InsCL: A Data-efficient Continual Learning Paradigm for Fine-tuning Large Language Models with instructions

Anonymous ACL submission

Abstract

Instruction tuning effectively optimizes Large Language Models (LLMs) for downstream tasks. Due to the changing environment in real-life applications, LLMs necessitate continual task-specific adaptation without catastrophic forgetting. Considering the heavy computational cost, replay-based Continual Learning (CL) methods are the simplest and most widely used for LLMs to address the forgetting issue. However, traditional replay-based methods do not fully utilize instructions to customize the replay strategy. In this work, we propose a novel paradigm called Instruction-based Continual Learning (InsCL). InsCL dynamically replays previous data based on task similarity, calculated by Wasserstein Distance with instructions. Moreover, we further introduce an Instruction Information Metric (InsInfo) to quantify the complexity and diversity of instructions. According to InsInfo, InsCL guides the replay process more inclined to high-quality data. We conduct extensive experiments over 16 tasks with different training orders, observing consistent performance improvements of InsCL. When all tasks have been trained, InsCL achieves performance gains of 3.0 Relative Gain compared with Random Replay, and 27.96 Relative Gain compared with No Replay.

1 Introduction

Large Language Models (LLMs) show remarkable capabilities from a wide range of Natural Language Processing (NLP) tasks (Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023), demonstrating large potential in handling various task-specific settings. To complete realistic downstream tasks, recent works suggest that instruction tuning is an incredible method for unleashing the power of LLMs (Wei et al., 2021; Peng et al., 2023; Shi et al., 2023). However, in real-life applications, the consistent emergence of new corpora and knowledge changes task schemas frequently, necessitating continual task-specific adaptation for LLMs (Jin et al., 2021; Daruna et al., 2021). Accordingly, Continual Learning (CL) is proposed to learn a sequence of tasks incrementally, updating models for the changing environment without catastrophic forgetting (Goodfellow et al., 2013; Kemker et al., 2018).

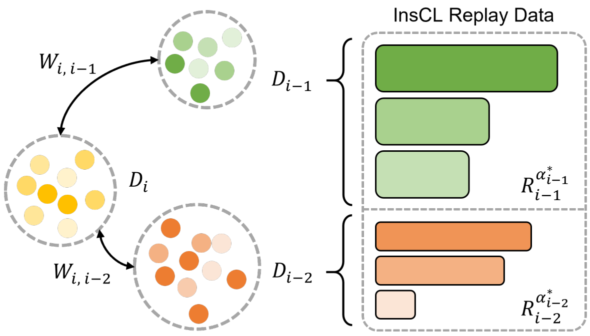

Figure 1: The framework of InsCL, the index denotes task id. D represents task data, and R represents the sampled data to replay. InsCL dynamically replays α^* data for each previous task based on the task similarity calculated via Wasserstein Distance W. The dots represent instructions included in each task, and the darker colors represent higher InsInfo. The size of each color bar denotes the corresponding amount of replay data.

Considering the heavy burden on computing time and GPU memory of tuning LLMs, replay-based methods are the simplest and most effective among all traditional CL methods. Despite several replay-based methods that have been well-studied (Sun et al., 2019; Wang et al., 2020; Mi et al., 2020; Qin et al., 2022), some traditional strategies cannot achieve optimal performance in continual instruction tuning due to the unique data composition. To address this issue, we propose a data-efficient paradigm called **Ins**truction-based **C**ontinual **L**earning (InsCL), applied to continual fine-tuning LLMs with natural language instructions. InsCL effectively utilizes instructions as high-quality task descriptions, designing a dynamic instruction-information-based replay method. As shown in Figure 1, when the new task D_i comes, InsCL will sample replay data R from all the previous tasks (here we list two previous tasks in Figure 1).

InsCL dynamically replays α^* data from previous tasks based on their similarity with the current task. We draw on the application of Optimal Transport (Torres et al., 2021) in comparing different distributions and adopt Wasserstein Distance (Liu et al., 2022) as a similarity measure. Since instructions naturally contain high-quality task-related descriptions, we use instructions to calculate Wasserstein Distance instead of using the full amount of data, significantly reducing the computational cost (Cuturi, 2013). For the previous tasks that are more different from the current task, InsCL allocates a larger replay scale (**larger bar width** in Figure 1).

After determining the sample size based on task similarity, InsCL leverages instruction information to guide the sampling process more inclined to high-quality data. Prior works have shown that the performance with less but high-quality data can be comparable with full data (Toneva et al., 2018; Abbas et al., 2023; Tirumala et al., 2023). For instruction tuning scenarios, early attempts (Wang et al., 2022a; Xu et al., 2023a; Ding et al., 2023) affirm that LLMs' performance can be improved by increasing the training template complexity and diversity. Inspired by this, we propose an Instruction Information Metric (InsInfo) to quantify the complexity and diversity of instructions. With InsInfo-guided sampling, InsCL replays more high-quality data (**longer bar length** in Figure 1). We empirically demonstrate that replaying more data with high InsInfo helps to alleviate the forgetting issue.

The main contributions of this paper include: (1) We propose InsCL, a novel replay-based CL paradigm for instruction tuning. InsCL allocates replay size based on task similarity, dynamically replaying high-quality data with high InsInfo. (2) Experiments are conducted over 16 tasks with different training orders, demonstrating the effectiveness of InsCL. (3) We further analyze the forgetting phenomenon in continual instruction tuning. Without replying, we found that complex reasoning tasks suffer from a higher forgetting rate, where forgetting instances are mainly instruction-unrelated.

Instruction : In this task, you're given reviews from Amazon's products. Your task is to generate the Summary of the review.

Input : Totally screwed up my system. Instructions terrible. Disk gives long list of files, had to determine what does what. Has already wasted 4 hours of my time. I gave up and pulled the thing. Don't buy this.

Output : Terrible. Instructions are non-existent.

Table 1: A case of data template in instruction tuning.

2 Related Work

2.1 Instruction Tuning

Recently, LLMs have demonstrated impressive performance across various NLP tasks. After being unsupervised pre-trained on large-scale raw text, LLMs are further trained via instruction tuning to generate appropriate outputs based on the given input instructions (Sanh et al., 2021; Mishra et al., 2021; Chung et al., 2022). Prior works supervised fine-tuned (SFT) LLMs with datasets consisting of {instruction, input, output} pairs, as shown in Table 1, and evaluated on another set of held-out tasks (Wei et al., 2021; Longpre et al., 2023). They demonstrate that the performance of unseen tasks can be improved with more tasks and templates. To improve the diversity and complexity of instruction, a broad range of open-source instruction tuning datasets are proposed. Some are gathered through crowd-sourcing (Conover et al., 2023; Zhou et al., 2023) while others are distilled from strong proprietary models (Wang et al., 2022a; Peng et al., 2023; Taori et al., 2023).

