Mitigating Catastrophic Forgetting in Large Language Models with Self-Synthesized Rehearsal

Jianheng Huang[1,3,5] **Leyang Cui**[2] **Ante Wang**[1,3,5] **Chengyi Yang**[1]
Xinting Liao[4] **Linfeng Song**[2] **Junfeng Yao**[5] **Jinsong Su**[1,3,5]*
[1]School of Informatics, Xiamen University [2]Tencent AI Lab
[3]Shanghai Artificial Intelligence Laboratory [4]Zhejiang University
[5]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan (Xiamen University), Ministry of Culture and Tourism, China
enatsu@stu.xmu.edu.cn jssu@xmu.edu.cn

Abstract

Large language models (LLMs) suffer from catastrophic forgetting during continual learning. Conventional rehearsal-based methods rely on previous training data to retain the model's ability, which may not be feasible in real-world applications. When conducting continual learning based on a publicly-released LLM checkpoint, the availability of the original training data may be non-existent. To address this challenge, we propose a framework called Self-Synthesized Rehearsal (SSR) that uses the LLM to generate synthetic instances for rehearsal. Concretely, we first employ the base LLM for in-context learning to generate synthetic instances. Subsequently, we utilize the latest LLM to refine the instance outputs based on the synthetic inputs, preserving its acquired ability. Finally, we select diverse high-quality synthetic instances for rehearsal in future stages. Experimental results demonstrate that SSR achieves superior or comparable performance compared to conventional rehearsal-based approaches while being more data-efficient. Besides, SSR effectively preserves the generalization capabilities of LLMs in general domains.

1 Introduction

Large language models (LLMs) have demonstrated remarkable performance across various natural language processing (NLP) tasks (Touvron et al., 2023b; OpenAI, 2023). In real-world applications, LLMs are often updated in a continual learning (CL) manner (de Masson d'Autume et al., 2019), where new instruction tuning data is incrementally introduced over time. However, a significant issue that limits the effectiveness of LLMs is catastrophic forgetting, which refers to the LLM's tendency to forget previously acquired knowledge when learning new instances (Kirkpatrick et al., 2017; Li et al., 2022; Luo et al., 2023).

*Corresponding author.

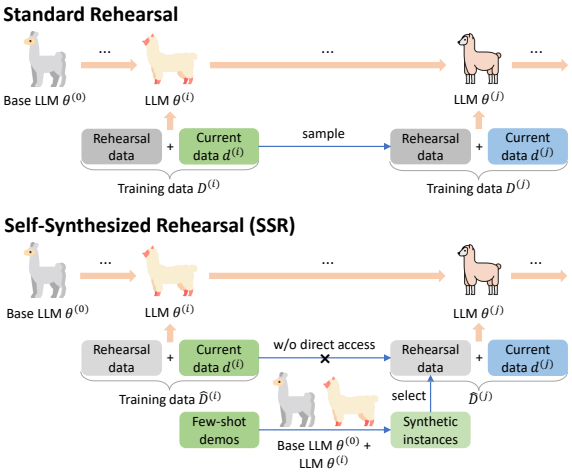

Figure 1: Comparison of standard rehearsal and our proposed Self-Synthesized Rehearsal (SSR).

To mitigate catastrophic forgetting, a line of work focuses on rehearsing previous training instances (de Masson d'Autume et al., 2019; Rolnick et al., 2019; Scialom et al., 2022). These rehearsal-based methods maintain the model's ability by training on real data from previous training stages. However, the real data may not always be desirable in practical applications. For instance, when conducting continual learning based on a publicly-released LLM checkpoint (e.g. Llama-2-chat), the availability of the original training data may be non-existent. This raises an interesting research question: *Can we maintain the LLM's ability during continual learning without using real data in previous training stages?*

We propose the **S**elf-**S**ynthesized **R**ehearsal (**SSR**) framework to mitigate catastrophic forgetting in continual learning. As shown in Figure 1, unlike standard rehearsal-based continual learning that samples training instances from previous stages as rehearsal data, SSR framework uses the LLM to generate synthetic instances for rehearsal. Specifically, we first use the base LLM to generate

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1416–1428

synthetic instances, conducting in-context learning (ICL) with few-shot demonstrations. These demonstrations can be collected from the previous data or human-constructed containing similar knowledge to the previous data. Then, the latest LLM is used to refine the outputs of synthetic instances to retain the latest LLM's ability. Finally, we select diverse high-quality synthetic instances for rehearsal in the future stages.

Extensive experiments on the task sequences derived from the SuperNI dataset (Wang et al., 2022) demonstrate that SSR has superior or comparable performance compared to the conventional rehearsal-based approaches, with higher data utilization efficiency. Besides, experiments on AlpacaEval and MMLU (Hendrycks et al., 2021) show that SSR can also effectively preserve the generalization capabilities of LLMs in general domains. We release our code and data at https://github.com/DeepLearnXMU/SSR.

2 Related Work

Learning a sequence of datasets continually while preserving past knowledge and skills is a crucial aspect of achieving human-level intelligence. Existing approaches to continual learning can be broadly categorized into three main categories: (i) regularization-based, (ii) architecture-based, and (iii) rehearsal-based methods. Regularization techniques (Kirkpatrick et al., 2017; Cha et al., 2021; Huang et al., 2021; Zhang et al., 2022) control the extent of parameter updates during the learning process, preventing interference with previously learned tasks. Nonetheless, these methods typically rely on hyperparameters that need to be carefully tuned for optimal performance. Architecture-based approaches (Xu and Zhu, 2018; Huang et al., 2019; Razdaibiedina et al., 2023) often take a different approach by learning separate sets of parameters dedicated to individual tasks. This enables the model to specialize and adapt its parameters for each task, avoiding interference between tasks and preserving task-specific knowledge. However, these approaches will introduce additional training parameters, which may not be very flexible and feasible for various LLMs.

