Don't Half-listen: Capturing Key-part Information in Continual Instruction Tuning

Yongquan He[2†], Wenyuan Zhang[4†], Xuancheng Huang[3†], Peng Zhang[1*], Lingxun Meng[2], Xiang Zhou[2], Ke Zeng[2], Xunliang Cai[2]

[1]School of Cyber Science and Engineering, Nanjing University of Science and Technology, China

[2]Meituan, China

[3]Zhipu.AI, China

[4]Institute of Information Engineering, Chinese Academy of Sciences, China

heyongquan{@meituan, 1997@gmail}.com

Abstract

Instruction tuning for large language models (LLMs) can drive them to produce results consistent with human goals in specific downstream tasks. However, the process of continual instruction tuning (CIT) for LLMs may bring about the catastrophic forgetting (CF) problem, where previously learned abilities are degraded. Recent methods try to alleviate the CF problem by modifying models or replaying data, which may only remember the surface-level pattern of instructions and get confused on held-out tasks. In this paper, we propose a novel continual instruction tuning method based on Key-part Information Gain (**KPIG**). Our method computes the information gain on masked parts to dynamically replay data and refine the training objective, which enables LLMs to capture task-aware information relevant to the correct response and alleviate overfitting to general descriptions in instructions. In addition, we propose two metrics, `P-score` and `V-score`, to measure the generalization and instruction-following abilities of LLMs. Experiments demonstrate our method achieves superior performance on both seen and held-out tasks.

1 Introduction

Large language models (LLMs) make remarkable breakthroughs in recent years (Zhao et al., 2023). LLMs such as PaLM (Chowdhery et al., 2023) and LLaMA (Touvron et al., 2023a) show powerful capabilities in multiple tasks such as information extraction, question answering, commonsense reasoning, and mathematical operations. One of the major issues is how to leverage the knowledge of LLMs pretrained with unsupervised or general objectives to produce results consistent with human intent during task-specific interactions (Zhang et al., 2023b). To endow LLMs with such "instruction-following" ability, instruction tuning is proposed as an effective technique that can bridge the gap between the generation process of LLMs and the objective of users (Ouyang et al., 2022; Zhang et al., 2023b,a).

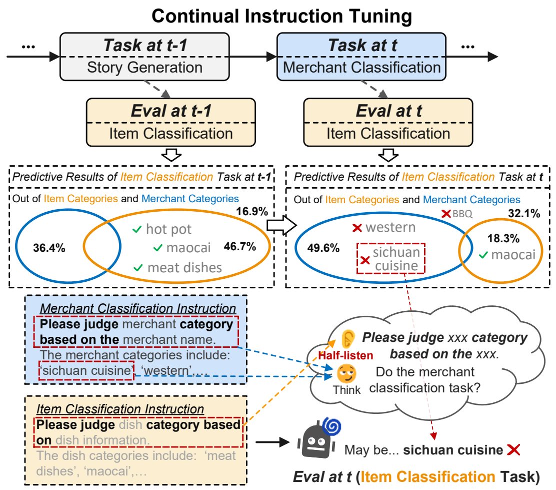

Figure 1: Task confusion on item classification (IC) after training merchant classification (MC). Note that IC is a held-out task for evaluation, and LLM at t generates more illegal categories defined in MC ($36.4\% \rightarrow 49.6\%$) as their instructions are similar.

Although tuning a pretrained LLM with instruction data before deployment gains wide application, it still faces challenges when dealing with incremental data and tasks (Zhang et al., 2023c). Continual learning (CL) (Biesialska et al., 2020) is introduced to avoid costly retraining on all collected instances (Biesialska et al., 2020), and continual instruction tuning (CIT) (Zhang et al., 2023c) is a sub-task of it about instruction data. However, catastrophic forgetting (CF) is still an unavoidable problem during CIT, which refers to the forgetting of previously learned tasks and the deterioration of original generalization ability (Zhao et al., 2022; Zeng et al., 2023b; Zhang et al., 2023c).

Recently, *replay*, *architecture*, and *regularization* are three main strategies to mitigate the CF

† Equal contribution.
* Corresponding authors: Peng Zhang (zhang_peng@njust.edu.cn).

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 23649–23668
July 27 - August 1, 2025

problem. *Replay* is the most prevalent strategy that leverages task-specific features to replay a small set of previous data (Yin et al., 2022; Mok et al., 2023) or generated pseudo samples (Zhao et al., 2022; Zeng et al., 2023b). *Architecture* obtains the target model by performing a model merging of other available LLMs (Xiao et al., 2023; Yu et al., 2023) or introducing task-specific components for newly emerging tasks (Madotto et al., 2021; Hu et al., 2022). Moreover, *regularization* is usually utilized as a penalty strategy to alleviate overfitting on seen tasks (Kirkpatrick et al., 2017).

Despite their impressive performance on seen tasks, these methods may only learn surface-level patterns (Zhang et al., 2023b) of instructions when applied to the CIT scenario. This observation is supported by prior research (Kung and Peng, 2023), which suggests that LLMs may generate unchanged responses on seen tasks and become confused on held-out tasks, even if we modify some components in original instructions. We also observe a similar phenomenon, as shown in Figure 1, compared to responses of the item classification task at $t-1$, the LLM generates more illegal categories that are not defined in the item classification instruction at t (after training on the merchant classification task). This half-listening phenomenon indicates that the overfitting to seen instructions is serious in CIT, potentially leading to confusion during inferring on held-out tasks. Therefore, we focus on a new challenge concerning the degradation of instruction-following and generalization abilities within the CIT framework, both of which are essential abilities of instruction-based LLMs.

In this paper, we propose a novel CIT paradigm based on key-part information gain (**KPIG**) to handle the above challenge. **Key parts** are consecutive spans in the instruction which provide task-aware guidance on the content, length, and format to generate desired responses. And we expect that LLMs can be sensitive to key parts for task-aware performance, which exhibits strong instruction-following and generalization abilities on various tasks. Firstly, we rewrite the instructions and corresponding key parts to diversify the combination of key parts and general descriptions. Then we selectively replay a small set of historical data whose information gain (IG) is the lowest. And IG is our proposed indicator used to measure the task-aware ability of LLMs, which is calculated by masking the key parts. Finally, we apply a Jensen–Shannon (Endres and Schindelin, 2003) divergence (JSD) on masked instructions, and IG is utilized as a dynamic temperature, to increase the IG margin relative to the surface-level patterns. Moreover, as instruction-following and generalization abilities are our concerns, we propose two novel evaluation metrics, `P-score` and `V-score`, instead of simply using Rouge-L (Lin, 2004) as previous methods (Mok et al., 2023; Zhang et al., 2023c). Experiments conducted on Super-NaturalInstructions (SupNatInst) (Wang et al., 2022) and our Chinese domain (Domain) datasets show superior performance on both seen and held-out tasks, and violations of instructions such as out-of-scope, wordy statements, and illegal formats are reduced.

Our contributions can be summarized as follows: 1) We propose a novel CIT paradigm by masking key parts to alleviate the half-listening problem of instructions. 2) We propose information gain as an indicator for measuring task-aware ability, which serves to dynamically replay data and refine the training objective. 3) We propose a novel evaluation metric `V-score` centered on instruction-following ability. 4) Compared to other CL baselines, our method achieves state-of-the-art performance on public and domain datasets.

2 Related Work

2.1 Instruction Tuning

LLMs show powerful emergent abilities in many downstream tasks (Chowdhery et al., 2023; Touvron et al., 2023b; Zhao et al., 2023). Since most LLMs are typically pretrained with the next word prediction error on large corpora, instruction tuning is proposed as an effective technique to further enhance the instruction-following ability of the generation process (Ouyang et al., 2022; Zhang et al., 2023b). And increasing the quantity, diversity, and creativity of instructions is empirically validated as an effective strategy to improve the instruction-following and generalization capabilities of LLMs (Zhang et al., 2023b; Xu et al., 2023; Zeng et al., 2023a). Collecting existing datasets and synthesizing data with LLMs are main strategies to obtain high-quality instruction data (Zhang et al., 2023b,a, 2025b). The former collects existing data and converts it into instruction-style datasets through templates or machine translation (BELLEGroup, 2023; Taori et al., 2023), while the latter like Evol-Instruct (Xu et al., 2023), instructWild (Ni et al., 2023) and Self-Instruct (Wang et al., 2023b) ask LLMs to rewrite seed instructions

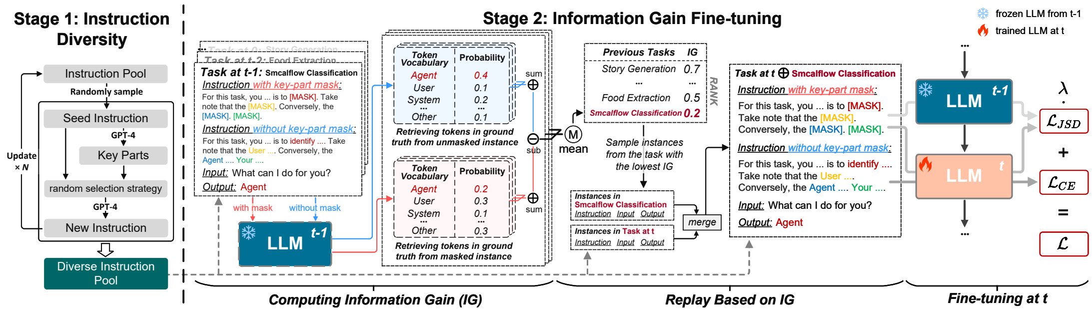

Figure 2: The continual instruction tuning framework of our KPIG. In the instruction diversity stage, we require GPT-4 to pay more attention to key parts during the rewriting process. In the information gain fine-tuning stage, we dynamically replay previous tasks with our learning objective based on IG to alleviate the half-listening problem.

based on specific strategies. In addition to increasing the diversity of task data, optimizing LLM with a comparison dataset collected by human feedback or LLMs also helps to generate desired responses (Ouyang et al., 2022; Zhang et al., 2023b). However, LLMs may only remember surface-level patterns of seen instructions, causing the output results not satisfy all constraints on held-out instructions (Zhang et al., 2023b; Kung and Peng, 2023). In this paper, we propose a key-part information mask mechanism to make LLMs focus more on tokens in instructions that are pertinent to the content, length, and format of the ground truths.