With the help of various low-cost methods of constructing high-quality templates, instruction datasets can expand easily over time as new tasks appear. When the data scale grows dynamically, we can easily obtain sufficient task-specific data. Considering this, rather than evaluating zero-shot ability on held-out tasks, we are more concerned about adapting an instruction-tuned model to a new task without suffering from catastrophic forgetting. In this work, we fine-tune LLMs in a continuous manner and analyze their performance on previous tasks, aiming to explore the forgetting issue in a changeable environment.

2.2 Traditional CL Methods

CL aims to learn a sequence of tasks incrementally without forgetting the previously learned knowledge. Early attempts in CL can be generally divided into three categories: (1) **Consolidation-based methods** aim at protecting important parameters. As the representative of the regularization sub-family, EWC (Kirkpatrick et al., 2017) constrains the loss based on parameter importance calculated by the fisher information matrix. Several works distill the model from the previous stage to keep relevant knowledge (Zhang et al., 2020; Monaikul et al., 2021; Liu et al., 2021; Qin and Joty, 2021). (2) **Architecture-based methods** add task-specific parameters to the base model for each task (Rusu et al., 2016; Gu et al., 2020; Madotto et al., 2020). By separating trainable parameters, the model can mitigate the impact on old tasks when updating parameters. However, the model scale grows linearly when tasks increase, bringing inevitable memory costs. (3) **Replay-based methods** store a small subset of previous training examples and replay when the new task comes. Sun et al. (2019); Zhang et al. (2022) leverage language models to generate pseudo-examples for previous tasks, but the quality of examples cannot be guaranteed (Ke et al., 2021).

Despite the success of traditional CL methods, their backbones are relatively small in scale, such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019). Under LLMs' full fine-tuning scenarios, consolidation-based and architecture-based methods will bring additional parameter storage and training costs. Considering the heavy burden on computing time and GPU memory, replay-based CL methods are the simplest and most widely used in tuning LLMs as data-efficient methods that do not change the model structure.

2.3 CL for LLMs instruction tuning

Due to the scaling laws for neural language models, LLMs emerge with capabilities when the scale increases. They can be better adapted to various downstream tasks through instruction tuning, offering immense practical value in real-world applications. The exploration of CL for LLMs is still in its early stages. Continual-T0 (Scialom et al., 2022) first fine-tuned LLMs with instructions in an incremental manner, claiming that well-pretrained models can be continual learners by randomly replaying several previous examples. Several works (Song et al., 2023; Wang et al., 2023) focus on CL methods with parameter-efficient tuning (Hu et al., 2021), largely alleviating the forgetting issue under limited training resources. For full fine-tuning, replay-based methods were preliminarily investigated (Yin et al., 2023), proving that replaying data based on diverse instructions can alleviate catastrophic forgetting and help better generalize to unseen tasks. However, there is still a lack of detailed analysis of replay strategies.

In this work, we focus on the appropriate replay-based method for LLMs' full fine-tuning with instructions. Considering that instructions naturally provide high-quality task-related descriptions, it is necessary to fully utilize instruction information to customize a replay strategy for instruction tuning.

3 Method

Continual Learning of LLMs focuses on adapting an instruction-tuned model to handle a sequence of tasks in a specific application scenario. This approach accounts for consistently emerging materials while processing the tasks simultaneously. We define n tasks to be learned as a sequence $D = \{D_1, \ldots, D_n\}$. When LLMs are tuned with i-th task, we form a replay dataset R_j^α by sampling examples from D_j, where $j \in [1, i-1]$. Formally, the training data augmented with replay data is defined as:

$$D_i^\alpha = D_i \cup \sum_{j=1}^{i-1} R_j^\alpha$$

where α is the replay hyper-parameter, controlling the sampling quantity from previous tasks.

3.1 Dynamic Replay

Prior works optimize CL methods based on the similarity between previous tasks and the current one (Mi et al., 2020; Xu et al., 2023b; Gogoulou et al., 2023). As the similarity increases, it becomes easier to retain knowledge from previous tasks. Inspired by this, we propose a dynamic replay strategy based on task similarity, replaying more data from previous tasks with large differences.

The concept of task similarity is at the core of various machine learning paradigms, such as domain adaptation and meta-learning. Optimal Transport (Alvarez-Melis and Fusi, 2020; Torres et al., 2021) offers a way to calculate the least amount of cost for transferring between different distribution pairs. As the representative of the Optimal Transport framework, Wasserstein Distance (Chen et al., 2022; Liu et al., 2022) provides a metric for calculating the similarity between two dataset distributions. The definition of Wasserstein Distance is as follows:

$$W(\mu_A, \mu_B) = \inf_{\pi} \left(\int_{\mathbb{R}} d(x_A, x_B) d\pi(x_A, x_B) \right)$$

where $\pi \in \prod(\mu_A, \mu_B)$ is meant to be the set of all joint probabilities that exhibit μ_A and μ_B as marginal distributions. The d denotes a metric for calculating the cost matrix, and here we define it as the cosine distance. For instruction tuning, NLP tasks can be described via natural language instructions. We consider the instruction embeddings for a task pair as x_A and x_B, and calculate the proportion of instructions for each task as a probability distribution. Consequently, we measure task similarity by calculating their Wasserstein Distance. When LLMs are fine-tuned on the current task D_i, the amount of dynamic replay data for the j-th previous task is defined as:

$$\alpha_j^* = \frac{W_{j,i}}{\sum_{k=1}^{i-1} W_{k,i}} \times \alpha, \quad j \in [1, i-1]$$

where $W_{j,i}$ denotes the Wasserstein Distance between D_j and D_i. We dynamically allocate the amount of previous data to replay according to its similarity with the current task. With the help of dynamic replay, LLMs selectively recall the corresponding knowledge.

3.2 Instruction Information Metric

It has been proven that a small amount of high-quality data can achieve a promising performance, demonstrating the rationality of careful data selection (de Masson D'Autume et al., 2019; Wang et al., 2020; Ke and Liu, 2022; Zhou et al., 2023). Inspired by this, we propose an Instruction Information Metric (InsInfo) to guide the sampling process, collecting high-quality replay data for continual instruction tuning.

