joint-training, which is regarded as the upper bound with re-training in continual IE. Differently, our model is more cost-effective without re-training. Regarding GPT-4o, although GPT-4o is a powerful model with a large number of parameters, it still performs poorly in the continual IE scenario. This suggests that contemporary LLMs are not specialized in continual IE and cannot replace continual learning models yet.

Ablation study. To evaluate the effectiveness of each module in our model, we conduct an ablation study. The final results are listed in Table 4, and the complete results are listed in Appendix D. Specifically, for "w/o LoRA experts", we remove all LoRA experts. For "w/o Task experts", we remove all task experts. For "w/o IE experts", we remove the IE experts. For "w/o Router KD", we disable the router knowledge distillation in gate reflection. For "w/o Keys KD", we disable the keys knowledge distillation in gate reflection. For "Token $\rightarrow$ sentence", we switch the expert selection in MoLE-CIE from the token level to the sentence level. We observe that all modules are effective.

Knowledge forgetting. We analyze the performance of knowledge forgetting in continual IE. Table 3 and the F.Ra metric in Table 5 present the results. In Table 3, after training on all splits, our MoLE-CIE consistently maintains the highest per-

Sequence 1	ED_1	RE_1	NER_1	ED_2	RE_2	NER_2	ED_3	RE_3	NER_3	ED_4	RE_4	NER_4	Avg.
SeqLoRA	0.00	0.00	0.76	0.00	0.79	4.11	0.00	0.19	37.82	0.13	16.11	94.58	12.87
L2P$^{\triangle}$	35.96	58.86	15.42	42.78	65.66	35.53	47.13	52.45	44.25	50.34	67.98	91.43	50.65
ProgPrompt$^{\triangle}$	36.71	61.78	21.18	40.54	62.58	32.95	39.62	58.65	53.45	45.99	70.01	88.52	50.99
NoRGa$^{\triangle}$	40.67	66.59	15.18	47.98	71.08	27.03	51.21	60.15	44.49	57.43	68.22	90.63	53.39
O-LoRA	29.67	60.26	22.61	39.02	55.33	32.21	46.04	52.29	57.08	48.04	68.05	91.18	50.15
SAPT	31.71	57.95	4.92	49.78	65.65	20.64	33.65	56.74	29.27	47.66	63.30	94.00	46.27
MoLE-CIE	**52.62**	**73.56**	**35.77**	**54.48**	**79.99**	**48.48**	**67.51**	**79.15**	**68.52**	**61.05**	**85.30**	**95.76**	**66.85**
Sequence 2	ED_1	ED_2	ED_3	ED_4	RE_1	RE_2	RE_3	RE_4	NER_1	NER_2	NER_3	NER_4	Avg.
SeqLoRA	0.00	0.00	0.08	0.00	0.00	0.23	0.17	0.39	0.13	8.57	47.61	94.03	12.60
L2P$^{\triangle}$	44.53	39.91	48.41	48.42	60.67	67.02	60.90	68.85	15.27	39.90	60.91	89.85	53.72
ProgPrompt$^{\triangle}$	40.70	49.51	53.81	56.82	66.41	70.01	59.70	74.44	17.73	35.00	50.81	91.57	55.54
NoRGa$^{\triangle}$	41.78	49.73	50.32	56.87	65.03	71.25	59.03	73.68	14.88	**49.72**	61.71	92.03	57.16
O-LoRA	37.26	44.74	43.84	46.12	60.42	68.07	57.07	67.55	28.10	34.04	68.13	90.19	53.79
SAPT	37.79	34.99	47.69	46.35	79.09	62.46	35.55	51.23	15.55	21.93	49.52	93.32	47.95
MoLE-CIE	**58.71**	**54.84**	**60.48**	**58.86**	**82.54**	**80.41**	**71.01**	**81.11**	**52.79**	44.29	**74.78**	**95.60**	**67.95**

Table 3: Accuracy comparison on each individual split after training on all splits. We did not report the results of joint-training and GPT-4o, because they do not involve knowledge forgetting.

	Sequence 1	**Sequence 2**
MoLE-CIE (full)	**66.10**	**67.29**
w/o LoRA experts	55.81	58.33
w/o Task experts	59.80	63.48
w/o IE experts	64.36	66.09
w/o Router KD	64.74	66.61
w/o Keys KD	64.21	65.70
Token $\rightarrow$ Sentence	61.10	63.31

Table 4: Accuracy of ablation study after training on all splits. "KD" is abbreviated for knowledge distillation.