2.2 Continual Learning

Compared with multi-task learning, CL (Biesialska et al., 2020) refers to learning from sequential data across multiple time steps, which may lead to CF problem. Since CIT (Zhang et al., 2023c) is a sub-task of CL applied to instruction data, we do not discuss them separately. Recent methods mainly focus on tackling the forgetting of previously learned tasks, and CITB (Zhang et al., 2023c) categorizes them into three groups, *replay*, *architecture*, and *regularization*. *Replay*-based methods replay experience with historical data (Yin et al., 2022; Scialom et al., 2022; Mok et al., 2023) or generated pseudo samples (Zhao et al., 2022; Zeng et al., 2023b), while *architecture*-based methods introduce task-specific parameters (Madotto et al., 2021; Hu et al., 2022) or gradually merging models trained on different tasks (Xiao et al., 2023; Yu et al., 2023). Moreover, *regularization*-based methods are strategies for objective optimization and overfitting penalty (Hinton et al., 2015; Kirkpatrick et al., 2017), which are used alone or in combination with other methods (Mok et al., 2023; Zhao et al., 2022; Zeng et al., 2023b). Despite effectively alleviating the forgetting of previously learned tasks, they lack attention to instructions and may half-listen to surface-level descriptions in held-out instructions when applied to CIT. In this paper, we selectively replay a few historical data and employ temperature based on the IG of masked key parts, which encourages LLMs to be more sensitive to task-aware information in instructions.

3 Methodology

This section introduces our proposed method, named Key-part Information Gain (**KPIG**), for continual instruction tuning on LLMs. We first define the task and notations in §3.1. Then we detail our instruction diversity module (§3.2) and information gain fine-tuning (§3.3) module in Figure 2. Moreover, considering the specificity of sequential training in CIT, we introduce how to reconstruct datasets and evaluate performance (§3.4).

3.1 Task Definition and Notations

We finetune a LLM with a stream of task sets $\mathcal{T}^T = \{\tau_1, \tau_2, \ldots, \tau_n\}_{t=1}^T$ sequentially, where T is the number of time steps and n is the number of tasks at corresponding time t. Each instance d_τ in the task τ can be formed as a triple (i, c, y): instruction i, which is a natural language text to demonstrate the definition of current task in human style; an optional context c which provides supplementary information for context; an expected output y corresponding to the instruction and the context. And each task τ can be split into τ_{train} and τ_{test}. At each time step t, we finetune the LLM on a mixture of τ_{train}, where $\tau \in \mathcal{T}^t$. After completing the T-step training, we evaluate its performance on the τ_{test} of seen tasks $\mathcal{T}_{seen}$ and held-out tasks $\mathcal{T}_{unseen}$, where $\mathcal{T}_{seen} \cap \mathcal{T}_{unseen} = \varnothing$.

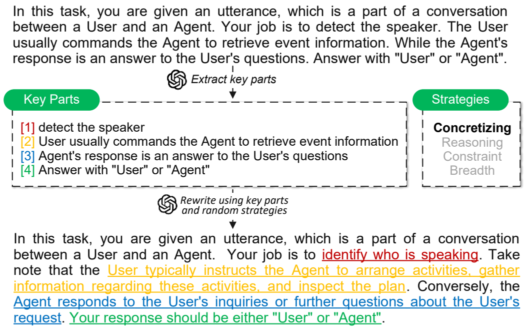

Figure 3: An example of the instruction diversity.

3.2 Instruction Diversity

Due to the inefficiency of manual annotation, the number of instructions for a task may be scarce (Zhang et al., 2023b, 2025c). Taking the SupNatInst as an example, each task has hundreds to thousands of instances but only one human-handwritten instruction. Moreover, humans may struggle to produce different instructions with the same meaning (Xu et al., 2023; Zhang et al., 2025a). Motivated by WizardLM (Xu et al., 2023), we diversify the combination of key parts and other general descriptions in instructions via GPT-4 (OpenAI, 2023) and different templates, which aims to prompt the LLM to identify task-aware information in instructions with varying levels of complexity.

As shown in Figure 3, we first ask GPT-4 to generate key parts of the seed instruction and return them as a list. Then we input seed instructions and their corresponding key parts to gain new instructions and key parts recursively. As we expect that key parts can play an important role in controlling text generation, we encourage GPT-4 to focus on key parts when evolving with the following strategies: 1) Concretizing, which replaces general concepts in key parts with more specific concepts. 2) Reasoning, which explicitly requests multiple-step reasoning if key parts can be organized into a few simple thinking processes. 3) Constraint, which adds one more constraint on seed instructions. 4) Breadth, which rewrites the seed instruction while keeping length close and key parts unchanged. It should be noted that we also need to give GPT-4 some demonstrations when rewriting the instructions and obtaining the key parts. More details about templates can be found in Appendix A.

3.3 Information Gain Fine-tuning

Although diversified instructions are proven helpful to train instruction-following LLMs (Xu et al., 2023), we argue that LLMs sometimes may only half-listen to surface-level patterns (Zhang et al., 2023b; Kung and Peng, 2023), which refers to overfitting on seen instructions and confusion with held-out tasks. To measure the task-aware ability of current LLM for specific task τ, we propose the concept of information gain (IG) by masking key parts in each instance $d_\tau = (i, c, y)$:

$$\mathcal{G}(d_\tau, d_\tau^m) = \mathrm{Info}(y|x) - \mathrm{Info}\big(y|\mathrm{mask}(x)\big), \quad (1)$$

where $d_\tau \in \tau$ and $\mathrm{Info}(\cdot)$ is a weighted sum of the generation probabilities of sequence y. Meanwhile, $\mathrm{mask}(\cdot)$ denotes the operation of applying a key-part mask to the instruction i, which replaces the key parts in instruction i with `[MASK]` symbol to obtain mask instruction i^m and surface-level instance $d_\tau^m = (i^m, c, y)$. Giving an input $x = (i, c)$ and an expected output $y = \{t_1, t_2, \dots, t_K\}$, where K is the length of y after the tokenization process. The $\mathrm{Info}(y|x)$ can be calculated as follows:

$$\mathrm{Info}(y|x) = \mathrm{Info}\big(y|(i, c)\big) = \sum_{k=1}^{K} \alpha^k p(t_k), \quad (2)$$

where $p(t_k)$ is the probability of expected token t_k in y given the concatenation of input x and the previously generated output tokens $t_{history}$. α is an exponential decay hyperparameter because the probability of subsequent tokens always becomes greater as the inference process progresses. As for the $p(t_k)$, we gain it via retrieving from the softmax function results on the head logits according to its index in the vocabulary:

$$p(t_k) = \mathrm{softmax}\big(\mathrm{LLM}(x; t_{history})\big)[t_k]. \quad (3)$$

The information gain defined above represents the uncertainty reduction of the masked part to the expected output. For example, if the information gain is similar between complete and masked instructions, it indicates that the LLM may half-listen to the surface-pattern of the input instruction.