Considering **complex** and **diverse** instructions induce impressive performance, a more comprehensive analysis of multiple intentions embedded within instructions is necessary. High-performing open-source LLMs demonstrate the ability to annotate queries with tag entities, and the precision and consistency are proven through manual annotation (Lu et al., 2023). Consequently, we employ GPT-4 (OpenAI, 2023) as an intention tagger and clean the raw tags, representing instructions at a fine-grained entity level. The detailed process of obtaining normalized tags is shown in Appendix A.1. After obtaining fine-grained annotations for instructions, we utilize the number and frequency of tags as quantifiable indicators of diversity and complexity. Motivated by Inverse Document Frequency (IDF), one of the most useful and widely used concepts in information retrieval (Gupta et al., 2022; Tayal et al., 2023), we

Algorithm 1: InsInfo-guided sampling

Data: Dataset D_j, Instruction Pool I_i, Replay Number α
Result: Replay dataset R_j^α

Initialize Empty R_j^α and InsInfo List S_j;
Extract task j instruction set I_j from I_i;
for *Query* $I_{j,k} \in I_j$ **do**
  $s_{j,k} \leftarrow$ calculate InsInfo for $I_{j,k}$;
  $S_j \leftarrow S_j \cup s_{j,k}$;
end
for $k = 1$ *to* $|I_j|$ **do**
  $\beta \leftarrow \frac{s_{j,k}}{sum(S_j)} \times \alpha$;
  $D_{j,k} \leftarrow$ {data in D_j with $I_{j,k}$} ;
  $R_j^\alpha \leftarrow$ sample β data from $D_{j,k}$;
end
return R_j^α

proposed InsInfo as follows to quantify instruction information:

$$\text{InsInfo} = \sum_{t=1}^{T} log \frac{N}{f_t}$$

where N denotes the total amount of previous instructions. When tasks come into a stream, we store all previous instructions in memory. For each instruction, T denotes the number of tags, and f_t denotes the frequency of the t-th tag among the instruction pool. Hence, instruction gets a large InsInfo when the number of individual tags increases, quantifying complexity and diversity interpretably. As shown in Algorithm 1, we follow the InsInfo-guided sampling strategy to obtain the replay data. Moreover, the strategy can be combined with dynamic replay by modifying α to α_j^*, as claimed in Section 3.1, which forms our InsCL finally.

4 Experimental Setup

Data Collection. To facilitate our research, we mainly utilize the SuperNI dataset (Wang et al., 2022b), a comprehensive benchmark focusing on specific NLP tasks distilled from real-world demands. SuperNI is annotated by NLP practitioners from GitHub and NLP courses, ensuring that each instance is coupled with respective natural language instructions. At the most comprehensive level, we integrate 765 English tasks from SuperNI into 16 categories, as shown in Figure 2. And we demonstrate details of the data composition in Appendix A.2. Following the setting of prior CL

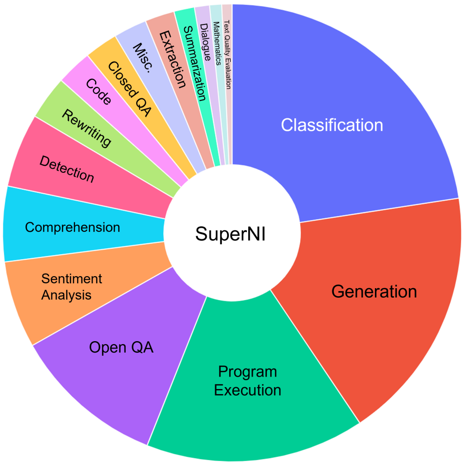

Figure 2: We obtain 16 categories by integrating English tasks in the SuperNI dataset. And we conduct further experiments based on 16 reallocated tasks.

studies (Scialom et al., 2022; Yin et al., 2023), we randomly hold out 20% instances on each task to evaluate LLMs on different training stages.

Model and Training Details. Our work is most related to the continual instruction tuning setting as Continual-T0 (Scialom et al., 2022). We conduct our task-incremental experiments with the popular LLaMA-7B (Touvron et al., 2023), training each task for 2 epochs with a batch size of 64. We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 2e-5 and utilize the standard language modeling objective:

$$\mathcal{L} = -\frac{1}{|y|}\sum_{i=1}^{|y|}\log p_\theta\left(y_i \mid x, y_{<i}\right)$$

where x denotes the combination of instruction and input, and y denotes the corresponding output.

Evaluate Forgetting. Following the evaluation metric proposed by Scialom et al. (2022), we leverage Relative Gain to focus on the forgetting issue. We train expert LLM on each single task only and test with their respective holdout data, taking the results as upper bounds (Jang et al., 2023). The Relative Gain in stage i can be defined as:

$$\text{Relative Gain}^i = \frac{1}{i-1}\sum_{j=1}^{i-1}\frac{R_j^i}{\text{upper bound}_j} \times 100\%.$$

Here we utilize Rouge-L (Lin, 2004) to calculate R_j^i and the upper bound.

5 Experiments

We leverage LLaMA-7B to calculate sentence embeddings and compare our InsCL with the following strategies:

- **No Replay**: Train LLMs incrementally without any replay data.
- **Random Replay**: Sample α instances randomly from each previous task as the replay setting in Continual-T0.
- **Prototype Data**: To collect the most representative data, we cluster the training data embedding space with k-means (Wang et al., 2021a). For each previous task, we set the cluster number as the amount of instructions. We sort the data in descending order according to cosine distance from the corresponding center and take the top-α as replay data.
- **Prototype Instruction**: We cluster instructions on previous tasks with the optimal silhouette coefficient (Dinh et al., 2019), taking the closest instructions to their respective centers as the most representative. We randomly select α data with prototypical instructions.
- **Diverse Instruction**: Following the optimal replay strategy proposed by Yin et al. (2023), we replay data with instructions diverging most from the current task instructions. By computing the cosine similarity matrix with the current instruction embedding, we take the most diverse instruction with the least column sum and replay α corresponding data for each previous task.

For fairness of comparison among different methods, we note $M_i = (i-1) \times \alpha$ as the total amount of replay data when the task sequence comes to stage i. Here we set α to 200.