Therefore, we focus on rehearsal-based methods (de Masson d'Autume et al., 2019; Rolnick et al., 2019), which are also called replay-based methods. These methods typically involve the storage of a subset of data from previous tasks. These stored data are used for future rehearsal through techniques such as experience replay (Rolnick et al., 2019) and representation consolidation (Bhat et al., 2022). Prior rehearsal-based approaches for language models mainly focus on using a little bit of precedent data (Scialom et al., 2022; Mok et al., 2023; Zhang et al., 2023b). However, these approaches often ignore discussion on real-world applications where previous data may be limited or unavailable. Although data-free knowledge distillation methods (Yin et al., 2020; Smith et al., 2021) introduce auxiliary generative models for data construction, they are primarily designed for classification tasks, which may not be effective in LLMs, where a wide range of NLP tasks are involved. Additionally, similar to introducing teacher models (Miao et al., 2023; Cheng et al., 2023; Huang et al., 2024), it can be challenging and time-consuming to train additional generative models. Self-distillation methods (Zhang et al., 2023a) may be useful, but catastrophic forgetting of the latest LLMs and the knowledge discrepancy among LLMs from distinct stages are still inevitable challenges.

In this work, we propose a rehearsal-based continual learning framework in which LLMs can be trained on self-synthesized data to retain the knowledge of the previous stages, with several demonstrations used during data construction. Unlike other approaches, our framework does not depend on additional generative models for data construction or require previous real data for rehearsal. This offers advantages in terms of data efficiency and application flexibility.

3 Rehearsal-Based Continual Learning

In continual learning, the LLM is sequentially updated for T stages, with each stage t having its corresponding instruction data $d^{(t)}$. To mitigate catastrophic forgetting, in each stage t, rehearsal-based methods (Scialom et al., 2022; Mok et al., 2023) sample some training instances of previous stages to expand the training data in the current stage. Formally, the augmented training data $D^{(t)}$ can be formulated as follows:

$$D^{(t)} = d^{(t)} \bigcup \sum_{i=1}^{t-1} (rd^{(i)}), \quad (1)$$

where r represents the rehearsal ratio determining the percentage of sampled training instances. Finally, we use $D^{(t)}$ to fine-tune the LLM $\theta^{(t-1)}$, obtaining the updated LLM $\theta^{(t)}$. Particularly, in

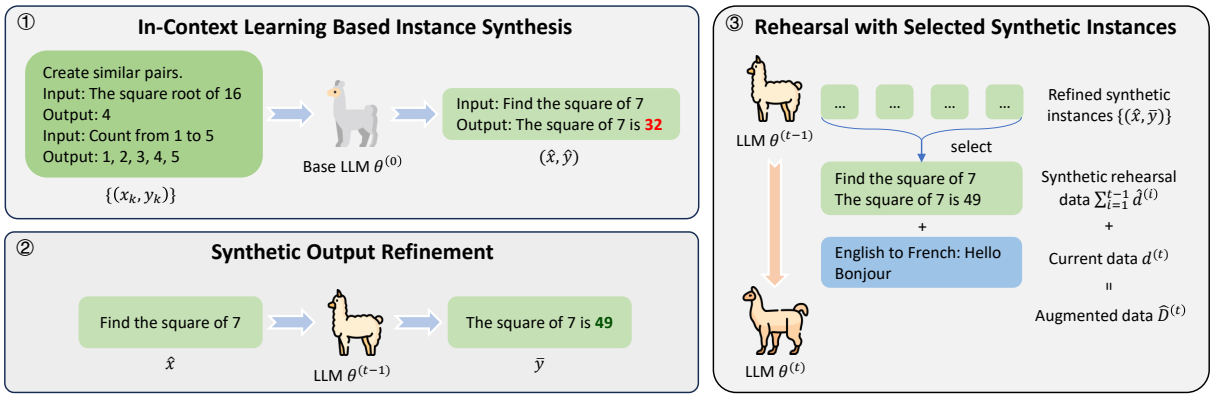

Figure 2: Our SSR framework. To mitigate catastrophic forgetting with limited or no rehearsal data, we first adopt the base LLM $\theta^{(0)}$ with in-context learning to generate synthetic instances $\{(\hat{x}, \hat{y})\}$. We then utilize the latest LLM $\theta^{(t-1)}$ to generate the refined output $\bar{y}$ based on $\hat{x}$. Finally, diverse high-quality synthetic instances are selected for rehearsal in the future stages.

the first stage, we fine-tune the base LLM $\theta^{(0)}$ on $D^{(1)} = d^{(1)}$. By doing so, the catastrophic forgetting problem of LLM can be effectively alleviated, which has been verified in previous studies (Scialom et al., 2022; Mok et al., 2023; Zhang et al., 2023b).

4 Our Framework

In this section, we detail the proposed **S**elf-**S**ynthesized **R**ehearsal (**SSR**) framework, which involves three main steps: 1) in-context learning based instance synthesis, 2) synthetic output refinement, and 3) rehearsal with selected synthetic instances, as illustrated in Figure 2.

In-Context Learning Based Instance Synthesis Rehearsal-based methods utilize the training instances to cache the knowledge acquired by the LLM from previous stages. Nevertheless, in real-world scenarios where a publicly-released LLM checkpoint is used, the availability of original training data may be limited. To address this limitation, we try to generate rehearsal training instances synthetically. To ask the LLM to follow abstract instructions, we leverage the in-context learning (ICL) capability of LLMs for instance synthesis.

Formally, during each training stage t, we first acquire K demonstrations $\{(x_k, y_k)\}_{k=1}^{K}$. To retain previously acquired knowledge, these demonstrations can be collected from the previous instruction data $d^{(t-1)}$ or manually constructed containing similar knowledge to $d^{(t-1)}$. We concatenate all demonstrations and utilize the base LLM to generate the synthetic instance $(\hat{x}, \hat{y}) = \text{LLM}(\text{concat}_{k=1}^{K}(x_k, y_k); \theta^{(0)})$. By reordering the demonstrations and sampling multiple times, we can easily obtain different synthetic instances. It should be noted that we use the base LLM $\theta^{(0)}$ rather than the latest LLM $\theta^{(t-1)}$ to conduct ICL. This is because the ICL ability of LLMs tends to exhibit a significant degradation after supervised fine-tuning (SFT) on specific tasks, as analyzed in (Wang et al., 2023).

Synthetic Output Refinement Through the above process, we obtain a series of synthetic instances, some of which, however, may be of low quality with unreliable outputs. To address this issue, we use the latest LLM $\theta^{(t-1)}$ to refine the output of each synthetic instance: $\bar{y} = \text{LLM}(\hat{x}; \theta^{(t-1)})$. By doing so, we can ensure that each refined synthetic instance $(\hat{x}, \bar{y})$ retains the knowledge acquired by the latest LLM.