At each time step t, we randomly sample N instances and compute their IG for each seen task. Then we select M seen tasks with the lowest mean IG as replay tasks, and merge the $|MN|$ instances into current training tasks $\mathcal{T}^t$. Since our goal is to widen the gap between the complete (without mask) instance d_τ and the surface-level (with mask) instance d_τ^m, the loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{CE}(d_\tau) + \lambda \mathcal{L}_{JSD}\Big(\frac{p_t(d_\tau^m)}{\beta} \Big\| \frac{p_{t-1}(d_\tau^m)}{\beta}\Big), \quad (4)$$

where $\mathcal{L}_{CE}$ is cross entropy loss to maximize the ground truths. $\mathcal{L}_{JSD}$ is Jensen–Shannon (Endres and Schindelin, 2003) divergence of two distributions output by current LLM and frozen LLM from $t-1$, which is usually utilized as penalty in CL methods (Zhao et al., 2022; Zeng et al., 2023b; Mok et al., 2023) to preserve original abilities. However, we only apply JSD on masked instance d_τ^m rather than complete instance d_τ. And JSD value is symmetric and in $[0, 1]$ to easily balance $\mathcal{L}_{JSD}$ and $\mathcal{L}_{CE}$. λ is a hyperparameter that controls the weight of $\mathcal{L}_{JSD}$. Moreover, β is the dynamic temperature to soften probability distribution and is calculated as follows:

$$\beta = 2 - \min\big(\mathcal{G}(d_\tau, d_\tau^m), 1\big), \tag{5}$$

where $\min$ represents the scaling of $\mathcal{G}(d_\tau, d_\tau^m)$ into the range $(-\infty, 1]$. The lower the information gain, the greater the opportunity we give other tokens to improve the generalization ability of the LLM. By doing so, $\mathcal{L}_{CE}$ maximizes likelihood for complete instances, and $\mathcal{L}_{JSD}$ dynamically adjusts the degree of conservatism when instructions are masked, enabling the LLM to be sensitive to key parts and alleviate the half-listening problem. The detailed implementation is shown in Appendix C.

3.4 Evaluation Protocol

Since CIT trains tasks sequentially, we first introduce our construction method of multi-step datasets in this section. Furthermore, different from using ROUGE-L (Lin, 2004) as metric in previous methods (Mok et al., 2023; Zhang et al., 2023c), we propose a multi-dimensional evaluation method that pays more attention to the instruction-following ability of LLMs.

Data restructuring. We evaluate our method on SupNatInst and Domain datasets, where each task contains a task definition, a few demonstrations, and several instances. SupNatInst consists of over 1000 NLP tasks and 76 categories (e.g., text classification, information extraction and etc.) (Wang et al., 2022). We select 128 tasks in 40 categories from SupNatInst, 88 tasks are used for training $\mathcal{T}_{seen}$ and 40 as held-out tasks $\mathcal{T}_{unseen}$. And our Chinese domain dataset has 20 tasks and 12 categories, where 13 tasks are used for training $\mathcal{T}_{seen}$ and 7 as held-out tasks $\mathcal{T}_{unseen}$. We use two strategies, single-task (ST) and single-category (SC), to build multi-step training datasets. For the ST setting, we fix n equal to 1, where only 1 task in $\mathcal{T}_{seen}^t$ at time step t. For the SC setting, we divide seen tasks into multiple groups according to their categories, and train different categories at each time step, because real training scenarios may gradually enhance model abilities of specific categories when training tasks are not available synchronously. Furthermore, to enhance the balance and diversity of each test dataset while accelerating the evaluation process, we sample a few instances for each τ_{test} based on Self-BLEU (Zhu et al., 2018) score and label distribution. More details about datasets can be found in Appendix D.

Evaluation metrics. Previous methods use the ROUGE-L score to measure model performance (Mok et al., 2023; Zhang et al., 2023c), which may not comprehensively evaluate the instruction-following ability. For example, `{[1, 2, 3]}` and `[1, 2, 3]` have same Rouge-1 scores with the ground truth `[1, 2]`, but the instruction explicitly requires generating a one-dimensional list format. We evaluate model on τ_{test} of trained (seen) task set $\mathcal{T}_{seen}$ and held-out (unseen) task set $\mathcal{T}_{unseen}$ with the following metrics:

- **WFR** measures the wrong-format rate of tasks that instructions in them explicitly constrain delimiters, sequence, formats, or length limits.
- **OOS** measures the out-of-scope rate of classification or extraction tasks whose instructions constrain output choices.
- **WR** measures wordy rate when the length of responses are greater than the threshold.
- **F1** measures the performance for sequence labeling tasks.
- **ACC** measures the precision for classification tasks or execution accuracy for code tasks.
- **ROUGE** and **BLEU** measures the similarity for tasks such as summarization.
- **Match** measures the match rate for tasks that the ground truths are unordered sets.
- **GPT** leverages GPT-4 to measure whether tasks of generating open-ended short texts are reasonable, which require commonsense or reasoning skills to verify.

Then we use `P-score` and `V-score` to measure the performance and instruction-following ability. `P-score` is the average of F1, ACC, ROUGE, BLEU, Match, and GPT, which can measure generalization ability on held-out tasks. `V-score` is the average of WFR, OOS, and WR, acting as an indicator of instruction-violation degree.

Model	Sup-NatInst-ST				Sup-NatInst-SC				Domain-ST				Domain-SC			
	Seen Tasks		Held-out Tasks		Seen Tasks		Held-out Tasks		Seen Tasks		Held-out Tasks		Seen Tasks		Held-out Tasks	
	P-score	V-score	P-score	V-score	P-score	V-score	P-score	V-score	P-score	V-score	P-score	V-score	P-score	V-score	P-score	V-score
SFT	35.1	12.0	25.9	24.1	51.1	4.5	34.2	6.7	43.5	12.0	37.0	16.3	52.2	8.3	43.1	10.5
LoRA	33.7	12.4	26.7	23.0	48.7	4.7	36.1	5.3	41.8	12.8	38.2	15.9	49.5	8.9	44.6	10.0
L2	34.7	12.3	26.5	23.2	50.4	4.8	35.4	5.6	42.9	12.6	37.7	16.7	50.2	8.6	42.9	10.4
EWC	30.2	13.5	25.1	24.6	47.9	5.9	33.6	7.4	41.4	13.2	35.8	17.9	48.7	10.4	41.5	11.8
DARE	-	-	-	-	54.4	3.9	39.8	4.4	-	-	-	-	56.6	5.7	45.9	10.1
LM-Cocktail	-	-	-	-	55.0	3.7	40.0	4.1	-	-	-	-	56.9	6.3	46.4	10.5
PCLL	50.5	5.4	38.2	5.6	-	-	-	-	52.4	10.8	43.7	14.6	-	-	-	-
DCL	50.2	4.9	38.8	5.2	-	-	-	-	52.5	10.3	44.1	12.2	-	-	-	-
DYNAINST	50.9	4.6	38.7	4.4	54.2	4.2	40.7	3.3	53.2	9.1	44.6	10.9	56.3	8.3	47.2	9.6
InsCL	52.5	4.0	38.4	5.5	57.1	2.8	40.2	4.9	-	-	-	-	-	-	-	-
KPIG	52.2	3.5	42.5	1.7	56.5	2.4	43.6	$*$1.2	54.1	$*$4.8	47.8	$*$3.3	57.5	$*$4.0	49.7	$*$2.7
INIT	43.2	5.3	$*$43.8	$*$1.5	43.2	5.3	$*$43.8	1.5	28.9	10.8	39.1	13.5	28.9	10.8	39.1	13.5
MULTI	$*$59.8	$*$2.2	41.4	4.2	$*$59.8	$*$2.2	41.4	4.2	$*$60.0	9.7	$*$49.9	10.8	$*$60.0	9.7	$*$49.9	10.8

Table 1: Performance of different methods on Sup-NatInst and Domain datasets. $*$ indicates the best, and _ indicates the second best. The higher the `P-score`, the better the model performance. The lower the `V-score`, the stronger the instruction-following ability. Since **INIT** *sometimes* serves as an upper bound for held-out tasks, and **MULTI** is *sometimes* the upper bound for seen tasks, we report their results.

4 Experiments

4.1 Experimental Setup

Baselines. We compare our method in the CIT setting with the following baselines. **INIT** is the foundation LLM without training. **MULTI** shuffles instances in all training tasks and trains them together. **SFT** (Ouyang et al., 2022) directly fine tunes the LLM on seen tasks sequentially. **LoRA** (Hu et al., 2022) updates the low-rank matrices while the LLM backbone is fixed. **L2** and **EWC** (Kirkpatrick et al., 2017) mitigate forgetting by regularizing the loss to penalize the changes of important parameters. **DARE** (Yu et al., 2023) and **LM-Cocktail** (Xiao et al., 2023) obtain the target LLM by model merging, which train multiple models on different tasks and merge them into a single model through weighted average. **DYNAINST** (Mok et al., 2023) and **InsCL** (Wang et al., 2024) dynamically determines which instances are stored and replayed based on their proposed metrics. **PCLL** (Zhao et al., 2022) and **DCL** (Zeng et al., 2023b) generate pseudo samples for history tasks and utilize knowledge distillation strategy to mitigate catastrophic forgetting.

Hyperparameters. We choose LLaMA-2-7B-Chat (Touvron et al., 2023b) and baichuan-vicuna-chinese-7b [1] as foundation models for experiments on SupNatInst and Domain datasets respectively. Our experiments are implemented based on DeepSpeed (Rasley et al., 2020) and FastChat (Zheng et al., 2023). And 8 NVIDIA A100 GPUs are used. We optimize the model parameters by using AdamW optimizer (Loshchilov and Hutter, 2018) with the learning rate of $2e-5$. The batch size is 384 with 16 gradient accumulation steps and 3 sentences per GPU. We conduct a grid search to find other hyperparameters that maximize the average `P-score` on seen and held-out tasks. The optimal settings are: $\{\alpha = 0.3, \lambda = 0.02, M = 10, N = 10, epoch = 1\}$ on Sup-NatInst and $\{\alpha = 0.6, \lambda = 0.01, M = 3, N = 100, epoch = 1\}$ on Domain. Additionally, we iteratively perform 30 evolutions for each task in the instruction-diversity stage. And when evaluating held-out tasks, we add 2 additional pre-written demonstrations to the input context. More detailed information about the implementation can be found in Appendix C.