5.1 Main Results

We train LLaMA-7B on 16 tasks continuously with three different training orders. For each continual instruction tuning stage, the average Relative Gain results are shown in Figure 3. It can be observed that our InsCL is effective in mitigating forgetting, with a promising Relative Gain. When all tasks have been trained, InsCL achieves performance gains of 3.0 Relative Gain compared with Random Replay, and 27.96 Relative Gain compared with No Replay. InsCL sustainably maintains the performance on previous tasks over 90%, exhibiting high stability with a small fluctuation. Conversely, No Replay's Relative Gain shows a sharp decreasing trend as the task increases, accompanied by significant performance fluctuations. After training the 8th task, No Replay's performance remains at less

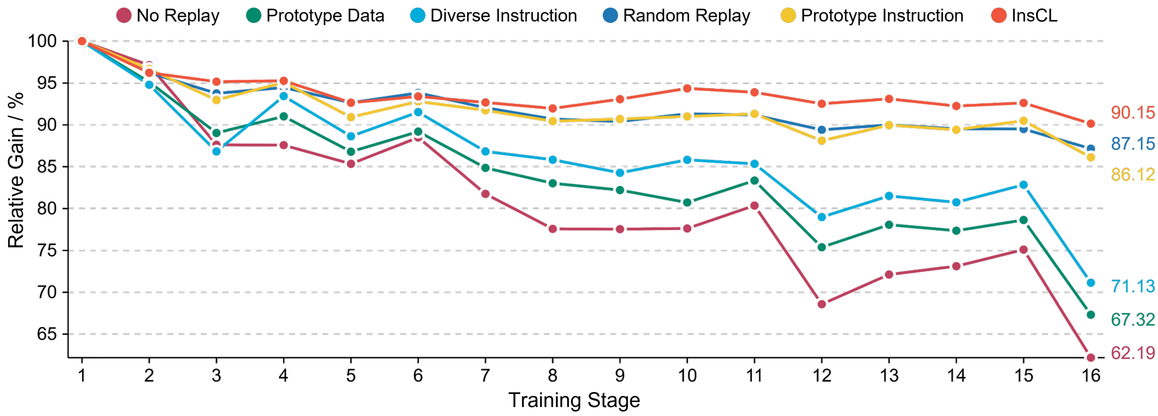

Figure 3: Progressive Relative Gain results for LLaMA-7B in continual instruction tuning. We set Relative Gain to 100 for training on the first task, denoting the initial performance without forgetting. When it comes to stage i, we plot the average score of corresponding Relative Gain with three different training orders. The closer the Relative Gain is to 100, the better to alleviate catastrophic forgetting and preserve knowledge.

	Reverse		Random		Curriculum	
Method	**AVG**	**STD**	**AVG**	**STD**	**AVG**	**STD**
No Replay	73.83	182.87	81.07	121.9	87.63	51.30
Random Replay	87.96	18.85	92.90	<u>10.84</u>	<u>95.18</u>	<u>4.80</u>
Prototype Data	78.07	92.71	83.51	93.71	90.07	29.79
Prototype Instruction	<u>88.29</u>	<u>15.73</u>	<u>93.01</u>	18.75	93.91	7.44
Diverse Instruction	80.87	72.09	86.47	81.60	91.14	23.34
InsCL	**90.50**	**9.32**	**94.43**	**7.62**	**96.20**	**2.81**

Table 2: Results on different training orders. AVG indicates average Relative Gain on 16 tasks, and STD indicates standard deviation ($\times$ e-4) on all the Relative Gain. Reverse denotes a converse training order with Curriculum. A promising method is expected with a large AVG and a small STD, indicating good performance and high stability. The best results are in bold, while the second-best are underlined.

than 80% and further drops to less than 65% upon finishing final training. No Replay setting severely suffers from catastrophic forgetting, demonstrating the necessity of replaying previous data.

Moreover, we further analyze other replay-based methods. Despite being the optimal method in the previous work, Diverse Instruction underperforms when compared with Random Replay and Prototype Instruction. For prototype-based methods, Prototype Instruction outperforms Prototype Data. We find that clustering results of Prototype Data are significantly affected by instances with long instruction and short input, leading to practically identical embeddings for this subset. The uneven distribution will cause a high semantic duplicate selection, which has been proven to lead to a negative impact (Abbas et al., 2023). The data composed of instruction and input has a different structure from traditional SFT, resulting in several traditional replay-based methods not being directly applicable to instruction tuning. This observation also demonstrates the rationality of designing instruction-based replay methods, proving the consistency of our InsCL.

5.2 Training Order Analysis

To explore the impact of training order and obtain universe conclusions, we conduct a detailed analysis of all settings based on different task sequences. Inspired by Curriculum Learning (Wang et al., 2021c), we train the model from easy task to hard task by sorting the upper bounds in descending order, as *Classification* $\rightarrow$ *Text Quality Evaluation* $\rightarrow$ *Code* $\rightarrow$ *Detection* $\rightarrow$ *Sentiment Analysis* $\rightarrow$ *Comprehension* $\rightarrow$ *Closed QA* $\rightarrow$ *Extraction* $\rightarrow$ *Dialogue* $\rightarrow$ *Program Execution* $\rightarrow$ *Rewriting* $\rightarrow$ *Open QA* $\rightarrow$ *Misc.* $\rightarrow$ *Generation* $\rightarrow$ *Summarization*$\rightarrow$ *Mathematics.*

As shown in Table 2, we report the average Relative Gain scores and the standard deviations on 16 tasks with different training orders. When we utilize the "easy to hard" training strategy, Cur-

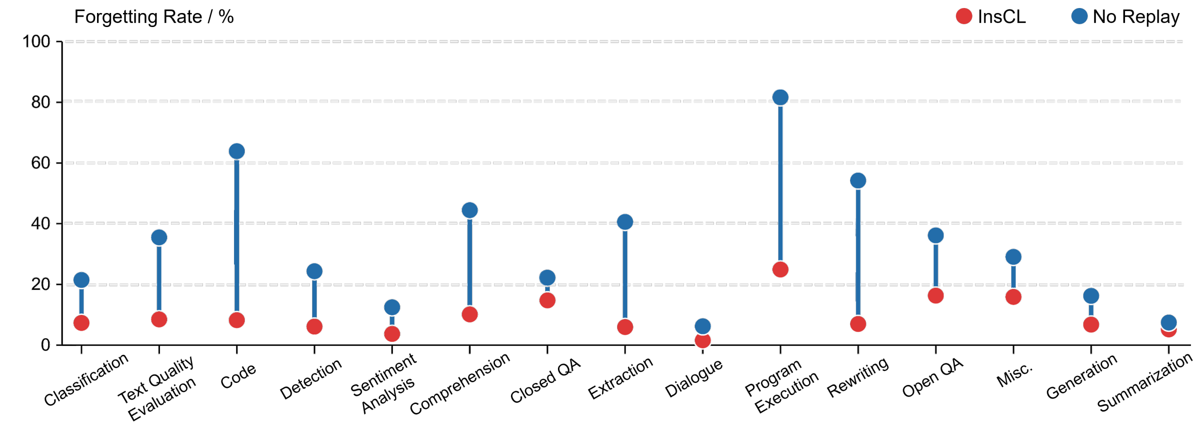

Figure 4: We analyze the forgetting rate based on Curriculum training order. The results of all previous tasks are reported when training is finished on the last task.