Sequence 1	**FWT** $\uparrow$	**BWT** $\uparrow$	**F.Ra** $\downarrow$
SeqLoRA	1.29	−82.21	89.68
L2P$^{\triangle}$	3.20	−41.83	45.82
ProgPrompt$^{\triangle}$	4.26	−38.96	41.84
NoRGa$^{\triangle}$	4.49	−37.57	40.99
O-LoRA	3.63	−40.68	44.38
SAPT	4.99	−45.38	49.50
MoLE-CIE	**6.46**	**−26.71**	**29.13**
Sequence 2	**FWT** $\uparrow$	**BWT** $\uparrow$	**F.Ra** $\downarrow$
SeqLoRA	2.70	−82.48	89.98
L2P$^{\triangle}$	2.50	−35.66	40.34
ProgPrompt$^{\triangle}$	4.99	−34.12	37.22
NoRGa$^{\triangle}$	5.03	−32.33	35.53
O-LoRA	2.19	−34.88	38.05
SAPT	4.56	−44.16	48.17
MoLE-CIE	**7.28**	**−25.82**	**28.16**

Table 5: FWT, BWT, and F.Ra scores.

formance on almost all splits. Particularly, on the first split ED_1, MoLE-CIE outperforms the best competitor by 11.95% and 14.18% in Sequence 1 and Sequence 2, respectively. In Table 5, MoLE-CIE also obtains the lowest F.Ra, demonstrating its strong ability to resist knowledge forgetting.

Knowledge transfer. We use two widely-used metrics, FWT and BWT, to measure the knowledge transferability of each model in continual IE. FWT measures the knowledge transferability from previous splits to new splits, while BWT measures the ability from new splits to previous splits. Our MoLE-CIE achieves the optimal results for both metrics, which shows the superiority of our model in knowledge transfer among IE tasks. Furthermore, MoLE-CIE outperforms all competitors in terms of BWT by a large margin. It verifies that MoLE-CIE can effectively use the knowledge of post-learning IE tasks to update the knowledge of previously learned IE tasks, in contrast to related works that freeze the learned knowledge.

Expert selection. To investigate the effect of expert selection by the LoRA router and task keys, we visualize the frequency of the expert selection for each split after training all splits. From Figure 2, we can find that: (1) For task expert selection, MoLE-CIE assigns task experts that match the IE tasks for the most tokens, which shows the effectiveness of task experts in preserving the knowledge of IE tasks to mitigate catastrophic forgetting. MoLE-CIE also assigns task experts that do not match the tasks for some tokens, such as assigning task expert E_{NER} to some tokens in the RE task. This indicates that MoLE-CIE selects all task experts for each IE task at the token level, effectively integrating the knowledge from various IE tasks.

(2) For LoRA expert selection, as the number of activated LoRA experts is set to 2, we can observe

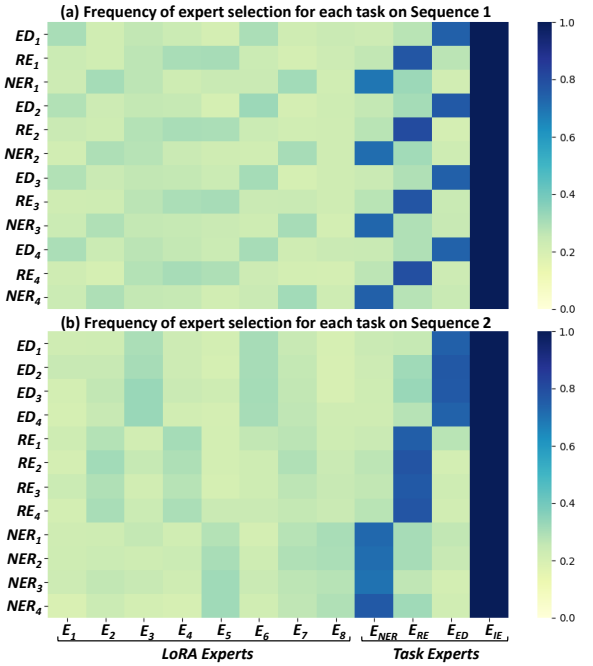

Figure 2: Heat maps of frequency of expert selection for each task on (a) Sequence 1 and (b) Sequence 2.

in Figure 2 that MoLE-CIE assigns two different LoRA experts to the majority of tokens for each IE task. For example, in Figure 2(a), most tokens for the ED task are assigned E_1 and E_2, while for RE are E_4 and E_5. This suggests that these LoRA experts store task-related knowledge and can be appropriately assigned to the corresponding tasks, further retaining task-specific knowledge. Furthermore, all LoRA experts are selected for each IE task. This shows that each LoRA expert learns different specific knowledge and uses it in different IE tasks.

(3) In Figure 2(a), although each new task learned is different, MoLE-CIE can still assign the correct experts, showing its strong adaptability to new tasks. In Figure 2(b), MoLE-CIE can consistently and stably assign LoRA experts for the same IE task, indicating it can continuously update the knowledge of corresponding LoRA experts.