4.2 Main Results

Table 1 summarizes the performances of different methods. It should be noted that **INIT** is a pretrained LM, and **MULTI** trains the LLM with all seen tasks together, so they have the same results on ST and SC. We train **PCLL** and **DCL** only on the datasets constructed by the ST strategy which are designed to learn single-task parameters sequentially. Moreover, we only conduct SC experiments on **DARE** and **LM-Cocktail** which merge peer models on each category, because training a sub-LLM for each task requires a much larger resource than training a sub-LLM for each category. Our observations are summarized as follows.

Firstly, foundation LLMs require more training on domain-specific datasets to achieve perfor-

[1] `https://huggingface.co/fireballoon/baichuan-vicuna-chinese-7b`

Model	Seen Tasks P-score	Seen Tasks V-score	Held-out Tasks P-score	Held-out Tasks V-score
w/o div	52.5	4.1	41.6	3.2
w/o mask	48.3	4.9	41.2	3.6
w/o jsd	*52.9	4.7	39.4	5.9
w/o temp	51.7	4.4	40.9	4.0
KPIG	52.2	*3.5	*42.5	*1.7

Table 2: Ablation studies on Sup-NatInst-ST.

mance improvements. In the benchmark of CIT (Zhang et al., 2023c), multi-task learning (**MULTI**) is served as an upper bound on seen tasks while **INIT** is the upper bound for held-out tasks. The difference is that **MULTI** achieves the best P-score on held-out tasks of Domain. This may be because our domain-specific dataset is highly specialized, which leaves the foundational model (**INIT**) lacking in pertinent knowledge without training. In addition, compared with the held-out results of **INIT** on Sup-NatInst, most methods show performance degradation of P-score and V-score, which may indicate forgetting ability in the foundation LLM.

Secondly, the catastrophic forgetting problem of the single-task setting is more severe than the single-category setting. The performance of **SFT**, **LoRA**, **L2**, and **EWC** on seen tasks and held-out tasks under the ST setting is significantly worse than the SC setting, while the performance gap between ST and SC on **DYNAINST** and **KPIG** is relatively small. Furthermore, **MULTI** stands out with the highest P-score on all seen tasks. The above phenomenons indicate that the training difficulty and overfitting become more pronounced when training on a single task sequentially, and mixing data from different tasks and replaying data can help mitigate performance degradation.

Thirdly, model-merge methods perform better on seen tasks than on held-out tasks. This may be because they selectively inherit abilities of different task categories from multiple models, but abilities are limited when faced with held-out tasks.

Finally, our proposed **KPIG** achieves the best performance, especially on held-out tasks and the instruction-following ability (V-score). The P-score of **KPIG** on seen tasks is slightly lower than the **InsCL**, which may be related to the replayed number. On the held-out tasks, our method performs significantly better than other CL baselines in both P-score and V-score, which shows stronger generalization ability and instruction-following ability. Moreover, the V-score of other baselines on the Domain dataset is much larger than the Sup-NatInst dataset, while our method maintains lower V-score on both Sup-NatInst and domain-specific datasets. This indicates half-listening and instruction violations may be more likely to occur on a specific domain, and our method can better capture the task-aware information and improve the instruction-following ability.

4.3 Ablation Study

To evaluate the effectiveness of each component in **KPIG**, we conduct ablation studies on the Sup-NatInst-ST dataset. Firstly, we remove the instruction diversity module (**w/o div**) and only extract key parts for the initial instruction of each task. Then, to investigate the significance of our key-part mask mechanism, we remove the mask step (**w/o mask**). The **w/o mask** setting replays data based on predictive entropy like **DYNAINST** and performs JSD on the predictive distribution of the complete instruction of current LLM and frozen LLM. Finally, we investigate the effects of removing $\mathcal{L}_{JSD}$ (**w/o jsd**) and dynamic temperature (**w/o temp**).

The results are shown in Table 2. When only initial instructions for each task are used without diversification (**w/o div**), the P-score of seen tasks is slightly higher than **KPIG**, but the performance of held-out tasks become worse, indicating that increasing data diversity helps alleviate overfitting and preserve generalization. In the **w/o mask** setting, the V-score drops significantly and the P-score of seen tasks is much lower than **KPIG**. It proves the effectiveness of measuring and learning task-aware information by masking key-part in instructions, which assists LLMs in comprehending the tasks to be executed rather than simply maintaining the original ability. The results of **w/o jsd** and **w/o temp** on held-out tasks suggest that they are helpful in maintaining instruction-following and generalization abilities. Moreover, the decline in **w/o mask** results for seen tasks and in **w/o jsd** for held-out tasks suggests an interdependence between key-part mask and $\mathcal{L}_{JSD}$. Without $\mathcal{L}_{JSD}$ and key-part mask, LLMs may struggle to widen the gap between task-aware constraints in key parts and some general descriptions in instructions, which is crucial for balancing learning new information with maintaining original judgments.

4.4 Investigations on Information Gain

Herein, we investigate the correlation between our information gain and the instruction-following abil-

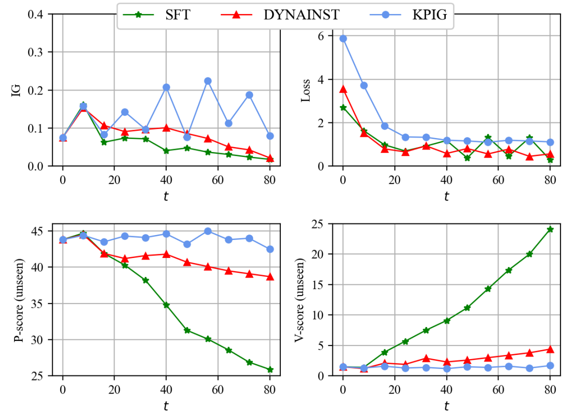

Figure 4: The changing trends of information gain, loss, P-score, and V-score on Sup-NatInst-ST over steps.

Metric	Ins_1	Ins_2	Ins_3	Ins_4	Ins_5	Ins_6
P-score	68.0	67.0	68.0	70.0	72.0	66.0
V-score	4.0	14.0	8.0	3.0	3.0	15.0
IG	0.28	0.22	0.26	0.29	0.30	0.15

Table 3: Results of the smcalflow classification task (one of seen tasks) on 6 held-out instructions.

ity on held-out tasks and held-out instructions.

Overfitting. As shown in Figure 4, the IG of our method oscillates above the initial value, while other methods begin to decline at approximately $t = 20$. This interesting oscillation may be related to our replay mechanism based on IG, which chooses tasks with the lowest IG. The curve gradually rises when IG is low, and then falls back to the level of the foundation model when it reaches the upper bound. Meanwhile, the changing trends of `P-score` and `V-score` of held-out tasks are in alignment with information gain, indicating the validity of employing IG as a metric for measuring task-aware ability. In addition, compared with other methods, our loss progression maintains a more stable and smooth decline. This may be because our method can effectively alleviate overfitting on individual tasks and does not require more recalibrations after training previous tasks.

Held-out instructions. To further explore the instruction-following and generalization abilities, we modify the instruction of smcalflow classification task after training. We collect 6 held-out instructions which are not seen during training. As Table 3 shows, the `P-score` and `V-score` on held-out instructions are significantly correlated with information gain, indicating that information gain

Response	INIT	MULTI	SFT	LM-Cocktail	DYNAINST	KPIG
User	0.0	49.0	0.0	4.0	85.0	14.0
Agent	0.0	50.0	0.0	58.0	15.0	8.0
user (required)	1.0	0.0	0.0	15.0	0.0	45.0
agent (required)	69.0	0.0	0.0	9.0	0.0	26.0
IG	0.13	0.07	0.00	0.11	0.04	0.19

Table 4: Statistics of responses after modifying constraints in the smcalflow classification instruction. It should be noted that this task requires the first letter of `User` and `Agent` to be capitalized during training, and we require `user` and `agent` during testing.

can be used to measure the generalization ability and instruction-following ability.

In addition, as shown in Table 4, we modify the constraints of capital letters (`Answer with User or Agent`) to obtain the misleading constraints (`Answer with user or agent`). Most of the responses of **INIT** are legal, indicating that the initial foundation LLM has strong instruction-following ability. **LM-Cocktail** gives a small proportion of legal responses because model-merging methods can inherit abilities of other LLMs. All responses of **MULTI** and **DYNAINST** are illegal, which means they are overfitting to training instructions and half-listen to the misleading instruction during testing. The responses given by **SFT** are all irrelevant due to catastrophic forgetting in CL, which forgets not only historical tasks but also the ability of the foundation LLM. Moreover, 71% responses of our method are `user` and `agent`, and our average information gain on the misleading instruction is the highest, which shows that **KPIG** has a stronger ability to alleviate the half-listening problem even if similar instructions are seen during training.