Method	AVG	STD
No Replay	80.84	118.69
Random Replay	92.01	11.50
+ Dynamic (Uniform)	93.14	8.67
+ Dynamic (Real)	93.25	8.57
+ InsInfo	93.52	17.90
InsCL	**93.71**	**6.58**

Table 3: Average results on three training orders. AVG indicates average Relative Gain, and STD indicates standard deviation ($\times$ e-4) on all the Relative Gain. The best results are in bold, while the second-best are underlined.

riculum outperforms other orders in all CL methods. Under the No Replay setting, Curriculum achieves performance gains of 13.80 average Relative Gain compared with Reverse and 6.56 compared with Random. Training tasks in Curriculum order demonstrates a more stable performance with a small standard deviation. Moreover, with our InsCL, Curriculum achieves performance gains of 5.70 average Relative Gain compared with Reverse and 1.77 compared with Random. It can be observed that InsCL alleviates the impact of different training orders, outperforming all methods with a high Relative Gain and stability.

5.3 Ablation Study

To investigate the effectiveness of each component in InsCL, we further apply our dynamic replay and InsInfo-guided sampling based on the Random Replay. Dynamic replay is determined by task similarity, calculated via Wasserstein distance. If the real distribution of instructions cannot be obtained, the uniform distribution assumption is generally used to obtain the Wasserstein distance. We evaluate the performance with average Relative Gain scores and standard deviations on all training stages.

The average results over three different training orders are reported in Table 3. It can be inferred that dynamic replay and InsInfo-guided sampling are both beneficial to mitigating catastrophic forgetting. InsInfo-guided sampling brings greater improvement in Relative Gain, effectively improving Relative Gain but lacking in stability. Instead, dynamic replay greatly reduces the standard deviation of Relative Gain thus improving stability. And dynamic replay with real distribution brings better performance compared with the uniform distribution assumption. When we utilize InsCL combined with dynamic replay and InsInfo-guided sampling, it achieves the best performance and strongest stability. Compared with Random Replay, InsCL delivers an improved average Relative Gain of 1.71 and a reduced standard deviation of 4.92. Furthermore, when compared with No Replay, InsCL achieves an improved average Relative Gain of 12.87 and a dramatic reduction of the standard deviation. The results prove the effectiveness of each component and demonstrate that InsCL leverages the strengths of each.

5.4 Forgetting Analysis

Forgetting Rate. For a further catastrophic forgetting analysis, several methods (Kemker et al., 2018; Luo et al., 2023) quantify the forgetting issue by evaluating performance decrease as training incrementally. Consequently, we propose a forgetting rate defined as:

$$FG_i = \frac{R_i^* - R_i^{-1}}{R_i^*} \times 100\%$$

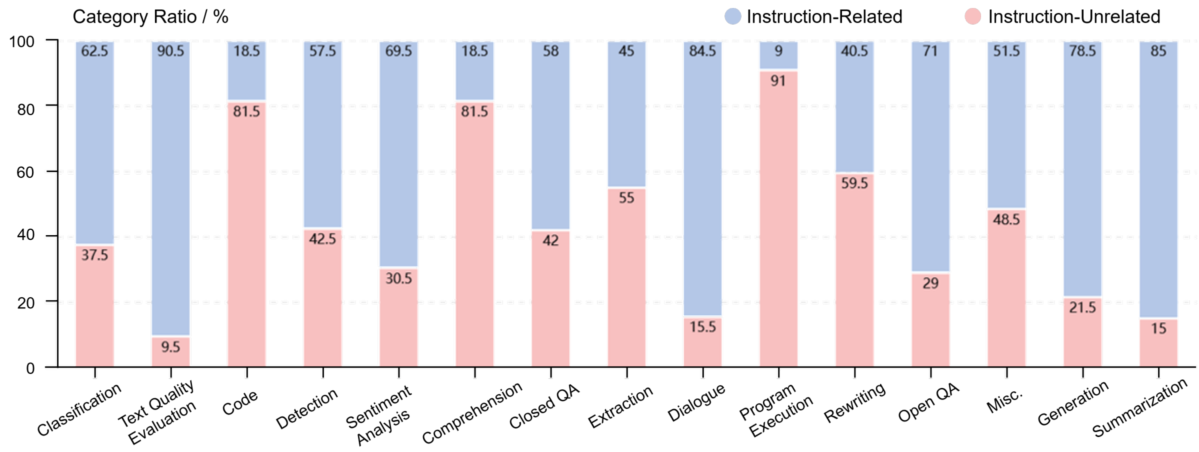

Figure 5: The analysis of forgetting category. We divide forgetting instances into Instruction-Related and Instruction Unrelated. After training on Curriculum order, the ratios of two categories in previous tasks are reported.

where R_i^* is the initial Rouge-L of task i after training on the corresponding task, and R_i^{-1} is the final Rouge-L of task i in the last training stage.

We evaluate the forgetting rate with Curriculum training order and report the results of No Replay and InsCL in Figure 4. It can be inferred that there is no inevitable relationship between task order and forgetting rate. For tasks that require complex reasoning, Program Execution and Code severely suffer from forgetting with the No Replay setting. Additionally, a large training data scale does not necessarily lead to a small forgetting rate. For example, Classification and Generation are the top-2 tasks with large training data and exhibit smaller forgetting rates, while Program Execution with the third largest dataset suffers from the largest forgetting rate. With our InsCL, the forgetting rates of almost all tasks are below 20%, which means that most of the previous knowledge is preserved.

