KSOD: Knowledge Supplement for LLMs On Demand

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities in various tasks, yet still produce errors in domain-specific tasks. To further improve their performance, we propose KSOD (Knowledge Supplement for LLMs On Demand), a novel framework that empowers LLMs to improve their capabilities with knowledge-based supervised fine-tuning (SFT). KSOD analyzes the causes of errors from the perspective of knowledge deficiency by identifying potential missing knowledge in LLM that may lead to the errors. Subsequently, KSOD tunes a knowledge module on knowledge dataset and verifies whether the LLM lacks the identified knowledge based on it. If the knowledge is verified, KSOD supplements the LLM with the identified knowledge using the knowledge module. Tuning LLMs on specific knowledge instead of specific task decouples task and knowledge and our experiments on two domain-specific benchmarks and four general benchmarks empirically demonstrate that KSOD enhances the performance of LLMs on tasks requiring the supplemented knowledge while preserving their performance on other tasks. Our findings shed light on the potential of improving the capabilities of LLMs with knowledge-based SFT.

1 Introduction

Large Language Models (LLMs) have demonstrated excellent performance across a wide range of tasks, showing their remarkable general-purpose capabilities(Brown et al., 2020; Ouyang et al., 2022; OpenAI, 2023; Touvron et al., 2023; Chowdhery et al., 2023; Jiang et al., 2024a). However, LLMs still hallucinate and produce factually incorrect, irrelevant, or incomplete content, leading to the errors in their outputs.

Existing methods(An et al., 2023; Ying et al., 2024; Tong et al., 2024) for improving the outputs of LLMs commonly depend on supervised fine-tuning (SFT). These methods necessitate extensive training datasets generated with stronger LLMs (e.g. GPT-4) or costly human annotations, which may not always be accessible. Furthermore, the application of SFT on datasets collected from some tasks can potentially compromise the capabilities of LLMs on other tasks. Consequently, many studies explore the potential of self-correction, where the LLM itself is prompted or guided to repair the errors in its own output by refining the output(Pan et al., 2023). Despite the self-correction method improves the fluency and understandability, it leads to false positive optimization and reduces diversity in text generation because of the universal existence of self-bias(Xu et al., 2024). Moreover, LLMs cannot solve the errors caused by the lack of knowledge or ability based on the feedback generated by

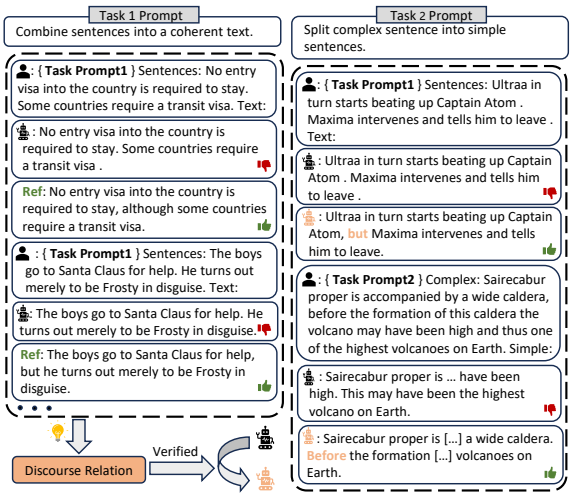

Figure 1: On the left side of the figure, samples from Task 1 is presented in the form of (input, output with errors, correct reference). Based on these samples, KSOD identifies the missing knowledge as discourse relations. After verify that LLM lacks this knowledge, it is supplemented into the LLM. As shown on the right side of the figure, after supplementation, the model generates correct outputs not only for Task 1 but also for another task (Task 2) that requires the discourse relation knowledge.

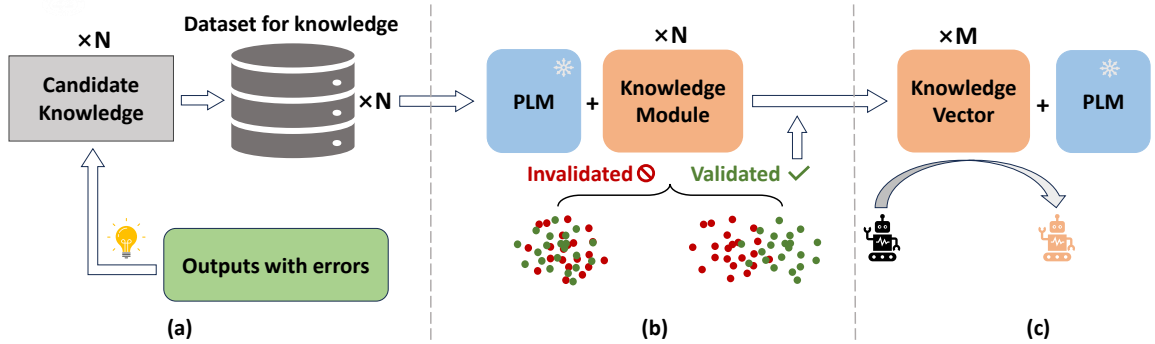

Figure 2: Our KSOD framework consists of three stages: (a) Knowledge Identification; (b) Knowledge Verification; (c) Knowledge Supplement.

themselves.