5 Conclusion

In this paper, we propose a novel CIT method to alleviate catastrophic forgetting and half-listening problems, which enables LLMs to be sensitive to task-specific constraints of both seen and held-out tasks. Our method calculates the information gain of masked key parts, to selectively replay historical data and dynamically adjust the temperature. Experimental results show strong instruction-following and generalization abilities in comparison to other continual learning methods. Furthermore, our investigation into the proposed `P-score`, `V-score`, and IG not only confirms their relevance in model performance and instruction adherence, but also demonstrates that our method effectively alleviates overfitting to seen-task instruction and maintains the generalization ability.

Acknowledgments

This work is supported by the fund of Laboratory for Advanced Computing and Intelligence Engineering. We also appreciate the assistance from Hongbin Zhang, Xinyu Wang, and Chen Shi of Nanjing University of Science and Technology. We thank all anonymous reviewers for their valuable comments and suggestions on this work.

Limitations

In this paper, we use GPT-4 to extract key parts of instructions and diversify instructions, but the gap between this method and manual writing in controllability and accuracy is not fully evaluated. We propose WFR, OOS, and WR as evaluation dimensions of the instruction-following ability based on manually annotating explicit constraints in instructions. However, there may be other constraints or ways for evaluating the instruction-following ability that exist and deserve to be considered. Moreover, we dynamically replay instances and adjust the training objective by calculating information gain of key parts, making the LLM more sensitive to task-specific constraints in instructions and thereby alleviating the half-listening problem. Our experiments (Table 4) also find that such half-listening problem also occurs in multi-task learning, so the implications of our mask information gain on other natural language processing tasks involving LLMs and the effects of masking other parts (e.g., context, demonstrations) within instances can be explored in the future.

Ethics Statement

In this paper, we propose a novel CIT paradigm to alleviate the half-listening problem, which aims to improve instruction-following ability and generalization ability of LLMs. Our experiments are conducted with the publicly available Super-NaturalInstructions dataset, our in-house dataset, and LLMs from open sources, one of whose initial intentions is to promote the development of instruction-based LLMs. Since LLMs trained with web data may produce toxic content, we must state that the texts generated by our method do not represent our opinions. To alleviate such potential negative impacts, we can adopt appropriate detoxification strategies and principle constraints, and we encourage future work to explore these issues.

References

BELLEGroup. 2023. Belle: Be everyone's large language model engine. https://github.com/LianjiaTech/BELLE.

Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6523–6541. International Committee on Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Dominik Maria Endres and Johannes E Schindelin. 2003. A new metric for probability distributions. *IEEE Transactions on Information theory*, 49(7):1858–1860.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Po-Nien Kung and Nanyun Peng. 2023. Do models really learn to follow instructions? an empirical study of instruction tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1317–1328. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul A. Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. 2021. Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7452–7467. Association for Computational Linguistics.

Jisoo Mok, Jaeyoung Do, Sungjin Lee, Tara Taghavi, Seunghak Yu, and Sungroh Yoon. 2023. Large-scale lifelong learning of in-context instructions and how to tackle it. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 12573–12589. Association for Computational Linguistics.

Jinjie Ni, Fuzhao Xue, Yuntian Deng, Jason Phang, Kabir Jain, Mahir Hitesh Shah, Zangwei Zheng, and Yang You. 2023. Instruction in the wild: A user-based instruction dataset. `https://github.com/XueFuzhao/InstructionWild`.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 6107–6122. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9840–9855. Association for Computational Linguistics.

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 *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 663–677. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Xingrun Xing. 2023. Lm-cocktail: Resilient tuning of language models via model merging. *CoRR*, abs/2311.13534.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin

Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *CoRR*, abs/2304.12244.

Wenpeng Yin, Jia Li, and Caiming Xiong. 2022. Contintin: Continual learning from task instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3062–3072. Association for Computational Linguistics.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2023. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *CoRR*, abs/2311.03099.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023a. Agenttuning: Enabling generalized agent abilities for llms. *CoRR*, abs/2310.12823.

Min Zeng, Wei Xue, Qifeng Liu, and Yike Guo. 2023b. Continual learning with dirichlet generative-based rehearsal. *CoRR*, abs/2309.06917.

Ge Zhang, Yemin Shi, Ruibo Liu, Ruibin Yuan, Yizhi Li, Siwei Dong, Yu Shu, Zhaoqun Li, Zekun Wang, Chenghua Lin, Wenhao Huang, and Jie Fu. 2023a. Chinese open instruction generalist: A preliminary release. *CoRR*, abs/2304.07987.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2023b. Instruction tuning for large language models: A survey. *CoRR*, abs/2308.10792.

Wenyuan Zhang, Tianyun Liu, Mengxiao Song, Xiaodong Li, and Tingwen Liu. 2025a. SOTOPIA-Ω: Dynamic strategy injection learning and social instruction following evaluation for social agents.

Wenyuan Zhang, Shuaiyi Nie, Jiawei Sheng, Zefeng Zhang, Xinghua Zhang, Yongquan He, and Tingwen Liu. 2025b. Revealing and mitigating the challenge of detecting character knowledge errors in llm role-playing. *arXiv preprint arXiv:2409.11726*.

Wenyuan Zhang, Shuaiyi Nie, Xinghua Zhang, Zefeng Zhang, and Tingwen Liu. 2025c. S1-Bench: A simple benchmark for evaluating system 1 thinking capability of large reasoning models. *arXiv preprint arXiv:2504.10368*.

Zihan Zhang, Meng Fang, Ling Chen, and Mohammad-Reza Namazi-Rad. 2023c. CITB: A benchmark for continual instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9443–9455. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Yingxiu Zhao, Yinhe Zheng, Zhiliang Tian, Chang Gao, Jian Sun, and Nevin L. Zhang. 2022. Prompt conditioned VAE: enhancing generative replay for lifelong learning in task-oriented dialogue. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11153–11169. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100. ACM.

A Templates

Table 5 shows our English templates for key-part extraction and instruction diversity. We have four evolving strategies. The strategies of concretizing, reasoning, and constraint make instructions more detailed, complex, and longer. The breadth strategy rewrites the general description within the instruction while keeping the key parts and length of the instruction nearly unchanged.

We obtain more combinations of key parts and instructions for each task with the following instruction-diversity process. Initially, each task has an instruction pool, which contains a manually written instruction related to the task definition. For each task, we first randomly select an instruction as the seed instruction from the instruction pool. We use the OpenAI-API[2] (gpt-4-0613[3]) and key-part extraction template to extract key parts for the seed instruction. Then we randomly apply one strategy from the four evolving templates on the seed instruction to obtain the evolution instruction. Finally, we extract the key parts of the evolution instruction and add them to the instruction pool. We iteratively repeat such process until the size of the instruction pool reaches 31.

B Hyperparameter Sensitivity

In our experiments, we finetune the exponential decay α in $\{0.05, 0.1, 0.3, 0.6, 1.0\}$, the weight λ in $\{0.001, 0.01, 0.02, 0.03, 0.05, 0.1\}$, the learning rate in $\{5e-6,\ 1e-5,\ 2e-5,\ 3e-5,\ 5e-5\}$ according to the average `P-score` on seen and held-out tasks. In addition, since the number of replay instances M and N are key hyperparameters that affect model performance and runtime, we conduct further experiments on it. As shown in Figure 5, we report the influence of N, and find that increasing N improves the `P-score` of seen tasks, but may have a negative effect on the performance of held-out tasks and instruction-following abilities. Therefore, we choose a trade-off reports based on the average `P-score` on seen and held-out tasks by fixing M and then find N.

As shown in Table 11, we add more experiments on hyperparameter and conduct further analysis. When fixing $\alpha = 0.3$, increasing λ from 0.001 to 0.1 shows a clear trade-off. While the P-score of seen tasks decreases from 54.7 to 47.6, the P-score of held-out tasks increases from 37.1 to 44.0, indicating stronger regularization benefits for generalization. Moreover, with fixed $\lambda = 0.02$, increasing α from 0.3 to 1.0 causes the P-score of both seen and held-out tasks to decline, suggesting excessive values impair overall performance. This may be because as the generation progresses, the posterior probability of the subsequent tokens increases, requiring a smaller α penalty. Therefore, to achieve a relative optimal balance, we choose $\lambda = 0.02$ and $\alpha = 0.3$ based on the average P-score on seen and held-out tasks. However, we must admit that our exploration of λ and α has limitations, because these two indicators may be related to the alignment tax or other issues of the CL process of the LLM. Dynamically adjusting these two hyperparams according to our IG may be more rigorous.