6 Conclusion

In this paper, we define continual IE and design a novel model MoLE-CIE. We propose the mixture of LoRA experts and gate reflection to mitigate catastrophic forgetting and facilitate knowledge transfer. Experiment results on three benchmark IE tasks show that MoLE-CIE achieves superior accuracy. The results also verify its effectiveness in knowledge reservation and transfer.

Acknowledgments

This work was funded by National Natural Science Foundation of China (No. 62272219).

Limitations

Our model may have two limitations: (1) It focuses on the three most typical IE tasks, namely NER, RE, and ED. However, there exist other IE tasks, such as sentiment analysis. While it is feasible to run these tasks with MoLE-CIE, we have not carried out in-depth evaluation on them yet. (2) Our model uses a PEFT method, which may result in the loss of accuracy compared to the continual learning algorithms employing full-parameter fine-tuning. Our current hardware environment limits our experiments on this.

References

Peggy M Andersen, Philip J Hayes, Steven P Weinstein, Alison K Huettner, Linda M Schmandt, and Irene Nirenburg. 1992. Automatic extraction of facts from press releases to generate news stories. In *ANLP*, pages 170–177.

Amos Azaria, Rina Azoulay, and Shulamit Reches. 2024. ChatGPT is a remarkable tool—for experts. *Data Intelligence*, pages 240–296.

Pengfei Cao, Yubo Chen, Jun Zhao, and Taifeng Wang. 2020. Incremental event detection via knowledge consolidation networks. In *EMNLP*, pages 707–717.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, pages 532–547.

Huajun Chen. 2024. Large knowledge model: Perspectives and challenges. *Data Intelligence*, pages 587–620.

Li Cui, Deqing Yang, Jiaxin Yu, Chengwei Hu, Jiayang Cheng, Jingjie Yi, and Yanghua Xiao. 2021. Refining sample embeddings with relation prototypes to enhance continual relation extraction. In *ACL*, pages 232–243.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. *NeurIPS*.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *LREC*.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *EMNLP*, pages 671–683.

Robert M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3:128–135.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and Vs Subrahmanian. 2025. MoLA: MoE LoRA with layer-wise expert allocation. In *Findings of NAACL 2025*, pages 5097–5112.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yucan Guo, Zixuan Li, Xiaolong Jin, Yantao Liu, Yutao Zeng, Wenxuan Liu, Xiang Li, Pan Yang, Long Bai, Jiafeng Guo, and Xueqi Cheng. 2024. Retrieval-augmented code generation for universal information extraction. In *NLPCC*, pages 30–42.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *EMNLP*, pages 4803–4809.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. 2024. SEEKR: Selective attention-guided knowledge retention for continual learning of large language models. In *EMNLP*, pages 3254–3266.

Yuzhao Heng, Chunyuan Deng, Yitong Li, Yue Yu, Yinghao Li, Rongzhi Zhang, and Chao Zhang. 2024. ProgGen: Generating named entity recognition datasets step-by-step with self-reflexive large language models. In *Findings of ACL*, pages 15992–16030.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *NAACL, Companion Volume: Short Papers*, pages 57–60.

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 *ICLR*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. GPT-4o system card. *arXiv preprint arXiv:2410.21276*.

Minh Le, Tien Ngoc Luu, An Nguyen The, Thanh-Thien Le, Trang Nguyen, Tung Thanh Nguyen, Linh Ngo Van, and Thien Huu Nguyen. 2025. Adaptive prompting for continual relation extraction: A within-task variance perspective. *AAAI*.

Minh Le, An Nguyen, Huy Nguyen, Trang Nguyen, Trang Pham, Linh Van Ngo, and Nhat Ho. 2024a. Mixture of experts meets prompt-based continual learning. *NeurIPS*, pages 119025–119062.

Thanh-Thien Le, Viet Dao, Linh Nguyen, Thi-Nhung Nguyen, Linh Ngo, and Thien Nguyen. 2024b. SharpSeq: Empowering continual event detection through sharpness-aware sequential-task learning. In *NAACL*, pages 3632–3644.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, pages 3045–3059.

Junpeng Li, Zixia Jia, and Zilong Zheng. 2023. Semi-automatic data enhancement for document-level relation extraction with distant supervision from large language models. In *EMNLP*, pages 5495–5505.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *ACL*, pages 7999–8009.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *NeurIPS*, 30.

Shawn N Murphy, Griffin Weber, Michael Mendis, Vivian Gainer, Henry C Chueh, Susanne Churchill, and Isaac Kohane. 2010. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17:124–130.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *ICLR*.

Mark Bishop Ring. 1994. *Continual learning in reinforcement environments*. Ph.D. thesis, University of Texas at Austin.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2024. GoLLIE: Annotation guidelines improve zero-shot information-extraction. In *ICLR*.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *ACL*, pages 2895–2905.