Forgetting Category. When all the tasks have been trained under the No Replay setting, we collect previous tasks' instances with a decreased Rouge-L, called forgetting instances. We randomly sampled 200 forgetting instances from each previous task, manually analyzing the forgetting category for a detailed conclusion. We divide forgetting instances into two categories based on the instruction's following ability: (1) Instruction-Related: The output is relevant to the instruction, according to the space defined by the instruction. This category indicates LLMs do not forget the corresponding instruction following ability. (2) Instruction-Unrelated: The output is unrelated to the instruction. We demonstrate representative cases and respective explanations in Appendix A.3.

Figure 5 reports category ratios in the curriculum training order. The forgotten instances of most tasks are mainly Instruction-Related, while the forgetting instances in 5 tasks are mainly Instruction-Unrelated. Additionally, more than 80% of forgetting instances in Program Execution, Code, and Comprehension tasks are Instruction-Unrelated. It can be inferred that failure to understand instructions mainly leads to the performance decline of complex reasoning tasks.

6 Conclusions

In this paper, we mainly discuss the efficient adaptation of LLMs to continual downstream tasks with instructions. Replay-based CL methods do not require additional modifications to LLMs and fully utilize previous data, mitigating catastrophic forgetting effectively. We proposed InsCL, an effective data-efficient method to mitigate catastrophic forgetting for LLMs instruction tuning. InsCL is a model-agnostic and training-free method, indicating strong transferability. Different from existing replay-based methods, we fully utilize instructions as representative task descriptions to design the replay strategy. InsCL leverages instruction embeddings and distributions to calculate Wasserstein distance for task similarity, adjusting the replay ratio dynamically. Then, with our InsInfo-guided sampling, InsCL selects more high-quality data with complex and diverse instructions. We conduct extensive experiments over 16 tasks with different training orders, observing consistent performance improvements of InsCL. Additionally, we further analyze the forgetting rate and forgetting category, aiming to provide a guideline for future work.

7 Limitations

The promising performance demonstrated by InsCL is dependent on high-quality instructions. Instead, fuzzy instructions can affect the calculation of task similarity and the InsInfo-guided sampling, which may mislead our InsCL. However, if the instruction-based dataset is unsatisfied, the performance of tuned LLMs will also be greatly affected. Therefore, we tend to use our method after collecting high-quality instruction-based data to further mitigate catastrophic forgetting.

References

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*.

David Alvarez-Melis and Nicolo Fusi. 2020. Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems*, 33:21428–21439.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yao Chen, Qingyi Gao, and Xiao Wang. 2022. Inferential wasserstein generative adversarial networks. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):83–113.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26.

Angel Daruna, Mehul Gupta, Mohan Sridharan, and Sonia Chernova. 2021. Continual learning of knowledge graph embeddings. *IEEE Robotics and Automation Letters*, 6(2):1128–1135.

Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems*, 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Duy-Tai Dinh, Tsutomu Fujinami, and Van-Nam Huynh. 2019. Estimating the optimal number of clusters in categorical data clustering by silhouette coefficient. In *Knowledge and Systems Sciences: 20th International Symposium, KSS 2019, Da Nang, Vietnam, November 29–December 1, 2019, Proceedings 20*, pages 1–17. Springer.

Evangelia Gogoulou, Timothée Lesort, Magnus Boman, and Joakim Nivre. 2023. A study of continual learning under language shift. *arXiv preprint arXiv:2311.01200*.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Xiaotao Gu, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. 2020. On the transformer growth for progressive bert training. *arXiv preprint arXiv:2010.12562*.

Ishu Gupta, Sloni Mittal, Ankit Tiwari, Priya Agarwal, and Ashutosh Kumar Singh. 2022. Tidf-dlpm: Term and inverse document frequency based data leakage prevention model. *arXiv preprint arXiv:2203.05367*.

Michael Hahsler, Matthew Piekenbrock, and Derek Doran. 2019. dbscan: Fast density-based clustering with r. *Journal of Statistical Software*, 91:1–30.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2023. Exploring the benefits of training expert language models over instruction tuning. *arXiv preprint arXiv:2302.03202*.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2021. Lifelong pretraining: Continually adapting language models to emerging corpora. *arXiv preprint arXiv:2110.08534*.

Zixuan Ke and Bing Liu. 2022. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22443–22456.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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.

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

Qingbin Liu, Xiaoyan Yu, Shizhu He, Kang Liu, and Jun Zhao. 2021. Lifelong intent detection via multi-strategy rebalancing. *arXiv preprint arXiv:2108.04445*.

Xinran Liu, Yikun Bai, Yuzhe Lu, Andrea Soltoggio, and Soheil Kolouri. 2022. Wasserstein task embedding for measuring task similarities. *arXiv preprint arXiv:2208.11726*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. *arXiv e-prints*, pages arXiv–2308.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, and Zhiguang Wang. 2020. Continual learning in task-oriented dialogue systems. *arXiv preprint arXiv:2012.15504*.

Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. 2020. Continual learning for natural language generation in task-oriented dialog systems. *arXiv preprint arXiv:2010.00910*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Natural instructions: Benchmarking generalization to new tasks from natural language instructions. *arXiv preprint arXiv:2104.08773*, pages 839–849.

Natawut Monaikul, Giuseppe Castellucci, Simone Filice, and Oleg Rokhlenko. 2021. Continual learning for named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Chengwei Qin and Shafiq Joty. 2021. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. *arXiv preprint arXiv:2110.07298*.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Elle: Efficient lifelong pre-training for emerging data. *arXiv preprint arXiv:2203.06311*.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

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*, pages 6107–6122.

Chufan Shi, Yixuan Su, Cheng Yang, Yujiu Yang, and Deng Cai. 2023. Specialist or generalist? instruction tuning for specific nlp tasks. *arXiv preprint arXiv:2310.15326*.

Chenyang Song, Xu Han, Zheni Zeng, Kuai Li, Chen Chen, Zhiyuan Liu, Maosong Sun, and Tao Yang. 2023. Conpet: Continual parameter-efficient tuning for large language models. *arXiv preprint arXiv:2309.14763*.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. *arXiv preprint arXiv:1909.03329*.

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.

Madhuri A Tayal, Vanshika Bajaj, Ankita Gore, Preeti Yadav, and Vaishnavi Chouhan. 2023. Automatic domain classification of text using machine learning. In *2023 International Conference on Communication, Circuits, and Systems (IC3S)*, pages 1–5. IEEE.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari S Morcos. 2023. D4: Improving llm pretraining via document de-duplication and diversification. *arXiv preprint arXiv:2308.12284*.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2018. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*.