Rehearsal with Selected Synthetic Instances Finally, we select the refined synthetic instances for rehearsal. During this process, to ensure the diversity and quality of selected synthetic instances, we first adopt a clustering algorithm (e.g. K-means) to group $\{(\hat{x}, \bar{y})\}$ into C clusters. Then we calculate the distance between each synthetic instance and its corresponding cluster centroid, and finally select a certain amount of synthetic instances near cluster centroids as the rehearsal data.

Formally, we use $\hat{d}^{(t-1)}$ to represent the set of selected synthetic instances. Thus, the augmented training data in stage t can be formulated as

$$\hat{D}^{(t)} = d^{(t)} \bigcup \sum_{i=1}^{t-1} \hat{d}^{(i)}, \tag{2}$$

where $\hat{d}^{(i)}$ denotes the selected synthetic data similar to the previous training data $d^{(i)}$. Note that $\hat{d}^{(1)}, \hat{d}^{(2)}, ..., \hat{d}^{(t-2)}$ have been generated in previous stages, thus we will not regenerate them in stage t. Finally, we use $\hat{D}^{(t)}$ to fine-tune $\theta^{(t-1)}$, updating the LLM as $\theta^{(t)}$. In this way, the LLM can preserve previously learned knowledge even without real data from previous stages.

5 Experiments

5.1 Setup

Datasets We conduct several groups of experiments on the SuperNI dataset (Wang et al., 2022), a vast and comprehensive benchmark dataset for instruction tuning. First, to simulate a typical continual learning process, we choose a subset of 10 tasks from SuperNI, encompassing various categories and domains. Each task is trained in a separate stage for empirical studies. For each task, we randomly sample 2,000 instances for training and 500 for evaluation. Please refer to Appendix A for the details of these tasks. To simplify the empirical study, we adopt default continual learning orders on {5, 10} SuperNI tasks: QA → QG → SA → Sum. → Trans. (→ DSG → Expl. → Para. → PE → POS).

Base LLMs Our main experiments involve three base LLMs: Llama-2-7b (Touvron et al., 2023b), Llama-2-7b-chat (Touvron et al., 2023b), Alpaca-7b (Taori et al., 2023).

Baselines We compare SSR with the following baselines:

- **Multi-task Learning (MTL)**. This is the most commonly used baseline, where all tasks are trained simultaneously.
- **Non-rehearsal**. It is a naive baseline that the LLM is fine-tuned with only the instruction data $d^{(t)}$ in each stage t.
- **RandSel(r)** (Scialom et al., 2022). We randomly sample $r = \{1, 10\}\%$ of the original instruction data for each previous task. Note that as mentioned in (Scialom et al., 2022), the abilities of language models can be effectively preserved with $r = 1\%$.
- **KMeansSel(r)**. Unlike the above approach, we first employ K-means clustering to group real instances into 20 clusters and then select $r = \{1, 10\}\%$ of instances with the highest similarities to the cluster centroids. Here we adopt SimCSE (Gao et al., 2021) to obtain instance representations before clustering.

Evaluation Metrics Due to the diversity and the open-ended sequence generation characteristic of SuperNI tasks, we adopt the ROUGE-L metric (Lin, 2004) to evaluate the performance of LLM on each task. This metric shows a good alignment with human evaluation, as demonstrated by Wang et al. (2022). Besides, we follow Lopez-Paz and Ranzato (2017) to consider the following metrics based on ROUGE-L. Here $a_j^{(i)}$ denotes the ROUGE-L performance on the task j in training stage i.

- **Average ROUGE-L (AR)**. It is used to quantify the final average performance of LLM across all T tasks in stage T, which is defined as follows:

$$\mathbf{AR} = \frac{1}{T}\sum_{i=1}^{T} a_i^{(T)}. \tag{3}$$

- **Forward Transfer (FWT)**. It evaluates the LLM's generalization ability on unseen tasks, measuring the average zero-shot performance $a_i^{(i-1)}$ on the next task i in each stage $i-1$:

$$\mathbf{FWT} = \frac{1}{T-1}\sum_{i=2}^{T} a_i^{(i-1)}. \tag{4}$$

- **Backward Transfer (BWT)**. It is a metric used to evaluate the impact of learning subsequent tasks on a previous task. For each task i except for the final one, it compares the final performance $a_i^{(T)}$ to the online performance $a_i^{(i)}$ in stage i:

$$\mathbf{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1} (a_i^{(T)} - a_i^{(i)}). \tag{5}$$

  A negative BWT indicates that the LLM has forgotten some previously acquired knowledge.

Implementation Details During training, we utilize LoRA (Hu et al., 2021) with query and value projection matrices in the self-attention module to train LLMs, setting the LoRA rank to 8 and the dropout rate of 0.1. We employ the Adam optimizer with an initial learning rate of 2e-4. The

Model	Order 1		Order 2		Order 3		Avg.	
	AR	BWT	AR	BWT	AR	BWT	AR	BWT
Llama-2-7b								
MTL	53.05	-	53.05	-	53.05	-	53.05	-
Non-rehearsal	17.67	-44.09	15.25	-47.09	24.16	-35.99	19.03	-42.39
RandSel(1%)	51.16	-2.34	49.21	-4.36	48.63	-5.37	49.67	-4.02
KMeansSel(1%)	50.20	-3.12	49.75	-4.11	50.12	-3.61	50.02	-3.61
RandSel(10%)	50.81	-2.32	50.04	-3.31	50.11	-3.42	50.32	-3.02
KMeansSel(10%)	50.44	-3.03	50.61	-2.32	49.89	-3.53	50.31	-2.96
SSR	**52.61**	**-0.23**	**51.70**	**-1.22**	**52.16**	**-0.93**	**52.16**	**-0.79**
Llama-2-7b-chat								
MTL	52.81	-	52.81	-	52.81	-	52.81	-
Non-rehearsal	23.87	-36.31	30.96	-27.41	42.06	-13.50	32.30	-25.74
RandSel(1%)	51.28	-1.96	49.77	-3.70	49.41	-4.29	50.15	-3.32
KMeansSel(1%)	51.82	-1.25	50.71	-2.44	50.22	-3.42	50.92	-2.37
RandSel(10%)	50.59	-2.57	50.72	-2.45	50.24	-2.87	50.52	-2.63
KMeansSel(10%)	50.81	-2.55	51.39	-1.42	50.22	-2.84	50.81	-2.27
SSR	**52.52**	**-0.23**	**52.49**	**-0.35**	**52.73**	**0.05**	**52.58**	**-0.18**
Alpaca-7b								
MTL	52.79	-	52.79	-	52.79	-	52.79	-
Non-rehearsal	17.24	-44.21	45.40	-9.03	35.60	-21.45	32.75	-24.90
RandSel(1%)	51.61	-0.93	49.08	-4.68	49.01	-4.85	49.90	-3.49
KMeansSel(1%)	51.37	-1.53	50.53	-2.68	50.15	-3.17	50.68	-2.46
RandSel(10%)	50.91	-1.82	50.88	-2.11	49.98	-3.59	50.59	-2.51
KMeansSel(10%)	50.78	-2.05	51.20	-1.76	49.76	-3.48	50.58	-2.43
SSR	**52.52**	**-0.14**	**51.74**	**-1.21**	**52.33**	**-0.51**	**52.20**	**-0.62**