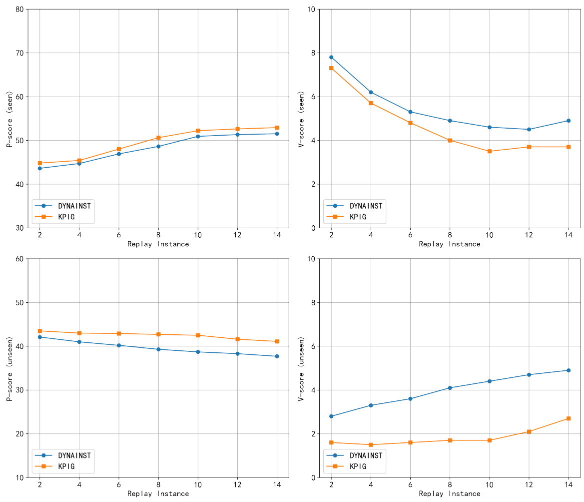

Figure 5: The impact of N on model performance.

C Implementation

Our detailed algorithm implementation is shown in Algorithm 1. In practice, sampling new instruction for current seed instruction from the instruction-diversity cached pool (line 13) can serve as a pre-processing step. And the IG results and outputs of $\mathcal{M}_{t-1}$ calculated during the replay stage (line 3-8) can be reused during the fine-tuning stage (line 12-19). Therefore, in real fine-tuning, our method only has $\mathcal{M}_t$ in the graphics memory.

For the experimental settings of other baselines, we give priority to the hyperparameters reported in their paper. In particular, to be fair and reproduce the better performance of baselines, M and N of **DYNAINST** adopt the same settings as **KPIG**, and

[2]https://api.openai.com/v1/chat/completions

[3]https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo

Key-part extraction

```
What is the #key part# in the #instruction#?
The #key part# refers to the consecutive span in the #instruction# that has guiding significance
for the format, length, content, and rationality of the ground truth when bridging #input# to
#output#.
Please return key parts as a list.
#instruction#:
{...}
#input#:
{...}
#output#:
{...}
```

□Concretizing ☑Reasoning □Constraint

```
I want you act as an Instruction Creator.
Your goal is to draw inspiration from the #Given Instruction# and #Key Part# to create a brand
new instruction #Created Instruction#.
The #Created Instruction# must be reasonable and must be understood and responded by humans.
And this #Created Instruction# can guide the #Input# to give the #Output#.
Your #Created Instruction# cannot omit the non-text parts such as the table and code in the #Given
Instruction#.
You should complicate the #Given Instruction# using the following method:
□Please replaces general concepts in #Key Part# with more specific concepts.
☑If #Key Part# can be organized into a few simple thinking processes, you can rewrite it to
explicitly request multiple-step reasoning.
□Please add one more constraints/requirements into #Given Instruction#.
You should try your best not to make the #Created Instruction# become verbose, #Created Instruction#
can only add 10 to 20 words into the #Given Instruction#.
’#Given Instruction#’, ’#Created Instruction#’, ’given instruction’ and ’created instruction’ are
not allowed to appear in #Created Instruction#.
#Given Instruction#:
{...}
#Key Part#:
{...}
#Input#:
{...}
#Output#:
{...}
#Created Instruction#:
```

Breadth

```
I want you act as a Instruction Rewriter.
Your goal is to draw inspiration from the #Given Instruction# and #Key Part# to rewrite a brand
new instruction #Rewritten Instruction#.
This #Rewritten Instruction# should belong to the same domain as the #Given Instruction# but be
even more rare.
And this #Rewritten Instruction# can guide the #Input# to give the #Output#.
#Key Part# in the #Given Instruction# should be unchanged.
The LENGTH and complexity of the #Rewritten Instruction# should be similar to that of the #Given
Instruction#.
The #Rewritten Instruction# must be reasonable and must be understood and responded by humans.
’#Given Instruction#’, ’#Rewritten Instruction#’, ’given instruction’ and ’rewritten instruction’
are not allowed to appear in #Rewritten Instruction#.
#Given Instruction#:
{...}
#Key Part#:
{...}
#Input#:
{...}
#Output#:
{...}
#Rewritten Instruction#:
```

Table 5: Our templates for extracting key parts and evolving instructions.

Dataset	Task	Category	Training	Test	Held-out	Time Step
Sup-NatInst-ST	128	40	50,901	12,800	40	88
Sup-NatInst-SC	128	40	50,901	12,800	40	34
Domain-ST	20	12	52,000	10,000	7	13
Domain-SC	20	12	52,000	10,000	7	9

Table 6: Statistics of datasets.

we set M to be the number of historical tasks and N to be the same as **KPIG** in **PCLL** and **DCL**. For InsCL, we set the maximum replayed number to 200 (consistent with their publication).

As for the training time, our KPIG takes 340 minutes to complete the training on Sup-NatInst-ST dataset. Compared with **SFT** (200 minutes), the extra time cost is mainly in the calculation stage of information gain, which takes about 1 minutes for each time step. In addition, under the setting of $M = 10$ and $N = 10$, *replay*-based methods like **PCLL**, **DCL** and **DYNAINST** take about 300 minutes, and our time difference (40 minutes) is that the logits of the masked part need further calculation. **InsCL** takes about 440 minutes due to the setting of a larger replay number. However, to achieve the results reported in Table 1, **PCLL** and **DCL** need to replay all historical tasks, which takes about 400 minutes. The above analysis shows that our method offers a relatively balanced trade-off between performance and training efficiency under the setting of CIT.

D Dataset

Table 6 shows the details of our datasets. For Sup-NatInst, we have 40 held-out tasks. and we select 100 instances from each task based on Self-BLEU score and label distribution, which are used for evaluation. For Domain, we have 7 held-out tasks, and we select 500 instances from each task for evaluation. The difference between ST and SC settings lies in the time steps. The former trains a single task at each time step, while the latter trains all tasks of different category at different time step.

In addition, we list the details of each task in Table 7, Table 8, Table 9 and Table 10. We mark the evaluation method, format constraints, and response range for each task based on manual annotation. For example, `In choice` usually represents a classification task that must be selected from within the scope of instruction constraints. `In context + In entity type` represents a combination constraint on named entity recognition tasks, which means that entities of the given type must be extracted from the given context. Based on these manually annotated constraints, we can calculate `P-score` and `V-score` for all tasks.

Algorithm 1 Algorithm of our proposed KPIG

Input: A sequence of task sets $\mathcal{T}^T = \{\tau_1, \tau_2, \ldots, \tau_n\}_{t=1}^{T}$, initial foundation LLM $\mathcal{M}_0$, instruction-diversity cached pool $\mathcal{I}_\tau$ for each task τ

Output: Target LLM $\mathcal{M}_T$

$t \leftarrow t + 1$
while $t <= T$ **do**
  Replay task set $\mathcal{R} = \{\}$
  for each $\tau \in \mathcal{T}^{t'<t}$ **do**
    Randomly sample N instances and calculate IG for them ▷Eq. 1
    Calculate the average IG of N instances as the IG of task τ via $\mathcal{M}_{t-1}$
  end for
  Put the M tasks with the lowest IG into $\mathcal{R}$, $|\mathcal{R}| = M \times N$
  $\mathcal{T}^t \leftarrow \mathcal{T}^t \cup \mathcal{R}$
  Deepcopy $\mathcal{M}_t \leftarrow \mathcal{M}_{t-1}$
  Frozen $\mathcal{M}_{t-1}$
  for each instance $d_\tau = (i^{seed}, c, y) \in \mathcal{T}^t$ **do**
    Sample an instruction i for d_τ from $\mathcal{I}_\tau$
    Mask the key parts in i with `[MASK]` symbol to obtain i^m
    $d_\tau \leftarrow (i, c, y)$, $d_\tau^m \leftarrow (i^m, c, y)$
    Get output of d_τ via $\mathcal{M}_t$, and apply $\mathcal{L}_{CE}$ on it
    Get outputs of d_τ^m via $\mathcal{M}_{t-1}$ and $\mathcal{M}_t$, and apply $\mathcal{L}_{JSD}$ on them
    Optimize Loss ▷Eq. 4
  end for
  $t \leftarrow t + 1$
end while

E 6 Held-out Instructions

The six held-out instructions we used in our investigations on the information gain (§4.4) are listed in Table 12. The smcalflow classification task is a seen task, which requires determining whether the sentence is spoken by a user or an agent. Ins_6 has the smallest information gain and the worst model performance in Table 3. This may be because it is not concise enough and has more redundant constraints compared with other instructions, which