Luis Caicedo Torres, Luiz Manella Pereira, and M Hadi Amini. 2021. A survey on optimal transport for machine learning: Theory and applications. *arXiv preprint arXiv:2106.01963*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Chengyu Wang, Haojie Pan, Yuan Liu, Kehan Chen, Minghui Qiu, Wei Zhou, Jun Huang, Haiqing Chen, Wei Lin, and Deng Cai. 2021a. Mell: Large-scale extensible user intent classification for dialogue systems with meta lifelong learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 3649–3659.

Shufan Wang, Laure Thompson, and Mohit Iyyer. 2021b. Phrase-bert: Improved phrase embeddings from bert with an application to corpus exploration. *arXiv preprint arXiv:2109.06304*.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023. Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021c. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022a. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

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

Zirui Wang, Sanket Vaibhav Mehta, Barnabás Póczos, and Jaime Carbonell. 2020. Efficient meta lifelong-learning with limited memory. *arXiv preprint arXiv:2010.02500*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Zihao Xu, Xuan Tang, Yufei Shi, Jianfeng Zhang, Jian Yang, Mingsong Chen, and Xian Wei. 2023b. Continual learning via manifold expansion replay. *arXiv preprint arXiv:2310.08038*.

Da Yin, Xiao Liu, Fan Yin, Ming Zhong, Hritik Bansal, Jiawei Han, and Kai-Wei Chang. 2023. Dynosaur: A dynamic growth paradigm for instruction-tuning data curation. *arXiv preprint arXiv:2305.14327*.

Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. 2020. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2022. Continual sequence generation with adaptive compositional modules. *arXiv preprint arXiv:2203.10652*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

A Appendix

A.1 InsTag Process

Follow Lu et al. (2023), we use the prompt shown in Table 4 to employ GPT-4, providing fine-grained intention tags for given queries. To make the word format and granularity consistent, we filter the noise in raw tags as the following steps:

- **Rule Aggregation**: We replace all special characters with spaces and transform words into lowercase. Then, we apply lemmatization via NLTK (Bird et al., 2009) to unify tag formats.
- **Semantic Aggregation**: We obtain semantic embeddings of tags through PHRASE-BERT (Wang et al., 2021b), a BERT-based model designed for embedding phrases. Then, we cluster tags with semantic similarity via the DBSCAN algorithm (Hahsler et al., 2019). Here, we calculate the cosine similarity and set the cluster threshold to 0.1.

You are a tagging system that provides useful tags for instruction intentions to distinguish instructions for a helpful AI assistant. Below is an instruction: [begin] {instruction} [end] Please provide coarse-grained tags, such as "Spelling and Grammar Check" and "Cosplay", to identify main intentions of the above instruction. Your answer should be a list including titles of tags and a brief explanation of each tag. Your response has to strictly follow this JSON format: [{"tag": str, "explanation": str}]. Please respond in English.

Table 4: Prompt template for annotating intention tags of the given instruction.

A.2 Data Composition

SuperNI (Wang et al., 2022b) collects diverse NLP tasks with instructions using the Apache-2.0 license. The dataset curates task data in independent files, starting with a unique task ID (e.g., task001_quoref_question_generation.json). We integrate 765 English tasks from SuperNI into 16 categories, representing corresponding task IDs for each category in Table 5.

A.3 Forgetting Category Annotation

We invite 5 Chinese graduate students whose research field is related to NLP as annotation volunteers, manually labeling forgetting instances with Instruction-Related or Instruction-Unrelated. Additionally, we have procured approval from the annotator for utilizing the data in scientific research. We randomly sampled 3000 forgetting instances from 15 previous tasks for annotation (200 instances per task). To better understand the forgetting category, we demonstrate detailed cases and relevant explanations in Table 6.

Category	**Size**	**Task ID**
Classification	633k	20, 50, 65, 66, 69, 70, 109, 112, 114, 115, 116, 141, 142, 143, 145, 146, 147, 148, 149, 150, 155, 190, 199, 200, 201, 202, 226, 232, 233, 242, 274, 276, 280, 290, 291, 298, 340, 341, 342, 343, 345, 346, 347, 349, 350, 351, 364, 375, 379, 382, 391, 392, 393, 400, 428, 429, 430, 431, 457, 458, 459, 472, 495, 496, 514, 515, 516, 520, 521, 564, 566, 573, 575, 577, 583, 584, 590, 614, 617, 623, 625, 629, 630, 632, 633, 638, 640, 641, 642, 679, 681, 682, 738, 767, 827, 828, 848, 854, 855, 856, 890, 907, 908, 925, 935, 936, 937, 970, 1167, 1168, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1285, 1288, 1308, 1336, 1344, 1347, 1354, 1385, 1386, 1387, 1388, 1393, 1418, 1429, 1434, 1439, 1442, 1488, 1489, 1495, 1505, 1516, 1529, 1541, 1548, 1549, 1554, 1559, 1560, 1573, 1583, 1584, 1592, 1593, 1599, 1612, 1615, 1624, 1640, 1645, 1705, 1712
Generation	506k	1, 23, 25, 26, 59, 60, 67, 68, 71, 72, 74, 81, 82, 102, 103, 105, 166, 167, 182, 184, 191, 193, 219, 220, 246, 269, 270, 277, 278, 283, 287, 288, 294, 299, 300, 301, 303, 311, 381, 389, 405, 418, 453, 454, 455, 461, 470, 471, 489, 492, 500, 510, 547, 560, 563, 565, 568, 569, 574, 576, 581, 585, 592, 594, 599, 602, 610, 611, 619, 631, 639, 649, 672, 677, 739, 743, 760, 769, 821, 845, 847, 853, 857, 859, 860, 861, 871, 886, 897, 901, 917, 919, 927, 928, 957, 963, 964, 965, 967, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1161, 1217, 1325, 1326, 1339, 1342, 1356, 1358, 1359, 1360, 1379, 1381, 1383, 1398, 1400, 1407, 1409, 1508, 1509, 1519, 1540, 1566, 1567, 1580, 1582, 1585, 1586, 1590, 1594, 1598, 1600, 1602, 1603, 1609, 1631, 1657, 1659, 1660, 1665, 1703, 1704, 1711, 1713, 1714, 1728, 1729, 1730
Program Execution	433k	62, 63, 64, 78, 79, 91, 93, 94, 95, 96, 97, 98, 99, 100, 101, 113, 122, 123, 124, 125, 157, 158, 159, 160, 161, 162, 163, 205, 206, 207, 208, 243, 244, 245, 267, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 376, 377, 378, 488, 497, 499, 504, 505, 506, 507, 509, 523, 600, 605, 606, 622, 636, 637, 755, 756, 850, 851, 852, 1087, 1088, 1089, 1148, 1150, 1151, 1188, 1189, 1190, 1194, 1315, 1316, 1331, 1404, 1405, 1406, 1443, 1444, 1445, 1446, 1542, 1551
Open QA	302k	2, 24, 28, 61, 75, 80, 83, 84, 144, 151, 152, 153, 154, 170, 194, 247, 302, 309, 310, 339, 344, 380, 390, 460, 469, 490, 491, 580, 582, 591, 595, 596, 597, 598, 615, 740, 741, 742, 745, 750, 751, 752, 753, 754, 820, 835, 849, 858, 861, 862, 863, 864, 865, 866, 867, 868, 870, 887, 898, 918, 1135, 1286, 1293, 1296, 1327, 1382, 1399, 1412, 1419, 1420, 1421, 1422, 1423, 1424, 1431, 1520, 1564, 1565, 1581, 1601, 1608, 1656, 1661, 1678, 1726, 1727, 1731