Sebastian Thrun and Tom M. Mitchell. 1995. Lifelong robot learning. *Robotics and Autonomous Systems*, 15:25–46.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. In *EMNLP*, pages 3534–3547.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019. Sentence embedding alignment for lifelong relation extraction. In *NAACL*, pages 796–806.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023a. Orthogonal subspace learning for language model continual learning. In *Findings of EMNLP*, pages 10658–10671.

Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. 2020. MAVEN: A massive general domain event detection dataset. In *EMNLP*, pages 1652–1671.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. 2024. InsCL: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In *NAACL*, pages 663–677.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022a. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *EMNLP*, pages 5085–5109.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, and Wenqiu Zeng. 2023b. Rehearsal-free continual language learning via efficient parameter isolation. In *ACL*, pages 10933–10946.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022b. Learning to prompt for continual learning. In *CVPR*, pages 139–149.

Zitao Wang, Xinyi Wang, and Wei Hu. 2023c. Continual event extraction with semantic confusion rectification. In *EMNLP*, pages 11945–11955.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 35:24824–24837.

Pengfei Yu, Heng Ji, and Prem Natarajan. 2021. Lifelong event detection with knowledge transfer. In *EMNLP*, pages 5278–5290.

Yahan Yu, Zhengdong Yang, Fei Cheng, and Chenhui Chu. 2025. Semantic-retention attack for continual named entity recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing*.

Yahan Yu, Duzhen Zhang, Xiuyi Chen, and Chenhui Chu. 2024. Flexible weight tuning and weight fusion strategies for continual named entity recognition. In *Findings of ACL*, pages 1351–1358.

Duzhen Zhang, Wei Cong, Jiahua Dong, Yahan Yu, Xiuyi Chen, Yonggang Zhang, and Zhen Fang. 2023a. Continual named entity recognition without catastrophic forgetting. In *EMNLP*, pages 8186–8197.

Ruoyu Zhang, Yanzeng Li, Yongliang Ma, Ming Zhou, and Lei Zou. 2023b. LLMaAA: Making large language models as active annotators. In *Findings of EMNLP*, pages 13088–13103.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *EMNLP*, pages 35–45.

Kang Zhao, Hua Xu, Jiangong Yang, and Kai Gao. 2022. Consistent representation learning for continual relation extraction. In *Findings of ACL*, pages 3402–3411.

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 *ACL*, pages 11641–11661.

Wenzheng Zhao, Yuanning Cui, and Wei Hu. 2023. Improving continual relation extraction by distinguishing analogous semantics. In *ACL*, pages 1162–1175.

Junhao Zheng, Zhanxian Liang, Haibin Chen, and Qianli Ma. 2022. Distilling causal effect from miscellaneous other-class for continual named entity recognition. In *EMNLP*, pages 3602–3615.

Mengna Zhu, Kaisheng Zeng, JibingWu JibingWu, Lihua Liu, Hongbin Huang, Lei Hou, and Juanzi Li. 2024. LC4EE: LLMs as good corrector for event extraction. In *Findings of ACL*, pages 12028–12038.

A Dataset Statistics and Splits

In this section, we introduce how we split the datasets used in our experiments.

For the event detection task, we follow (Lin et al., 2020) to pre-process the dataset for *ACE05-EN+* (Doddington et al., 2004). The original development set and test set miss several event types, and the number of instances in the test set is much

Tasks	Splits	Types	Instances		
			Training	Dev.	Test
ED	1	52	18,148	2,050	5,112
	2	46	17,754	2,000	4,675
	3	51	19,924	2,241	5,367
	4	52	18,194	2,047	4,696
RE	1	32	10,337	3,037	3,037
	2	29	10,127	3,013	3,013
	3	31	10,143	3,012	3,012
	4	28	10,282	3,033	3,033
NER	1	8	19,891	2,786	3,131
	2	10	20,840	3,771	5,402
	3	8	31,498	6,139	9,406
	4	8	21,610	3,975	4,885

Table 6: Dataset statistics.

smaller than that in the development set. Following (Yu et al., 2021), we re-split the training set for the development set and test set. We combine the original development set and test set as a new test set. Then, we randomly sample 10% of instances from the training set as a new development set. For the new test set, if the number of instances for a type is less than 10% of the total instances for this type, we randomly sample instances from the training set and remove them from the training set to make up for the difference. We do not include "*Justice:Pardon*" in both the development set and test set, as it only has two instances in the entire dataset. For *MAVEN* (Wang et al., 2020), we follow (Wang et al., 2023c) to take the original development set as the test set and re-split the training set.

For the relation extraction task, we follow (Cui et al., 2021; Zhao et al., 2022) to use 80 relations and each with 700 samples for *Few-Rel* (Han et al., 2018). We allocate 420 samples to the training set, 140 samples to the development set, and 140 samples to the test set. For *TACRED*, we follow (Cui et al., 2021; Zhao et al., 2022) to remove the samples of "*no_relation*".