Table 1: Final results on 5 SuperNI tasks under different continual learning (CL) orders. For more details, please refer to Appendix B.

global batch size is 32 for all our experiments. Besides, we set the maximum length of the input to 1,024 and the counterpart of the output to 512. Following Luo et al. (2023), we train each LLM for 3 epochs and use the final checkpoint for evaluation.

To conduct ICL, we utilize 1% of the training data from SuperNI tasks as demonstrations, considering $K = 2$ demonstrations and sampling multiple times to obtain diverse synthetic instances. When clustering instances, we use K-means clustering with $C = 20$ clusters for synthetic instances of SuperNI tasks, which is similar to KMeansSel(r).

5.2 Experiments on 5 SuperNI Tasks

Table 1 presents the experimental results on 5 SuperNI tasks. Overall, regardless of the continual learning order and the base LLM, SSR consistently outperforms all rehearsal-based baselines, exhibiting an improvement of approximately 2 scores in both the AR and BWT metrics. This result shows the superiority of SSR in mitigating catastrophic forgetting. Particularly, SSR closely approaches MTL which sets the upper bound of the AR performance. Besides, compared to rehearsal-based baselines, RandSel(r) and KMeansSel(r), SSR is more data-efficient with only 1% real data utilization for ICL and only synthetic data of previous stages for rehearsal.

After further analysis, we draw the following conclusions:

Non-rehearsal vs. rehearsal The non-rehearsal baseline shows the poorest, indicating severe catastrophic forgetting. Besides, it exhibits the highest metric variance, signifying its lack of robustness in different CL orders. In contrast, SSR and rehearsal-based baselines maintain better and more consistent performance regardless of the CL order.

Effect of r The appropriate rehearsal ratio r varies depending on the continual learning orders. A higher r is beneficial in certain cases, as observed in CL orders 2 and 3. However, this is not always the case. In CL order 1, regardless of the instance sampling strategy employed, the rehearsal-based baselines with $r = 1\%$ consistently outperform their $r = 10\%$ counterparts, respectively.

RandSel(r) vs. KMeansSel(r) When comparing RandSel(r) and KMeansSel(r), we can observe that K-means clustering-based selection of previous

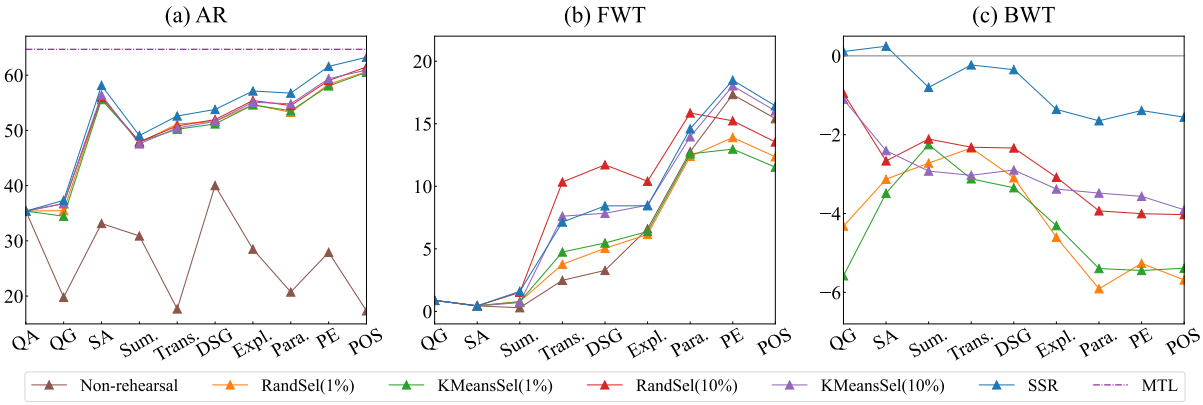

Figure 3: AR, FWT, and BWT during continual learning for Llama-2-7b on 10 SuperNI tasks.

Model	AR	FWT	BWT
Llama-2-7b			
MTL	64.69	-	-
Non-rehearsal	17.33	15.41	-53.64
RandSel(1%)	60.64	12.35	-5.69
KMeansSel(1%)	60.51	11.53	-5.39
RandSel(10%)	61.49	13.54	-4.03
KMeansSel(10%)	60.93	16.03	-3.90
SSR	**63.23**	**16.43**	**-1.56**

Table 2: Final results for Llama-2-7b on 10 SuperNI tasks.

data for rehearsal may slightly enhance the model performance when using only $r = 1\%$, demonstrating the importance of data representativeness. However, when r is set to 10%, significant differences in model performance may not be observed for LLMs such as Llama-2-7b and Alpaca-7b.

5.3 Experiments on 10 SuperNI Tasks

To further investigate the effectiveness of SSR in longer continual learning sequences, we evaluate SSR and all baselines on 10 SuperNI tasks. Table 2 shows that SSR surpasses all rehearsal-based baselines across all metrics. Moreover, as illustrated in Figure 3, SSR consistently achieves better performance in terms of AR and BWT compared to rehearsal-based baselines throughout the entire continual learning process. Although SSR with Llama-2-7b as the base LLM falls behind RandSel(10%) in terms of FWT in the early stage, it gradually strengthens its performance as the number of training stages increases, eventually surpassing RandSel(10%). Please refer to Appendix C for more details.