To address these challenges, we introduce the KSOD (Knowledge Supplement for LLMs On Demand) framework to correct the errors by supplementing LLMs with required knowledge on demand. KSOD framework diverges from conventional SFT methods by decoupling knowledge and task to correct the errors of specific task from the perspective of knowledge rather than the task itself. As illustrated in Figure 1, supplementing LLMs with missing knowledge, identified as the cause of errors in Task 1, not only mitigates these errors within in task 1, but also yields consistent improvements in tasks that also rely on the same knowledge, such as Task 2. Furthermore, our empirical evaluation on general tasks that explicitly require this knowledge demonstrates that KSOD introduces little to no degradation to LLMs' performance in these tasks.

In specific, KSOD first identifies the knowledge missing in LLMs that may lead to the errors and collects dataset containing the required knowledge from existing datasets. Subsequently, we train a knowledge module on the identified dataset. To ensure that the learned knowledge is genuinely missing in LLMs rather than already possessed, which could introduce noise, KSOD performs a knowledge verification step. Notably, only knowledge modules that pass this verification phase proceed to the knowledge supplement stage, where they are injected into the LLM. Finally, the verified knowledge module is integrated into LLMs to correct errors arising from missing knowledge. Figure 2 provides an overview of our framework.

To validate the performance of KSOD, we conduct comprehensive experiments with open source LLMs on both two error-prone tasks where the errors occurred and four general tasks. Our results show that supplementing knowledge with KSOD brings a notable reduction in errors, leading to notable performance improvements. Furthermore, after supplementing the missing knowledge through KSOD, the performance of LLMs on four general tasks and the remaining error-prone task remains unchanged or slightly decreases, with some cases even showing improvement. These empirical findings demonstrate that KSOD effectively correct errors by supplementing the desired knowledge missing in LLMs while preserving the LLMs' performance on other tasks. This finding highlights the potential of enhancing LLMs performance by supplementing knowledge in a task-agnostic way.

In summary, our contributions are as follows:

- Given specific knowledge, KSOD provides a general method to verify whether LLM lacks the knowledge.
- We propose the KSOD framework, which corrects the errors from LLMs by supplementing required knowledge on demand with knowledge-based SFT, while preserving LLM's performance on other tasks.
- With extensive experiments, we validate the effectiveness of our proposed framework and reveal the potential of enhancing LLMs' performance by supplementing knowledge in a task-agnostic way.

2 KSOD Framework

2.1 Proposed Research Questions

To correcting the errors in outputs of LLMs from the perspective of knowledge, we try to research knowledge-based SFT to supplement the knowledge desired to generate correct outputs but miss-

ing in LLMs. Specifically, we aim to address the following research questions (RQs):

- **RQ1**: How to verify whether LLMs lack specific knowledge?
- **RQ2**: What are the effects of knowledge-based SFT on tasks that require such knowledge and those that do not?

2.2 Preliminaries: Knowledge-based SFT

We begin with preliminaries, formally introducing knowledge-based SFT and the scope of two RQs.

In the context of LLMs, the knowledge can be implicit knowledge encoded within the model's parameters. Given a LLM represented by parameter θ_0, the objective of knowledge-based SFT is update the parameters of LLM as $\theta' = \theta_0 + \Delta\theta$, where $\Delta\theta$ is a low-rank update relative to θ_0 and encodes a specific type of knowledge that is missing in the original LLM.

In this study, we focus on knowledge of categories that can be formalized as a classification task. Therefore, the scope of two RQs are restricted to knowledge learned as a classification task in this work.

2.3 KSOD Overview

In this section, we present our knowledge-based SFT framework, KSOD, to correct the errors in outputs of LLMs from the perspective of knowledge. As shown in Figure 2, KSOD consists of three stages: Knowledge Identification (§3.1), Knowledge Verification (§3.2) and Knowledge Supplement (§4).

The knowledge identification stage aims to find dataset containing the missing knowledge, whose deficiency may cause the errors in outputs of LLMs. During knowledge verification stage, KSOD finetunes the LLMs on these datasets using LoRA(Hu et al., 2021) as a knowledge module and verifies whether the LLMs lack specific knowledge based on the embeddings distribution of the knowledge module. Clearly, not all knowledge identified in the knowledge identification stage is missing in the LLM. Only the knowledge that passes verification will be passed to the knowledge supplement stage. During this stage, the verified knowledge module will be supplemented into the LLMs to enhance the performance of LLMs on the tasks that requiring the knowledge.