For the named entity recognition task, we use the origin dataset for *OntoNotes-5.0* (Hovy et al., 2006) and *i2b2* (Murphy et al., 2010).

For the partition of each dataset, we ensure that the label and instance counts for each split are as consistent as possible. After partitioning, we merge the split of each dataset based on the task type. The statistics of the datasets after merging are presented in Table 6.

Event detection prompt:
Definition:
Please extract the information (such as named entity types, event types, or relation types) from the sentences and words below.
Input:
Sentence: {input sentence} Word1: {word1}
Output:

Relation extraction prompt:
Definition:
Please extract the information (such as named entity types, event types, or relation types) from the sentences and words below.
Input:
Sentence: {input sentence} Word1: {word1} Word2: {word2}
Output:

Named entity recognition prompt:
Definition:
Please extract the information (such as named entity types, event types, or relation types) from the sentences and words below.
Input:
Sentence: {input sentence} Word1: {word1}
Output:

Table 7: Instructions for different tasks.

B Task Instructions

In Table 7, we provide the prompt templates used for training and inference. Based on the task instructions, it can be observed that:

(1) The model is not informed of the specific task information during inference. This prevents the leakage of task information, making the continual IE problem more challenging and generalizable.

(2) Due to the absence of task information, existing task-specific continual learning methods (Wang et al., 2023c; Le et al., 2025; Zhao et al., 2023; Le et al., 2024b; Zhang et al., 2023a; Yu et al., 2024) for IE cannot be applied to the continual IE task. For continual learning methods with LLMs, such instructions place higher demands on the model's selection module. Thanks to the token-level selection module, MoLE-CIE effectively handles this challenge and achieves the best performance.

C Test Prompts and Results for GPT-4o

Table 8 depicts the test prompts used by GPT-4o. Considering that GPT-4o requires specific task definitions to guide its generation, we provide it with detailed task information in the test prompts. Therefore, compared to the task instructions in Appendix B, the test prompts for GPT-4o are simpler.

Table 9 reports the test results on each extraction task, comparing with the performance of MoLE-

CIE trained on all splits from Sequences 1 and 2 with that of GPT-4o. From the results, we can find that GPT-4o performs not well in the continual IE scenario, especially on the ED and RE tasks. This suggests that continual learning is still necessary in the field of IE even after the emergence of powerful LLMs.

Event detection prompt: Please determine the event type that appears in the sentence based on the trigger words. Sentence: {input sentence} Trigger words: {trigger word} Please select only a type from the following options: Event types: {event types}
Relation extraction prompt: Please determine the relation type between the two entities in the sentence. Sentence: {input sentence} Entity1: {entity1} Entity2: {entity2} Please select only a type from the following options: Relation types: {relation types}
Named entity recognition prompt: Please determine the named entity type based on the sentence and the entity. Sentence: {input sentence} Entity: {entity} Please select only a type from the following options: Entity types: {entity types}

Table 8: Test prompts for GPT-4o.

	ED	*RE*	*NER*	*IE*
GPT-4o	22.20	24.32	56.47	36.95
MoLE-CIE (Seq. 1)	58.92	79.50	62.13	66.10
MoLE-CIE (Seq. 2)	58.22	78.77	66.87	67.29

Table 9: Accuracy comparison between MoLE-CIE and GPT-4o given specific task definitions.

D Complete Results of Ablation Study

In Table 10, we present the complete results of the ablation study. From the results, we can find that:

(1) After training on each split, all models with removed or replaced modules exhibit a performance degradation. This further demonstrates the effectiveness of each module in MoLE-CIE.

(2) When MoLE-CIE is switched from token-level to sentence-level, the model's performance declines. This highlights the importance of the token-level expert selection module to MoLE-CIE.

E Analysis of Number of LoRA Experts

In Table 11, we conduct an analysis with different numbers n of LoRA experts. From the results, we can find that:

(1) The accuracy of MoLE-CIE degrades in the two sequences with $n = 4$ and $n = 12$. This suggests the rationale of setting the number of total experts equal to the number of training splits.

(2) When $n = 4$, MoLE-CIE does not have enough LoRA experts to learn the knowledge across different tasks, leading to knowledge conflicts within the same LoRA expert.

(3) When $n = 12$, the model distributes the knowledge across too many LoRA experts, causing that the model lacks the ability to fully utilize knowledge with the same number of activated LoRA experts.

F Analysis of Individual and All Tasks Training

We conduct experiments to compare the MoLE-CIE's performance after only training on each individual task with that after training on all tasks in Sequences 1 and 2. Table 12 presents the MoLE-CIE's test results on each individual task under different training strategies.

From the table, we can observe that the results of the models trained on all tasks are better than those of the model trained on any individual task. There is shared knowledge between the sub-tasks in IE, which can promote learning between each other. This is exactly one of our motivations.