5.4 Experiments on the Generalization Capability Preservation of Alpaca-7b

To further analyze the preservation of LLM's generalization ability in a broader domain beyond SuperNI tasks, we utilize Alpaca-7b to conduct continual learning on 5 SuperNI tasks and then investigate whether SSR can preserve the abilities of Alpaca-7b gained from the Alpaca-52k dataset[1]. Here, Llama-7b (Touvron et al., 2023a) serves as the base LLM $\theta^{(0)}$, and Alpaca-7b is considered as the updated LLM $\theta^{(1)}$ after fine-tuning on Alpaca-52k. Therefore, we also generate synthetic instances similar to Alpaca-52k for SSR and use Alpaca-52k for rehearsal-based baselines.

We evaluate the LLM from three perspectives: 1) General instruction-following ability. We use AlpacaEval 2.0[2] as an automatic evaluator. Concretely, we measure the LLM's performance in terms of the **win rate**, comparing the LLM's generations with those generated by `gpt-4-turbo`. To minimize financial costs, we utilize ChatGPT as the evaluation annotator. 2) General language understanding ability. We leverage the MMLU (Hendrycks et al., 2021) benchmark, where **accuracy (Acc.)** is used as the evaluation metric. 3) Task-specific ability. We evaluate the AR performance of the LLM on 5 SuperNI tasks. Please refer to Appendix D for more details.

From Table 3, we observe that SSR not only achieves the best on the 5 newly learned tasks but also maintains comparable or even superior performance on AlpacaEval and MMLU. These findings suggest that SSR effectively preserves the

[1] https://huggingface.co/datasets/tatsu-lab/alpaca

[2] https://github.com/tatsu-lab/alpaca_eval

Model	AlpacaEval	MMLU	SuperNI
	win rate	Acc.	AR
Alpaca-7b	21.08	41.0	22.80
Non-rehearsal	8.82	34.1	17.24
RandSel(1%)	19.45	36.9	51.80
RandSel(10%)	**20.06**	36.4	50.47
SSR	19.68	**37.1**	**52.11**

Table 3: Final results on Alpaca-52k + 5 SuperNI tasks.

Model	AR	BWT
Llama-2-7b		
SSR	52.61	**-0.23**
Llama-2-7b⇒Llama-7b	**52.71**	-0.36
Llama-2-7b⇒Alpaca-7b	52.07	-0.78
train demos⇒new demos	52.54	-0.44
w/ input-only demos	52.61	-0.34

Table 4: Effect of in-context learning for Llama-2-7b on 5 SuperNI tasks.

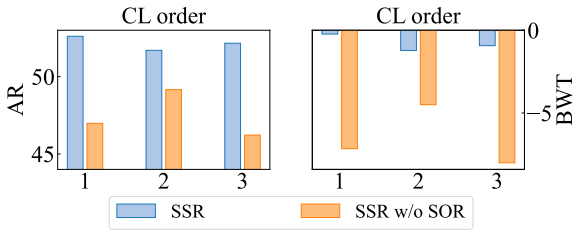

Figure 4: Effect of synthetic output refinement (SOR) for Llama-2-7b on 5 SuperNI tasks under different continual learning orders.

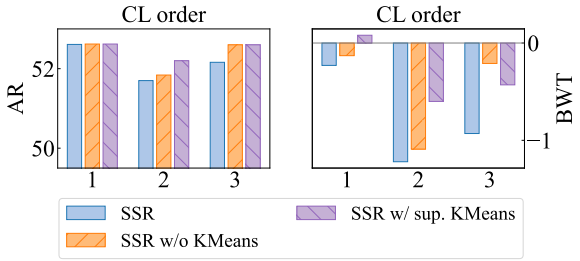

Figure 5: Effect of K-means clustering for Llama-2-7b on 5 SuperNI tasks under different continual learning orders.

generalization ability of Alpaca-7b throughout the continual learning process, even in the absence of Alpaca-52k as rehearsal data. This highlights the great potential of SSR in general domains.

5.5 Analysis

Effect of in-context learning To investigate the effect of in-context learning on SSR, we conduct experiments for Llama-2-7b on 5 SuperNI tasks, introducing the following variants: (a) **Llama-2-7b⇒Llama-7b**. This variant validates the scenario where we acquire a public-released fine-tuned LLM checkpoint, but the original base LLM is unavailable. Thus we employ a different LLM Llama-7b to conduct ICL. (b) **Llama-2-7b⇒Alpaca-7b**. Similar to the above one, but using Alpaca-7b. (c) **train demos⇒new demos**. In this variant, we utilize demonstrations that are not included in the previous training data but belong to the same SuperNI task, simulating manually constructed demonstrations to conduct ICL. (d) **w/ input-only demos**. This variant utilizes only the instance inputs in previous stages as demonstrations, simulating the scenario where real instances lack output annotations.

Table 4 illustrates that SSR can perform well even without the original base LLM or demonstrations from previous training data to conduct ICL. This provides convenience for replacing some ICL components in practical application scenarios. Comparing SSR and its variants (a) to (b), we notice a slight decrease in performance when conducting ICL using Alpaca-7b. This highlights the limitation of this fine-tuned LLM in terms of ICL capability. Besides, ICL with input-only demonstrations also yields comparable performance, indicating that output annotations are also not essential for ICL, further verifying the robustness of SSR.

Effect of synthetic output refinement In Section 4, we claim that synthetic output refinement provides more reliable synthetic outputs from the latest LLM. To verify the effectiveness, we conduct an experiment where SSR is implemented without synthetic output refinement.

As illustrated in Figure 4, this results in lower AR values and significant BWT inferiority, highlighting the negative impact of data noise originating from the base LLM. In contrast, by incorporating the synthetic input with the refined output, SSR can maintain the predictive distribution of the latest LLM during rehearsal, preserving the acquired knowledge.