3 Verifying whether the LLM lacks specific knowledge

This section focus on the verification of identified knowledge and aims to solve RQ1. The verification process is outlined in Algorithm 1.

Algorithm 1 Process of knowledge identification and knowledge verification stages.

Input: Language model π_{θ_0}, a set of samples $\mathcal{F} = \{s_1, ..., s_K\}$, expert model E, classification layer π_{θ_c}, knowledge module $\pi_{\Delta\theta}$, threshold ϵ

Output: A set of verified knowledge module $\mathcal{M}$

1: $\mathcal{K} \leftarrow E(\mathcal{F})$
2: $\mathcal{M} \leftarrow \{\}$
3: **for** knowledge k **in** $\mathcal{K}$ **do**
4: Dataset $d_k \leftarrow$ Collect(k)
5: Update π_{θ_c} with π_{θ_0} frozen on d_k
6: Update $\pi_{\Delta\theta}$ with π_{θ_0} and π_{θ_c} frozen on d_k
7: $E_k \leftarrow Embed(\pi_{\Delta\theta}, d_k)$
8: $S_k \leftarrow$ S_C(E_k, d_k)
9: **if** $S_k \geq \epsilon$ **then**
10: $\mathcal{M} \leftarrow \{\pi_{\Delta\theta}\} \cup \mathcal{M}$
11: **end if**
12: **end for**

This algorithm corresponds to stages (a) and (b) in Figure 2. Specifically, Collect(·) refers to gathering the dataset corresponding to the knowledge (line 4 of Algorithm 1), Embed(·) denotes obtaining the distribution of embeddings from knowledge module on d_k (line 7 of Algorithm 1) in and S_C(·) represents the computation of the Silhouette Coefficient (line 8 of Algorithm 1).

3.1 Knowledge Identification

To learn from errors from the perspective of knowledge, KSOD identifies the knowledge whose deficiency in LLMs may cause the errors.

Formally, given a set $\mathcal{F}$ consisting of samples in the format of (input, erroneous output, correct reference), the aim of knowledge identification stage is to construct the set that contains the knowledge whose absence in LLMs may cause the errors in $\mathcal{F}$. This set can be denoted as $\mathcal{K} = \{k_1, k_2...\}$.

To construct $\mathcal{K}$, we manually select N samples with similar errors from $\mathcal{F}$. Leveraging the powerful knowledge storage and language processing capabalities, we utilize strong LLMs like GPT-4 to identify the knowledge whose absence in LLM may cause the errors in $\mathcal{F}$. The simplified parompt

template is as follows:

Prompt for Knowledge Identification

{TASK DEFINITION} Please analyze the errors that arise in output of {TASK NAME} task in the given samples.
{
sample i:
Input: {input text}
Target: {correct reference}
Output: {output with errors}
} i = 1,2,...,N
Firstly, provide a step-by-step analysis for the common characteristics of the errors from all samples.
Next, identify the potential knowledge lacking in LLM that may have led to these errors.

After obtaining k_i, the process of finding the datasets containing knowledge K_i from available existing NLP datasets becomes more straightforward and simple. For sample, huggingface[1] offers more than 250K datasets where we can search and download dataset containing the identified knowledge.

3.2 Knowledge Verification

After fine-tuning a LLM on the target task using LoRA, its performance in maintaining capabilities on other tasks surpasses full fine-tuning, and even common regularization methods (Biderman et al., 2024). Therefore, LoRA is a suitable method for LLMs to learn the knowledge they lack without affecting the initial capabilities of LLMs on other tasks. Based on the hypothesis that the change in weights during fine-tuning is low rank, the vanilla LoRA is mathematically represented as:

$$\begin{aligned} W' &= W_0 + \Delta W \\ &= W_0 + \frac{\alpha}{r} BA \\ &= W_0 + \eta BA \end{aligned} \tag{1}$$

where $W', W_0 \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times r}$, and $A \in \mathbb{R}^{r\times n}$, with $r \ll \min(m, n)$. W_0 is the pre-trained weight matrix and η is a hyperparameter serving as a scalar weight, where both of them are frozen during fine-tuning. Only A and B contain trainable parameters. As stated in Section 2.2, knowledge-based SFT in this work is performed in the form of a classification task. Therefore, we additionally introduce a classifier layer, which is also trainable.

However, when utilizing the LoRA to learn knowledge, ideally, the scalar of LoRA should be large when current knowledge is deficient and employ the parameters in A, B for learning. Conversely, the scalar should be small when the knowledge is sufficient, thus avoid introducing noise. To enable LoRA adeptly adjust the scalar value based on different knowledge, we follows Liu et al. (2024) and set η to be trainable, which has already been proven to be an important design for improving LoRA's performance.