G Analysis of Random Sequences

To verify the robustness of our model across different sequences, we conduct an analysis with random sequences. We randomly generate three task sequences and present them in Table 13. We train and evaluate all models on the three sequences, and average the accuracy over the current and previous splits. The results are presented in Table 14. From the results, we can find that MoLE-CIE consistently maintains the best performance. These results demonstrate the strong robustness of MoLE-CIE on different randomly sequences.

H Case Study

In Table 15, we select two instances in the test set after training all splits in Sequence 1. From these two cases, we can observe that:

(1) In Case 1, the best competitor NoRGa (Le et al., 2024a) has forgotten the knowledge from ED_2, resulting in incorrect output. In contrast, MoLE-CIE successfully preserves the knowledge from ED_2 through task experts and assigned the

Sequence 1	ED_1	RE_1	NER_1	ED_2	RE_2	NER_2	ED_3	RE_3	NER_3	ED_4	RE_4	NER_4
MoLE-CIE (full)	**90.35**	**85.45**	**84.49**	**70.68**	**73.23**	**69.91**	**63.41**	**72.53**	**67.85**	**65.92**	**73.75**	**66.10**
w/o LoRA experts	87.98	71.32	73.66	62.06	63.77	63.51	53.51	62.25	58.11	56.13	63.75	55.81
w/o Task experts	90.07	78.44	80.78	69.72	71.52	64.54	59.66	69.33	61.78	59.91	68.66	59.80
w/o IE experts	89.78	83.74	82.78	69.15	71.96	67.92	62.28	70.86	64.78	62.31	70.88	64.36
w/o Router KD	90.35	83.83	82.65	69.92	71.42	68.08	61.43	70.64	65.68	63.77	72.04	64.74
w/o Keys KD	90.35	83.60	81.19	69.50	71.77	68.27	61.13	71.11	65.23	64.33	71.70	64.21
Token $\rightarrow$ Sentence	88.58	74.94	77.73	67.12	70.78	66.65	59.37	68.04	61.48	59.98	68.87	61.11
Sequence 2	ED_1	ED_2	ED_3	ED_4	RE_1	RE_2	RE_3	RE_4	NER_1	NER_2	NER_3	NER_4
MoLE-CIE (full)	**90.35**	**59.77**	**51.06**	**50.40**	**62.72**	**63.28**	**65.22**	**68.72**	**70.18**	**68.91**	**67.44**	**67.29**
w/o LoRA experts	87.98	52.71	46.01	40.30	57.71	48.79	49.83	57.61	54.37	55.40	57.98	58.33
w/o Task experts	90.07	54.14	46.44	41.81	58.85	58.69	59.66	60.14	63.71	61.48	62.93	63.48
w/o IE experts	89.78	58.09	49.61	48.13	60.16	61.42	61.25	64.86	69.24	67.13	65.84	66.09
w/o Router KD	90.35	57.93	50.09	48.64	60.57	60.35	63.73	66.43	68.95	66.89	65.33	66.61
w/o Keys KD	90.35	58.06	49.83	42.62	60.52	61.28	63.96	62.39	69.25	67.93	65.86	65.70
Token $\rightarrow$ Sentence	88.58	57.56	44.38	39.58	57.87	60.55	59.82	58.28	65.11	62.13	62.31	63.31

Table 10: Accuracy of ablation study on all current and previous splits. "KD" is abbreviated for knowledge distillation.

Sequence 1	ED_1	RE_1	NER_1	ED_2	RE_2	NER_2	ED_3	RE_3	NER_3	ED_4	RE_4	NER_4
$n = 8$ (default)	**90.35**	**85.45**	**84.49**	**70.68**	**73.23**	**69.91**	**63.41**	**72.53**	**67.85**	**65.92**	**73.75**	**66.10**
$n = 4$	89.44	84.61	82.48	68.04	73.01	68.05	61.93	70.71	65.55	63.41	71.02	64.00
$n = 12$	89.71	81.53	81.70	69.72	71.82	70.37	61.45	70.61	67.67	65.55	72.74	65.52
Sequence 2	ED_1	ED_2	ED_3	ED_4	RE_1	RE_2	RE_3	RE_4	NER_1	NER_2	NER_3	NER_4
$n = 8$ (default)	**90.35**	**59.77**	**51.06**	**50.40**	**62.72**	**63.28**	**65.22**	**68.72**	**70.18**	**68.91**	**67.44**	**67.29**
$n = 4$	89.44	58.13	49.32	44.12	61.30	61.33	63.33	64.15	69.12	66.99	64.83	66.00
$n = 12$	89.71	58.95	50.06	49.70	60.18	61.77	67.71	67.18	74.75	67.94	67.10	66.37

Table 11: Accuracy w.r.t. the number n of LoRA experts in the two sequences.