Effect of K-means clustering on synthetic instance selection In terms of application flexibility, we utilize an unsupervised K-means clustering algorithm, fitting and predicting solely with synthetic instances. To explore the effect of K-means clustering, we compare SSR with the following variants: (a) **SSR w/o KMeans**: Random selection of synthetic instances. (b) **SSR w/ sup. KMeans**:

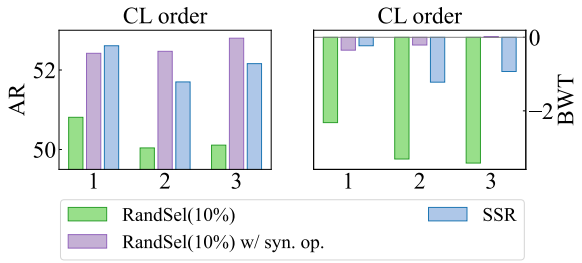

Figure 6: Effect of synthetic inputs and outputs for Llama-2-7b on 5 SuperNI tasks under different continual learning orders. Note that RandSel(10%) w/ syn. op. (synthetic outputs) in CL order 3 has the best BWT value of 0.02.

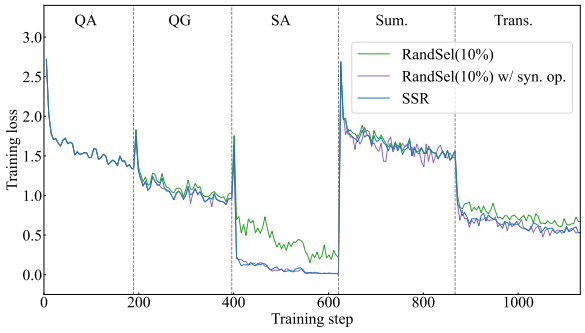

Figure 7: Effect of synthetic inputs and outputs on loss curve for Llama-2-7b on 5 SuperNI tasks.

K-means clustering-based synthetic instance selection, with the supervision of real instances to fit the K-means clustering and then predict on synthetic instances.

Figure 5 shows that the supervised K-means clustering method leads to a slight improvement in AR and reduces forgetting with larger BWT. Thus, incorporating real instances during clustering may allow for a more representative selection. Nonetheless, clustering is not essential in the absence of supervision, because SSR with random selection for synthetic instances can outperform SSR, sometimes even surpass SSR with supervised K-means. This indicates a certain level of robustness of SSR.

Real instances vs. synthetic instances Our main experiments demonstrate surprising results that rehearsal with synthetic instances may surpass those with real instances. For the comparison of real and synthetic instances, we consider the following variants: (a) **RandSel (10%)**: Real inputs and outputs for rehearsal. (b) **RandSel (10%) w/ syn. op.**: Real inputs and synthetic outputs for rehearsal. Concretely, we regenerate the outputs of randomly sampled previous instances by the latest LLM, with similar operations to SSR.

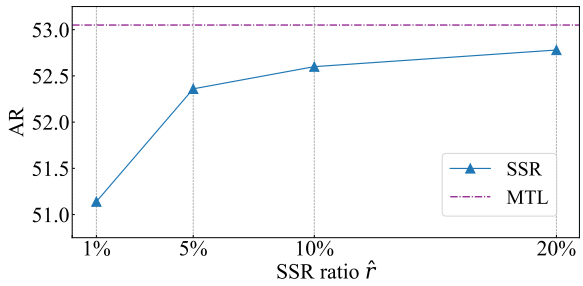

Figure 8: Effect of the SSR ratio $\hat{r}$ for Llama-2-7b on 5 SuperNI tasks.

Figure 6 demonstrates that RandSel (10%) with real inputs and synthetic outputs for rehearsal, outperforms RandSel (10%) utilizing only real instances. Meanwhile, SSR, which leverages both synthetic inputs and outputs for rehearsal, achieves intermediate performance, sometimes even surpassing the other two. This indicates that real instances are not always essential and appropriate for the continual learning of LLMs. As depicted in Figure 7, real instances often lead to a slower descent in loss. Therefore, they may not be conducive to optimization due to the distribution gap between distinct datasets. Conversely, synthetic instances, with lower model perplexity, embody the LLM's real-time acquired knowledge, which aids in smoothing the data distribution and discovering better local optima for the LLM.

Effect of synthetic instance quantity Here, we define $\hat{r} = |\hat{d}^{(i)}|/|d^{(i)}|$ as the **SSR ratio**, which represents the proportion of selected synthetic data compared to the original training data size. By default, we retain synthetic instances at an SSR ratio of $\hat{r} = 10\%$. However, as depicted in Figure 8, increasing $\hat{r}$ can lead to further improvements in the final AR, highlighting the potential of SSR. Moreover, it is important to note that the training cost and memory limit should also be taken into consideration when determining an appropriate $\hat{r}$.

SSR vs. regularization-based and architecture-based methods In this paper, we focus on generating synthetic rehearsal data for instruction tuning. Prior work (Zhang et al., 2023b) has demonstrated that rehearsal-based approaches are generally superior to regularization-based and architecture-based ones for instruction tuning of language models. Table 5 presents experimental results using two classical regularization-based baselines (L2 and

Model	AR	FWT	BWT
Llama-2-7b			
MTL	53.05	-	-
Non-rehearsal	17.67	2.49	-44.09
regularization-based			
L2	29.22	7.08	-28.45
EWC	15.94	2.43	-46.30
SSR	**52.61**	**7.14**	**-0.23**

Table 5: Comparison between SSR and regularization-based methods for Llama-2-7b on 5 SuperNI tasks under CL order 1.

EWC) for Llama-2-7b under CL order 1, with SSR still demonstrating its superiority. Besides, these lightweight strategies can be easily combined with SSR, potentially leading to further improvements in model performance. Moreover, architecture-based approaches heavily rely on additional task-specific parameters, which may not be practical in real-world applications where the inference time of LLMs is a crucial consideration.

6 Conclusion

In this work, we propose Self-Synthesized Rehearsal (SSR), a continual learning framework for mitigating catastrophic forgetting in LLMs, to effectively preserve knowledge without relying on real data during rehearsal. Through extensive experiments, SSR demonstrates its data efficiency and superior performance to conventional rehearsal-based approaches. Besides, it preserves LLM's generalization capability both in specific and general domains, with flexibility and robustness in real-world scenarios. Overall, SSR presents a promising solution for continual learning of LLMs in real-world settings, with implications for maintaining the acquired abilities of LLMs.

Limitations

Although SSR demonstrates superior performance in terms of AR and BWT, it may not always achieve the best FWT score, as shown in Figure 3(b). However, as discussed in Subsection 5.4, SSR effectively preserves the generalization capabilities of LLMs in general domains, highlighting its practical value. For the final FWT results on the 5 SuperNI tasks, please refer to Table 7 in Appendix B. Additionally, synthetic instances generated by LLMs may potentially contain unsafe content due to data bias during training.