Name	Category	Metric	Scope	Format	Usage
mctaco_wrong_answer_generation_event_ordering	Wrong Candidate Generation	GPT	-	-	train
mctaco_grammatical_logical	Text Quality Evaluation	ACC	In choice	-	train
essential_terms_identifying_essential_words	Question Understanding	F1	-	Split by ,	train
multirc_classify_correct_answer	Answer Verification	ACC	In choice	-	train
squad11_question_generation	Question Generation	ROUGE	-	-	train
conala_remove_duplicates	Program Execution	Match	-	List	train
commongen_sentence_generation	Data to Text	ROUGE	-	-	train
story_cloze-rocstories_sentence_generation	Text Completion	ROUGE	-	-	train
zest_text_modification	Question Rewriting	ROUGE	-	-	train
detoxifying-lms_classification_fluency	Text Completion	ACC	In choice	-	train
afs_argument_quality_death_penalty	Text Matching	ACC	In choice	-	train
count_nouns_verbs	Pos Tagging	ACC	-	Number	train
snli_contradiction_to_entailment_text_modification	Sentence Composition	ROUGE	-	-	train
snli_classification	Textual Entailment	ACC	In choice	-	train
hotpotqa_sentence_generation	Explanation	ROUGE	-	-	train
iirc_link_exists_classification	Answerability Classification	ACC	In choice	-	train
stereoset_sentence_generation_antistereotype	Fill in The Blank	GPT	-	-	train
dream_incorrect_answer_generation	Wrong Candidate Generation	ROUGE	-	-	train
tellmewhy_question_answerability	Answerability Classification	ACC	In choice	-	train
(296)storycloze_correct_end_classification	Text Completion	ACC	In choice	-	train
(298)storycloze_correct_end_classification	Coherence Classification	ACC	In choice	-	train
numeric_fused_head_resolution	Coreference Resolution	ACC	In choice	-	train
stereoset_classification_profession	Stereotype Detection	ACC	In choice	-	train
jigsaw_classification_obscene	Toxic Language Detection	ACC	In choice	-	train
winomt_classification_gender_anti	Gender Classification	ACC	In choice	-	train
winomt_classification_profession_pro	Gender Classification	ACC	In choice	-	train
squad20_answerable_unanswerable_question_classification	Answerability Classification	ACC	In choice	-	train
winomt_classification_gender_identifiability_anti	Gender Classification	ACC	In choice	-	train
casino_classification_negotiation_vouch_fair	Negotiation Strategy Detection	ACC	In choice	-	train
inverse_causal_relationship	Cause Effect Classification	ACC	In choice	-	train
numeric_fused_head_reference	Coreference Resolution	ACC	In context	-	train
com_qa_paraphrase_question_generation	Question Rewriting	ROUGE	-	-	train
scruples_anecdotes_title_generation	Title Generation	ROUGE	-	-	train
senteval_odd_word_out	Linguistic Probing	ACC	In choice	-	train
aquamuse_answer_given_in_passage	Answerability Classification	ACC	In choice	-	train
udeps_eng_coarse_pos_tagging	Pos Tagging	ACC	In choice	-	train
multi_woz_classification	Speaker Identification	ACC	In choice	-	train
esnli_classification	Textual Entailment	ACC	In choice	-	train
extreme_abstract_summarization	Summarization	ROUGE	-	-	train

Table 7: Details of 1-40 task in the SupNatInst dataset.

Name	Category	Metric	Scope	Format	Usage
ambigqa_text_generation	Question Rewriting	ROUGE	-	-	train
mmmlu_answer_generation_computer_security	Question Answering	ACC	In choice	-	train
mmmlu_answer_generation_world_religions	Question Answering	ACC	In choice	-	train
protoqa_question_generation	Question Generation	ROUGE	-	-	train
copa_commonsense_reasoning	Cause Effect Classification	ACC	In choice	-	train
copa_commonsense_cause_effect	Cause Effect Classification	ACC	In choice	-	train
synthetic_multiply_evens	Program Execution	Match	-	List	train
synthetic_multiply_odds	Program Execution	Match	-	List	train
cfq_mcd1_explanation_to_sql	Text to Code	GPT	-	-	train
cfq_mcd1_sql_to_explanation	Text to Code	ACC	In choice	-	train
freebase_qa_topic_generation	Question Understanding	ROUGE	-	-	train
dialogre_identify_names	Speaker Identification	ACC	-	-	train
coached_conv_pref_classifier	Speaker Identification	ACC	In choice	-	train
defeasible_nli_atomic_classification	Textual Entailment	ACC	In choice	-	train
librispeech_asr_text_auto_completion	Text Completion	ROUGE	-	-	train
librispeech_asr_missing_word_prediction	Fill in The Blank	GPT	-	-	train
bard_analogical_reasoning_affordance	Word Analogy	GPT	Noun	-	train
bard_analogical_reasoning_travel	Word Analogy	GPT	-	-	train
bard_analogical_reasoning_trash_or_treasure	Word Analogy	GPT	-	-	train
penn_treebank_coarse_pos_tagging	Pos Tagging	ACC	In choice	-	train
atomic_classification_causes	Commonsense Classification	ACC	In choice	-	train
hrngo_quality_classification	Text Quality Evaluation	ACC	In choice	-	train
glue_mrpc_paraphrasing	Text Matching	ACC	In choice	-	train
wiki_qa_answer_verification	Answer Verification	ACC	In choice	-	train
amazonreview_summary_classification	Summarization	ACC	In choice	-	train
numer_sense_multiple_choice_qa_generation	Fill in The Blank	ACC	In choice	-	train
cb_entailment	Textual Entailment	ACC	In choice	-	train
wscfixed_coreference	Coreference Resolution	ACC	In choice	-	train
dart_question_generation	Data to Text	ROUGE	-	Contain _	train
gene_extraction_chemprot_dataset	Named Entity Recognition	F1	In context	-	train
chemical_extraction_chemprot_dataset	Named Entity Recognition	F1	-	One answer	train
hatexplain_classification	Toxic Language Detection	ACC	In choice	-	train
imppres_longtextgeneration	Sentence Composition	GPT	-	-	train
daily_dialog_question_classification	Dialogue Act Recognition	ACC	In choice	-	train
parsed_pdfs_summarization	Title Generation	ROUGE	-	-	train
scitail_classification	Textual Entailment	ACC	In choice	-	train
blimp_binary_classification	Linguistic Probing	ACC	In choice	-	train
bless_hypernym_generation	Word Semantics	ROUGE	-	-	train
scifact_title_generation	Title Generation	ROUGE	-	-	train

Table 8: Details of 41-80 task in the SupNatInst dataset.

Name	Category	Metric	Scope	Format	Usage
smcalflow_classification	Speaker Identification	ACC	In choice	-	train
disfl_qa_text_modication	Question Rewriting	GPT	-	-	train
medical_question_pair_dataset_text_classification	Text Matching	ACC	In choice	-	train
winobias_text_generation	Coreference Resolution	Match	In context	Split by ,	train
civil_comments_threat_classification	Toxic Language Detection	ACC	In choice	-	train
civil_comments_insult_classification	Toxic Language Detection	ACC	In choice	-	train
web_nlg_data_to_text	Data to Text	GPT	-	-	train
quartz_question_answering	Question Answering	ACC	In context	-	train
mctaco_wrong_answer_generation_absolute_timepoint	Wrong Candidate Generation	GPT	-	-	test
mctaco_span_based_question	Answerability Classification	ACC	In choice	-	test
winogrande_question_generation_person	Question Generation	GPT	-	-	test
ropes_story_generation	Story Composition	ROUGE	-	-	test
abductivenli_classification	Coherence Classification	ACC	In choice	-	test
scan_structured_text_generation_command_action_short	Text to Code	Match	In choice	Split by _	test
odd-man-out_classification_no_category	Word Semantics	ACC	In context	-	test
combinations_of_list	Program Execution	Match	-	2D list	test
rocstories_correct_ending_classification	Text Completion	ACC	In choice	-	test
rocstories_title_answer_generation	Title Generation	ROUGE	-	Length <= 3	test
dream_classification	Question Understanding	ACC	In choice	-	test
scruples_event_time	Text Categorization	ACC	In choice	-	test
stereoset_classification_race	Stereotype Detection	ACC	In choice	-	test
gap_answer_generation	Coreference Resolution	ACC	-	-	test
winomt_classification_gender_pro	Gender Classification	ACC	In choice	-	test
hybridqa_answer_generation	Pos Tagging	ACC	In choice	-	test
casino_classification_negotiation_small_talk	Negotiation Strategy Detection	ACC	In choice	-	test
grailqa_paraphrase_generation	Question Rewriting	ROUGE	-	-	test
persent_sentence_sentiment_verification	Sentiment Analysis	ACC	In choice	-	test
senteval_inversion	Linguistic Probing	ACC	In choice	-	test
mwsc_question_generation	Question Generation	ROUGE	-	-	test
scruples_anecdotes_whoiswrong_classification	Ethics Classification	ACC	In choice	-	test
argument_consequence_classification	Text Matching	ACC	In choice	-	test
glucose_cause_event_detection	Cause Effect Classification	GPT	-	-	test
google_wellformed_query_sentence_generation	Text Quality Evaluation	ACC	In context	-	test
mmmlu_answer_generation_international_law	Question Answering	ACC	In choice	-	test
glucose_reverse_cause_emotion_detection	Information Extraction	ROUGE	-	A >Causes> B	test
conv_ai_2_classification	Speaker Identification	ACC	In choice	-	test
gap_fill_the_blank_coreference_resolution	Coreference Resolution	ACC	In choice	-	test
defeasible_nli_snli_classification	Textual Entailment	ACC	In choice	-	test
bard_analogical_reasoning_causation	Word Analogy	GPT	-	-	test
atomic_classification_xneed	Commonsense Classification	ACC	In choice	-	test
atomic_answer_generation	Fill in The Blank	GPT	-	One answer	test
superglue_multirc_answer_verification	Answer Verification	ACC	In choice	-	test
dart_text_generation	Data to Text	GPT	-	-	test
drug_extraction_ade	Named Entity Recognition	F1	In context	-	test
scitail11_sentence_generation	Sentence Composition	ROUGE	-	-	test
daily_dialog_formal_classification	Dialogue Act Recognition	ACC	In choice	-	test
smcalflow_sentence_generation	Dialogue Generation	ROUGE	-	-	test
ethos_text_classification	Toxic Language Detection	ACC	In choice	-	test

Table 9: Details of 81-128 task in the SupNatInst dataset.