MoLE-CIE after training on	*ED*	*RE*	*NER*
Each individual task	50.40	73.11	59.09
Sequence 1 (all tasks)	58.91	79.50	62.13
Sequence 2 (all tasks)	58.22	78.76	66.86

Table 12: Results of MoLE-CIE on individual tasks and all tasks training.

Random	**Task sequences**
1	$RE_1 \rightarrow EE_2 \rightarrow RE_2 \rightarrow EE_4 \rightarrow RE_4 \rightarrow NER_4 \rightarrow EE_1 \rightarrow NER_1 \rightarrow RE_3 \rightarrow NER_2 \rightarrow NER_3 \rightarrow EE_3$
2	$NER_4 \rightarrow NER_1 \rightarrow EE_2 \rightarrow RE_3 \rightarrow RE_2 \rightarrow RE_4 \rightarrow NER_2 \rightarrow EE_1 \rightarrow EE_3 \rightarrow EE_4 \rightarrow RE_1 \rightarrow NER_3$
3	$EE_1 \rightarrow NER_1 \rightarrow EE_3 \rightarrow NER_4 \rightarrow EE_2 \rightarrow RE_3 \rightarrow NER_3 \rightarrow RE_4 \rightarrow EE_4 \rightarrow NER_2 \rightarrow RE_1 \rightarrow RE_2$

Table 13: Three different random task sequences.

correct task expert E_{ED} to this instance, resulting in the correct answer. This shows the superiority of MoLE-CIE in mitigating catastrophic forgetting.

(2) In Case 2, NoRGa incorrectly uses knowledge from the NER task and chooses the wrong knowledge to answer this instance. This suggests that the sentence-level selection module of NoRGa is unable to distinguish knowledge from different IE tasks effectively, leading to forgetting. It also highlights the model's weakness in knowledge transfer among IE tasks. Thanks to the token-level mixture of LoRA experts module, MoLE-CIE successfully uses the correct experts for this instance and provides the correct answer. This demonstrates that MoLE-CIE can effectively distinguish and utilize knowledge from different IE tasks, showcasing its superiority in mitigating catastrophic forgetting and facilitating knowledge transfer.

In Table 16, we also provide two bad cases of our model to analyze the limitations from a comprehensive view.

In bad case 1, the semantics of "Scrutiny" and "Scouring" are very similar, with only subtle differ-

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
SeqLoRA	91.89	44.83	41.68	25.39	24.76	17.53	23.67	20.34	20.33	16.63	17.05	20.21
Joint-training	91.89	94.28	91.39	89.86	88.88	90.17	89.53	89.82	87.11	86.75	87.55	88.03
L2P$^{\triangle}$	92.35	70.38	65.78	60.76	52.14	60.24	52.51	52.68	49.57	42.94	46.31	45.79
ProgPrompt$^{\triangle}$	91.89	73.57	67.66	60.47	50.42	54.93	50.85	51.43	45.72	37.21	39.83	43.08
NoRGa$^{\triangle}$	92.37	72.10	71.31	67.33	61.54	62.75	57.34	62.28	50.76	48.96	51.83	52.94
O-LoRA	91.89	69.39	70.71	66.37	55.95	60.21	57.30	56.92	44.18	44.22	43.61	47.96
SAPT	92.51	69.28	53.53	52.56	54.06	51.84	48.33	47.67	41.55	41.46	42.91	40.05
GPT-4o	36.02	48.04	39.73	34.81	32.89	31.42	35.53	36.17	33.65	33.54	36.05	36.95
MoLE-CIE	**93.12**	**79.33**	**73.61**	**70.08**	**71.44**	**75.70**	**70.79**	**73.77**	**66.75**	**68.35**	**70.81**	**71.64**

Table 14: Average accuracy comparison between MoLE-CIE and competitors on all current and previous splits across three random task sequences. The best and second-best scores except for joint-training are marked in **bold** and with underline, respectively. $^{\triangle}$ denotes the methods replaced with LoRA.

Case 1:
Definition: Please extract the information (such as named entity types, event types, or relation types) from the sentences and words below.
Input: Sentence: In august 1944, the royal navy conducted operation goodwood, four more carrier raids against "tirpitz" which also failed and the task of sinking the battleship was transferred to the royal air force. Word1: raids.
Output:
Ground truth: Attack (The type in ED_2)

Output of NoRGa (Le et al., 2024a):
Patrolling (The type in ED_1)
Output of MoLE-CIE:
Attack (The type in ED_2)

Case 2:
Definition: Please extract the information (such as named entity types, event types, or relation types) from the sentences and words below.
Input: Sentence: The operation ultimately failed when FRELIMO forces regrouped and thrusted further south in the province of Tete, opening a new front and overstretching the Portuguese Army. Word1: operation.
Output:
Ground truth: Military_operation (The type in ED_4)

Output of NoRGa (Le et al., 2024a):
GPE (The type in NER_4)
Output of MoLE-CIE:
Military_operation (The type in ED_4)

Table 15: Case study of Sequence 1.