Acknowledgements

The project was supported by National Key R&D Program of China (No. 2022ZD0160501), National Natural Science Foundation of China (No. 62276219), and the Public Technology Service Platform Project of Xiamen (No.3502Z20231043). We also thank the reviewers for their insightful comments.

References

Prashant Bhat, Bahram Zonooz, and Elahe Arani. 2022. Task agnostic representation consolidation: a self-supervised based continual learning approach.

Sungmin Cha, Hsiang Hsu, Taebaek Hwang, Flavio P. Calmon, and Taesup Moon. 2021. Cpr: Classifier-projection regularization for continual learning.

Xuxin Cheng, Zhihong Zhu, Wanshi Xu, Yaowei Li, Hongxiang Li, and Yuexian Zou. 2023. Accelerating multiple intent detection and slot filling via targeted knowledge distillation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8900–8910, Singapore. Association for Computational Linguistics.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

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.

Jianheng Huang, Ante Wang, Linfeng Gao, Linfeng Song, and Jinsong Su. 2024. Response enhanced semi-supervised dialogue query generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18307–18315.

Shenyang Huang, Vincent François-Lavet, and Guillaume Rabusseau. 2019. Neural architecture search for class-incremental learning.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual learning for text classification with information disentanglement based regularization.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Dingcheng Li, Zheng Chen, Eunah Cho, Jie Hao, Xiaohu Liu, Fan Xing, Chenlei Guo, and Yang Liu. 2022. Overcoming catastrophic forgetting during domain adaptation of seq2seq language generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5441–5454, Seattle, United States. Association for Computational Linguistics.

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

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

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.

Zhongjian Miao, Wen Zhang, Jinsong Su, Xiang Li, Jian Luan, Yidong Chen, Bin Wang, and Min Zhang. 2023. Exploring all-in-one knowledge distillation framework for neural machine translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2929–2940, Singapore. 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)*, pages 12573–12589, Toronto, Canada. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report.

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

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. 2019. Experience replay for continual learning.

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, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. 2021. Always be dreaming: A new approach for data-free class-incremental learning.

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, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.

Yihan Wang, Si Si, Daliang Li, Michal Lukasik, Felix Yu, Cho-Jui Hsieh, Inderjit S Dhillon, and Sanjiv Kumar. 2023. Two-stage llm fine-tuning with less specialization and more generalization.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma,

Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ju Xu and Zhanxing Zhu. 2018. Reinforced continual learning.

Hongxu Yin, Pavlo Molchanov, Zhizhong Li, Jose M. Alvarez, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion.

Han Zhang, Sheng Zhang, Yang Xiang, Bin Liang, Jinsong Su, Zhongjian Miao, Hui Wang, and Ruifeng Xu. 2022. CLLE: A benchmark for continual language learning evaluation in multilingual machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 428–443, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Liang Zhang, Jinsong Su, Zijun Min, Zhongjian Miao, Qingguo Hu, Biao Fu, Xiaodong Shi, and Yidong Chen. 2023a. Exploring self-distillation based relational reasoning training for document-level relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13967–13975.

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

A Details of the Selected 10 SuperNI Tasks

Table 9 lists all of the selected 10 SuperNI tasks for our main experiments. To simplify the description, we utilize abbreviations to represent these tasks in this paper. The SuperNI dataset can be found at https://github.com/allenai/natural-instructions.

B More Details of Experiments on 5 SuperNI Tasks

Table 6 lists all of the continual learning orders on 5 SuperNI tasks conducted in our experiments.

Order	Task Sequence
1	QA → QG → SA → Sum. → Trans.
2	Trans. → SA → QA → Sum. → QG
3	Sum. → QG → Trans. → QA → SA

Table 6: Continual learning orders on 5 SuperNI tasks.

Table 7 shows the final FWT results on 5 SuperNI tasks. Our SSR framework surpasses $r = 1\%$ rehearsal-based baselines but falls behind $r = 10\%$ counterparts. However, as illustrated in Figure 3, the FWT performance of SSR will finally surpass the $r = 10\%$ rehearsal-based baselines as the number of training stages increases.

Model	Order 1	Order 2	Order 3	Avg.
Llama-2-7b				
Non-rehearsal	2.49	1.99	3.42	2.63
RandSel(1%)	3.77	2.84	4.21	3.61
KMeansSel(1%)	4.75	2.46	5.25	4.15
RandSel(10%)	**10.34**	**3.20**	**7.54**	**7.03**
KMeansSel(10%)	7.61	2.67	6.50	5.59
SSR	7.14	2.62	6.39	5.38
Llama-2-7b-chat				
Non-rehearsal	16.93	13.56	22.13	17.54
RandSel(1%)	19.06	17.14	18.67	18.29
KMeansSel(1%)	23.19	16.46	18.36	19.34
RandSel(10%)	26.29	**17.52**	19.53	21.11
KMeansSel(10%)	**27.05**	16.65	18.59	20.76
SSR	26.63	16.41	**26.10**	**23.05**
Alpaca-7b				
Non-rehearsal	22.32	13.13	21.88	19.11
RandSel(1%)	25.26	14.62	25.46	21.78
KMeansSel(1%)	25.07	15.10	25.50	21.89
RandSel(10%)	25.97	16.76	**30.34**	**24.36**
KMeansSel(10%)	**26.41**	17.14	27.01	23.52
SSR	25.45	**17.45**	27.69	23.53

Table 7: Final FWT results on 5 SuperNI tasks under different continual learning orders.

C More Details of Experiments on 10 SuperNI Tasks

Figure 9 depicts the heatmaps of the ROUGE-L performance for Llama-2-7b across 10 SuperNI tasks. A visual inspection reveals that the non-rehearsal baseline rapidly forgets previously learned tasks when subsequent tasks are learned. In contrast, after learning a task in its respective stage, SSR demonstrates minimal color change in future stages, indicating the least amount of forgetting.

Table 8 highlights that SSR retains its superiority in terms of AR and BWT for Llama-2-7b-chat and Alpaca-7b on 10 SuperNI tasks. However, its FWT performance is comparable or inferior to that of rehearsal-based baselines. It is worth noting that Alpaca-7b tends to exhibit poorer performance regardless of the CL approaches employed.