Name	Category	Metric	Scope	Format	Usage
sale_relevance	Relevance	ACC	In choice	Json	train
commodity_alignment	Alignment	Match	In choice	List	train
ingredient_identification	Identification	F1	In context	Json	train
recommendation	Recommendation	ACC	In choice	-	train
click_prediction	Recommendation	ACC	In choice	Json + Explanation	train
user_interest_mining	Mining	F1	In choice	List	train
recipe_generation	Generation	BLEU	-	Step 1 2 3	train
product_description_generation	Generation	BLEU	Contain center word	-	train
summary_generation	Generation	ROUGE	-	-	train
food_entity_extraction	Named Entity Recognition	F1	In context + In entity types	Json	train
comment_entity_extraction	Named Entity Recognition	F1	In context + In entity types	Json	train
text2sql	Code	ACC	-	Legal sql	train
merchant_classification	Classification	ACC	In choice	-	train
item_classification	Classification	ACC	In choice	-	test
logical_reasoning	Reasoning	ACC	In choice	Uppercase letter	test
conversation_completion	Completion	BLEU	-	Length <= 50	test
**_ner	Named Entity Recognition	F1	In context + In entity types	Json	test
property_rel	Relevance	ACC	In choice	-	test
post_extraction	Named Entity Recognition	ACC	In context + In entity types	Json	test
food_rewrite	Rewriting	GPT	-	Length <= 7	test

Table 10: Details of each task in the Domain dataset.

λ ($\alpha = 0.3$)	Seen P	Unseen P	α ($\lambda = 0.02$)	Seen P	Unseen P
0.001	54.7	37.1	0.05	51.9	41.7
0.01	52.6	40.8	0.1	51.6	42.1
0.02	52.2	42.5	0.3	52.2	42.5
0.05	49.6	43.3	0.6	51.3	41.4
0.1	47.6	44.0	1.0	48.6	38.5

Table 11: Performance on Sup-NatInst-ST dataset with varying λ and α.

may indicate that our information gain may be helpful in measuring the clarity of the task definition.

F Investigations on Saliency Score

We conduct further experiments on the saliency score (Wang et al., 2023a) of the ground-truth and key-part tokens. The saliency score is a crucial metric that indicates how much attention the model pays to relevant segments of user input. We randomly select two tasks from Sup-NatInst, which have significant constraints on the classification label space and the scope of generated content respectively. As shown in Figure 6, ground truth tokens (e.g., 'S' in 'Similar' or 'D' in 'Dissimilar') receive a higher saliency score in the last 4 layers compared to baselines. For the INIT foundation LLM, attention flow towards the ground-truth tokens from the key parts is more significant than other parts across most tasks. The above results

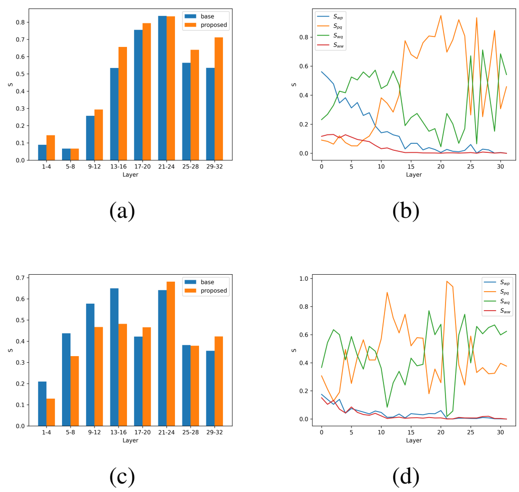

Figure 6: Saliency scores of task 1645 and 1664.

Ins_1
Instruction: In this assignment, you are presented with a dialogue segment, a piece of communication between a User and an Agent. Your responsibility is to identify the speaker. The User generally instructs the Agent to arrange activities, obtain event details, and inspect the timetable. In contrast, the Agent's reply is a response to the User's queries or additional inquiries based on the User's directive. Respond with "User" or "Agent". Additionally, it's important to note that the User may also ask the Agent to cancel events. **Key parts:** identify the speaker, User generally instructs the Agent to arrange activities, obtain event details, and inspect the timetable, Agent's reply is a response to the User's queries or additional inquiries based on the User's directive, Respond with "User" or "Agent", User may also ask the Agent to cancel events
Ins_2
Instruction: For this activity, you are presented with an excerpt from a dialogue involving a User and an Agent. It is your task to identify who is speaking. The User typically instructs the Agent to organize events, obtain data on events, or survey the event plan. In contrast, the Agent's replies often address the User's queries or extend the conversation based on the User's directives. Please respond with either "User" or "Agent". **Key parts:** identify who is speaking, User typically instructs the Agent to organize events, obtain data on events, or survey the event plan, Agent's replies often address the User's queries or extend the conversation based on the User's directives, Please respond with either "User" or "Agent"
Ins_3
Instruction: In this task, you will be presented with a statement, a fragment of a dialogue between a User and an Agent. Your responsibility is to identify the speaker. The User typically instructs the Agent to organize events, gather details about events, and verify the schedule. Conversely, the Agent's reply is a response to the User's inquiries or additional queries based on the User's directive. Respond with either "User" or "Agent". **Key parts:** identify the speaker, User typically instructs the Agent to organize events, gather details about events, and verify the schedule, Agent's reply is a response to the User's inquiries or additional queries based on the User's directive, "Respond with either "User" or "Agent"
Ins_4
Instruction: In this task, you are presented with a dialogue fragment, a piece of conversation between a User and an Agent. Your responsibility is to identify the speaker. The User typically instructs the Agent to arrange events, fetch details about events, and verify the schedule. Conversely, the Agent's reply is a response to the User's inquiries or additional queries based on the User's directive. Respond with either "User" or "Agent". **Key parts:** identify the speaker, User typically instructs the Agent to arrange events, fetch details about events, and verify the schedule, Agent's reply is a response to the User's inquiries or additional queries based on the User's directive, Respond with either "User" or "Agent"
Ins_5
Instruction: In this task, you are presented with a dialogue fragment from a conversation between a User and an Agent. Your responsibility is to identify the speaker. The User typically instructs the Agent to organize events, fetch details about events, and verify the schedule. Conversely, the Agent's reply is a response to the User's inquiries or additional queries based on the User's directive. Respond with either "User" or "Agent". **Key parts:** identify the speaker, User typically instructs the Agent to organize events, fetch details about events, and verify the schedule, Agent's reply is a response to the User's inquiries or additional queries based on the User's directive, Respond with either "User" or "Agent"
Ins_6
Instruction: In this assignment, you are presented with a snippet of a dialogue between a User and an Agent. The User typically instructs the Agent to organise events, gather details about an event, and inspect the agenda, whilst the Agent's reply consists of answers to the User's inquiries or additional questions pertaining to the User's directive. Your task is to identify the speaker from the dialogue snippet, taking into consideration the typical role of the User and the Agent, and to provide the speaker's identity as "User" or "Agent". Additionally, ensure your judgement is supported by reasonable analysis of the given dialogue. **Key parts:** identify the speaker from the dialogue snippet, taking into consideration the typical role of the User and the Agent, provide the speaker's identity as "User" or "Agent", ensure your judgement is supported by reasonable analysis of the given dialogue

Table 12: Six held-out instructions and corresponding key parts of the smcalflow classification task.

show that KPIG is more effective at identifying the key parts of user intent, leading to more accurate and reliable performance. However, compared to the classification-task 1645, such phenomenon is not so significant for generation-task 1664. This may be because the importance of context is not that different in such tasks, and it is difficult for GPT-4 to accurately identify the key parts (the instructions for such tasks have fewer constraints and definitions). And we believe these phenomena merit further exploration such as more carefully designed data construction and staged training.

G Details in Human Annotation

In this section, we show the details of manual annotation on the constraints and the metric for each task. We recruited 4 students aged 25 to 30 with computer background and proficient English communication skills. Since they are volunteers, they were not paid. We shuffled the data randomly and assigned data to them. The task is not included until at least 3 people have consistent annotations. Our annotation instruction is like: "Given the instruction of the task definition, two positive demonstrations, two negative demonstrations, and the corresponding explanations, mark out the format (such as JSON, separator, upper and lower case, numbers, letters and so on), length restrictions, inclusion of specified words, selection from a specified range, and other constraints that are critical to generating desired responses. And choose the most applicable metric from F1, ACC, ROUGE, BLEU, Match, and GPT". The above metrics are illustrated in §3.4, and we also give three demonstrations of applicable tasks for each metric.