Bad case 1:
Definition: Please extract the information (such as named entity types, event types, or relation types) from the sentences and words below..
Input: Sentence: The operation ended a nearly 10-year search for bin Laden, following his role in the September 11 attacks on the United States. Word1: search.
Output:
Ground truth: Scrutiny (The type in *EE*_3)

Output of MoLE-CIE:
Scouring (The type in *EE*_2)

Bad case 2:
Definition: Please extract the information (such as named entity types, event types, or relation types) from the sentences and words below.
Input: Sentence: Game 6 was a blowout in which the Lakers defeated the Nuggets 119 - 92 to advance to its franchise's 30th NBA Finals appearance . Word1: NBA Finals. Word2: Lakers
Output:
Ground truth: winner (The type in *RE*_3)

Output of MoLE-CIE:
participant (The type in *RE*_3)

Table 16: Bad cases of Sequence 1.

ences. MoLE-CIE confuses their semantics. How to distinguish similar semantics (e.g., using contrastive learning methods) is still very challenging in our work.

In bad case 2, both "winner" and "participant" can describe the relationship between two entities, yet "winner" is more appropriate in the given context. MoLE-CIE fails to correctly infer the more precise relationship. In future work, we plan to explore ways to enhance this aspect.

I Analysis of MoE-PEFT Methods

Finetuning methods combining MoE and PEFT can also be applied to continual information extraction. For comparison, we select the most recent method, MoLA (Gao et al., 2025), to conduct a comparative experiment. The experimental results are presented in Table 17.

We want to highlight: (1) In continual information extraction, our performance is significantly better than that of MoLA. (2) Compared to other finetuning methods that combine MoE and PEFT, MoLE-CIE utilizes task experts to preserve knowledge from extraction tasks during continual learning, while also designing a gate reflection method

Sequence 1	ED_1	RE_1	NER_1	ED_2	RE_2	NER_2	ED_3	RE_3	NER_3	ED_4	RE_4	NER_4
MoLE-CIE	90.35	85.45	84.49	70.68	73.23	69.91	63.41	72.53	67.85	65.92	73.75	66.10
MoLA	88.84	77.78	80.17	69.35	69.29	62.71	55.11	61.09	60.34	57.49	65.84	58.08
Sequence 2	ED_1	ED_2	ED_3	ED_4	RE_1	RE_2	RE_3	RE_4	NER_1	NER_2	NER_3	NER_4
MoLE-CIE	90.35	59.77	51.06	50.40	62.72	63.28	65.22	68.72	70.18	68.91	67.44	67.29
MoLA	88.84	52.14	45.33	47.35	56.05	58.50	56.28	58.77	61.20	61.37	61.72	62.36

Table 17: Accuracy comparison between MoLE-CIE and MoLA on all current and previous splits.

based on knowledge distillation to address forgetting. Therefore, MoLE-CIE performs better in continual information extraction.

J Analysis of Additional NLP Tasks

To assess the generalizability of MoLE-CIE, we further evaluate its continual learning capabilities on a set of additional NLP tasks. We use a widely adopted continual learning benchmark, *SuperNI* (Wang et al., 2022a), as our experimental dataset. Following (Zhao et al., 2024), we construct a sequence of 15 splits by selecting three representative splits from each of the following categories: dialogue generation, information extraction, question answering, summarization, and sentiment analysis. For each split, we randomly sample 1,000 instances for training, and 100 instances each for validation and testing. Following (Zhao et al., 2024), we employ the same task sequences and evaluation metric, Rouge-L (Lin, 2004). The task sequences of SuperNI are listed in Table 18. The experimental results are presented in Table 19, which demonstrate the strong adaptability and scalability of MoLE-CIE across diverse scenarios.

Seq.	**Task sequences**
1	Task1572 → Task363 → Task1290 → Task181 → Task002 → Task1510 → Task639 → Task1729 → Task073 → Task1590 → Task748 → Task511 → Task591 → Task1687 → Task875
2	Task748 → Task073 → Task1590 → Task639 → Task1572 → Task1687 → Task591 → Task363 → Task1510→ Task1729 → Task181 → Task511 → Task002 → Task1290 → Task875

Table 18: Two different task sequences of SuperNI.

	Sequence 1	**Sequence 2**
SeqLoRA	21.32	19.61
Joint-training	60.76	60.10
L2P$^{\triangle}$	50.21	47.40
ProgPrompt$^{\triangle}$	48.36	45.51
NoRGa$^{\triangle}$	51.27	47.80
O-LoRA	50.00	48.28
SAPT	<u>52.22</u>	<u>51.63</u>
MoLE-CIE	**55.32**	**55.39**

Table 19: Rouge-L comparison between MoLE-CIE and competitors after training all splits on two sequences.