Category	Size	Task ID
Sentiment Analysis	173k	195, 196, 284, 285, 293, 363, 397, 398, 399, 475, 476, 477, 478, 493, 494, 512, 517, 518, 586, 587, 588, 746, 761, 819, 823, 833, 843, 844, 875, 888, 889, 902, 903, 923, 929, 1292, 1310, 1311, 1312, 1313, 1338, 1361
Comprehension	149k	27, 33, 44, 46, 133, 168, 176, 192, 223, 227, 248, 249, 295, 304, 329, 330, 384, 401, 403, 462, 579, 593, 648, 673, 834, 846, 891, 892, 893, 899, 900, 966, 1289, 1294, 1328, 1366, 1369, 1390, 1391, 1664
Detection	147k	22, 88, 89, 108, 137, 209, 279, 286, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 333, 335, 337, 353, 354, 355, 356, 357, 358, 359, 386, 387, 513, 607, 608, 609, 904, 905, 1346, 1502, 1503, 1504, 1604, 1605, 1606, 1607, 1706, 1720, 1721, 1722, 1723, 1724, 1725
Rewriting	87k	34, 35, 45, 111, 121, 132, 177, 275, 402, 413, 442, 550, 626, 627, 628, 670, 671, 770, 933, 934, 955, 1195, 1340, 1345, 1364, 1368, 1401, 1557, 1562, 1622, 1669, 1670
Code	71k	76, 77, 107, 110, 126, 127, 128, 129, 130, 131, 210, 211, 212, 868, 869, 956
Closed QA	66k	47, 73, 104, 118, 119, 138, 139, 140, 156, 164, 165, 178, 228, 229, 268, 296, 297, 385, 664, 665, 666, 667, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 906, 909, 1378, 1380, 1389
Misc.	66k	43, 169, 183, 305, 306, 307, 308, 383, 567, 921, 922, 924, 1146, 1147, 1149, 1191, 1192, 1193, 1314, 1317, 1318, 1319, 1320, 1321, 1322, 1332, 1333, 1403, 1425, 1426, 1427, 1428, 1498, 1507, 1595, 1596
Extraction	59k	36, 39, 179, 180, 181, 281, 292, 388, 456, 578, 613, 620, 645, 683, 684, 874, 926, 1447, 1448, 1449, 1451, 1452, 1453, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1506, 1510, 1517, 1518, 1568
Summarization	40k	522, 589, 618, 668, 672, 1290, 1291, 1309, 1355, 1499, 1553, 1572
Dialogue	30k	362, 766, 879, 880, 1384, 1394, 1500, 1501, 1531, 1533, 1534
Mathematics	24k	85, 87, 90, 92
Text Quality Evaluation	20k	616, 674, 675, 1186, 1283, 1284, 1341

Table 5: We analyze the intention of instructions, reclassifying the task types into 16 categories. The task IDs contained in each category are reported.

Case	Explanation
In this task, you are given a context tweet, a question and corresponding answer of given question. Your task is to classify given passage into two categories: (1) "Yes" if the given context is useful in answering the question, and (2) "No" if the given context is not useful. Context: ... Ground Truth: No Instruction-Related Output: Yes Instruction-Unrelated Output: None	For close-domain instruction, we consider output within the specified range as instruction-related and vice versa as instruction-unrelated.
Craft one correct answer to the question given in input. In your answer, use as few words as possible from the given context. Use a response that is uncommon/non-stereotypical so that it is less predictable. Context: ..., Question: ... Ground Truth: He is my boyfriend. Instruction Related Output: We have a close relationship. Instruction Unrelated Output: 10	For open-domain instruction, we consider output that is relevant to the input as instruction-related, and vice versa as instruction-unrelated.
Given a command in a limited form of natural language, provide the correct sequence of actions that executes the command to thus navigate an agent in its environment. A command can be broken down into many different actions. ... There are only six actions: 'I_LOOK', 'I_WALK', 'I_RUN', 'I_JUMP', 'I_TURN_LEFT', and 'I_TURN_RIGHT'. jump opposite left and run opposite left. Ground Truth: I_TURN_LEFT I_TURN_LEFT I_JUMP I_TURN_LEFT I_TURN_LEFT I_RUN Instruction Related Output: I_JUMP I_TURN_LEFT Instruction Unrelated Output: turn left twice	For the instruction that imposes restrictions on the format (e.g., within 20 words / return in the form of / should be separated with a new line / ...), we consider output with the specified format as instruction-related, and vice versa as instruction-unrelated.
Given a factoid/trivia type question, generate the topic of the question. The topic is the entity the question talks about. For which paper was reporter Clark Kent/Superman employed? Ground Truth: superman, clark kent Instruction Related Output: paper Instruction Unrelated Output: planet	For Comprehension and Summarization tasks, we consider output containing the phrases extracted from the context as instruction-related, and vice versa as instruction-unrelated.
In this task, you will be given a list of integers. You should find the maximum absolute difference between 2 integers in the list. The absolute difference is the absolute value of one integer subtracted by another. The output should be a single integer which is the largest possible absolute distance. [-73, -93, -11, 79, -11, -17, -16, -52, -42, -28] Ground Truth: 172 Instruction Related Output: 170 Instruction Unrelated Output: [-11, -17, -16] or 999999	For tasks involving mathematical operations, we consider reasonable output in the same format as instruction-related, and vice versa as instruction-unrelated.

Table 6: We demonstrate representative cases of two categories for a better understanding.