Model	AR	FWT	BWT
Llama-2-7b-chat			
MTL	62.68	-	-
Non-rehearsal	47.05	22.24	-20.73
RandSel(1%)	60.70	25.77	-4.17
KMeansSel(1%)	61.53	26.62	-4.24
RandSel(10%)	58.46	**27.99**	-3.34
KMeansSel(10%)	59.27	27.52	-3.12
SSR	**63.34**	27.70	**-1.27**
Alpaca-7b			
MTL	63.60	-	-
Non-rehearsal	36.37	25.44	-31.50
RandSel(1%)	59.99	**28.95**	-4.58
KMeansSel(1%)	58.70	27.53	-3.53
RandSel(10%)	58.93	28.11	-3.16
KMeansSel(10%)	58.72	27.93	-3.09
SSR	**60.24**	27.55	**-2.35**

Table 8: Final results for Llama-2-7b-chat and Alpaca-7b on 10 SuperNI tasks.

D More Implementation Details of Experiments on the Generalization Capability Preservation of Alpaca-7b

The continual learning order is as follows: (Alpaca-52k →) QA → QG → SA → Sum. → Trans. To conduct ICL, we utilize 1% / 0.1% of the training data from SuperNI tasks and Alpaca-52k as demonstrations, respectively. When clustering instances, we use KMeans with $C = 20$ clusters for synthetic instances of SuperNI tasks and $C = 520$ for those of Alpaca-52k. We retain synthetic instances with the SSR ratio $\hat{r} = 10\%$ / 1% for SuperNI tasks and Alpaca-52k, respectively.

Abbr.	Category	Name	NLU task
QA	Question Answering	task024_cosmosqa_answer_generation	-
QG	Question Generation	task074_squad1.1_question_generation	-
SA	Sentiment Analysis	task1312_amazonreview_polarity_classification	+
Sum.	Summarization	task511_reddit_tifu_long_text_summarization	-
Trans.	Translation	task1219_ted_translation_en_es	-
DSG	Dialogue Sentence Generation	task574_air_dialogue_sentence_generation	-
Expl.	Explanation	task192_hotpotqa_sentence_generation	-
Para.	Paraphrasing	task177_para-nmt_paraphrasing	-
POS	POS Tagging	task346_hybridqa_classification	+
PE	Program Execution	task064_all_elements_except_first_i	-

Table 9: Details of the selected 10 SuperNI tasks.

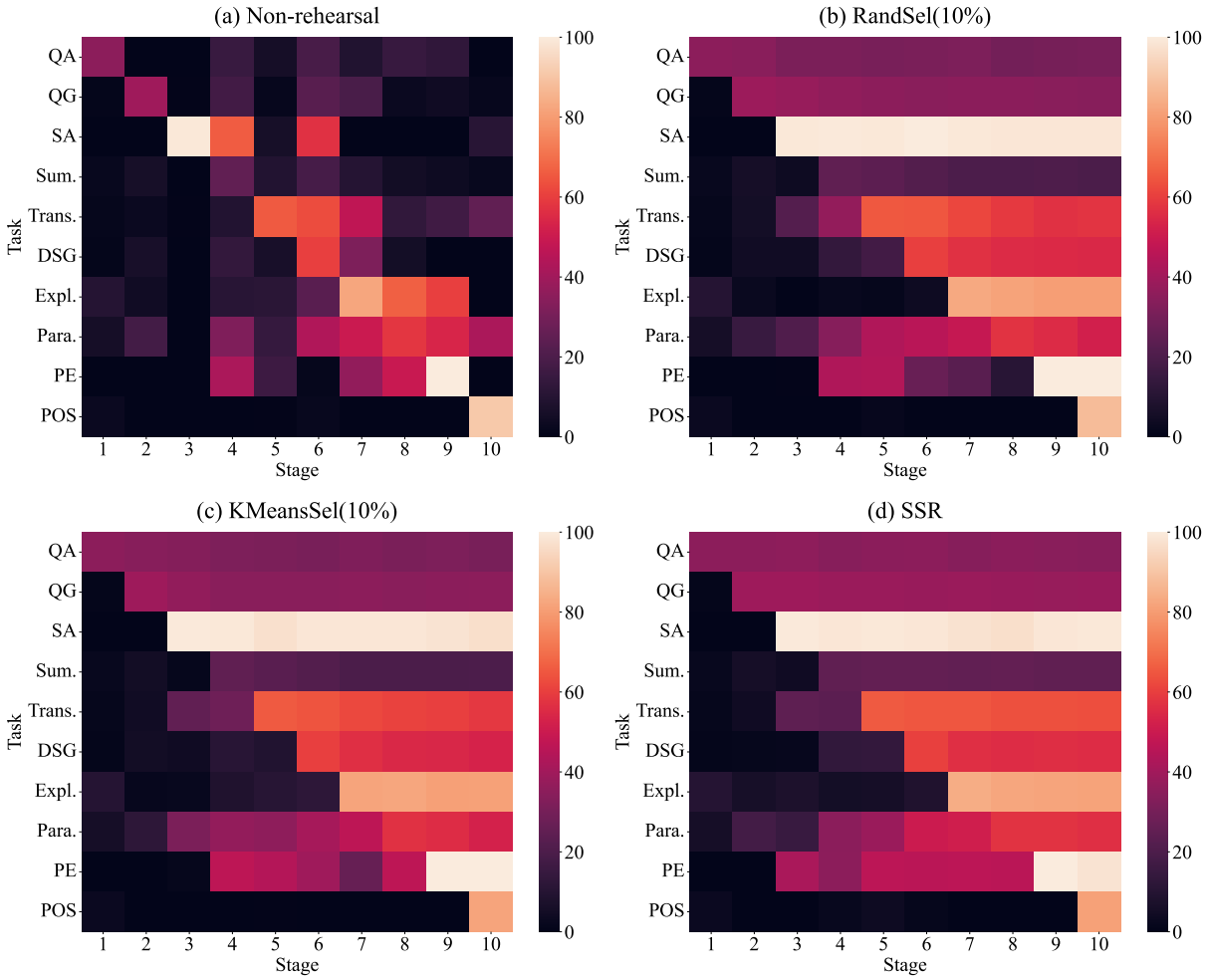

Figure 9: ROUGE-L heatmaps for Llama-2-7b on 10 SuperNI tasks.