Furthermore, to further reduce the impact of LoRA on the general capabilities of LLMs, we divided the training of LoRA on knowledge dataset into two stages: in the first stage, only the classification layer is tuned with LLM frozen; in the second stage, only LoRA is tuned with both LLM and the tuned classification layer frozen.

Based on the LoRA with trainable scalar, we call the LoRA variant, which is fine-tuned on a specific knowledge dataset, a knowledge module. We hypothesize that the embedding distribution of knowledge module will exhibit clustering characteristics consistent with knowledge categorization if and only if the LLMs lack the knowledge. To prove this hypothesis, we examine the embedding distribution of knowledge modules tuned on different knowledge datasets in Section 3.4.

In summary, the Knowledge Verification stage can be divided into two steps as shown in Algorithm 1: initially, we fine-tune LoRA based on the dataset procured during the knowledge identification stage to obtain the knowledge module; subsequently, we evaluate the effectiveness of the knowledge module by verifying whether the embedding distribution of the knowledge module exhibits clustering characteristics consistent with knowledge categories in dataset. If these clustering characteristics do not become apparent, it is inferred that the LLM does not lack this particular type of knowledge. As a result, we verify the next kind of knowledge identified during knowledge identification stage. If the clustering characteristics is apparent, the knowledge is verified and will be supplemented to LLMs during knowledge supplement stage.

3.3 Experiment for knowledge identification

Tasks and Datasets. We collect samples with errors in Sentence Fusion task, whose target is join-

[1] https://huggingface.co/datasets

Examples with errors	**Output**: He [...] managed eventually to pay his creditors in full. Some inheritance was left for his descendants.	**Target**: He [...] managed eventually to pay his creditors in full so that some inheritance was left for his descendants.
	Output: They finished third among the league's eight teams, with Gore as their starting center fielder. O'Rourke had moved to left due to the departure of Slattery.	**Target**: They finished third among the league's eight teams, with Gore as their starting center fielder, while O'Rourke had moved to left due the departure of Slattery.
Expert Justification	**Knowledge Type**: Understanding of Logical and Causal Relationships. *(GPT-4o)*	The model fails to detect and explicitly represent causal or logical links [...] *(GPT-4o)*
	Knowledge Type: Discourse Structure Understanding. *(DeepSeek-R1)*	LLMs struggle to recognize implicit discourse relationships (e.g., cause-effect, contrast) between sentences and select appropriate connectives (as a result, while, so that). [...] *(DeepSeek-R1)*

Table 1: One example of knowledge identification with two strong LLMs. This table presents 2 out of the 4 examples; the full prompt can be found in Appendix A.

ing several independent sentences into a single coherent text. Specifically, we used DiscoFuse(Geva et al., 2019), a large-scale sentence fusion dataset, to collect the initial outputs with errors from LLMs.

Experimental Setup. In terms of selecting LLMs for the experiments, we need to select two kinds of LLMs: the LLMs generate outputs with errors and the strong LLMs for identifying the missing knowledge that may cause the errors in outputs. For the former, we employ 2 different open-source LLMs for experiments as follows:

- **LLaMA-3.1-8B** (Dubey et al., 2024) (denoted as LLaMA-3) is a dense LLM with massive pre-training on extremely large corpora, which is developed by Meta.
- **Qwen2.5-7B** (Yang et al., 2024) (denoted as Qwen2) is a powerful multilingual LLM developed by Alibaba Cloud.

For the latter, we employ GPT-4o (Hurst et al., 2024) and DeepSeek-R1 (Guo et al., 2025) to identify knowledge.

We manually selected 4 samples with similar errors and used the prompt presented in Section 3.1 to analyze the potentially missing knowledge.

Identified Knowledge. As shown in Table 1, all of the samples with errors have difficulty in utilizing proper conjunctions. Consequently, both GPT-4o and DeepSeek-R1 concluded that discourse relations constitute the most probable knowledge whose absence causes the errors in samples. Therefore, we select discourse relation as the first knowledge to be verified.

Dataset	#Class	#Train	#Dev	#Test	Deficiency
DiscoWiki	4	20,000	2,500	2,500	Yes
SST-2	2	20,000	2,500	2,500	No
EXPECT	15	15,187	2,413	2,416	Yes
AEGIS2.0	2	21,446	1,087	1,567	No

Table 2: Data Statistics of four datasets for knowledge verification. For datasets with a large size (DiscoWiki, SST-2), we sample the same number of instances from the dataset itself as the experimental dataset.

3.4 Experiment for knowledge verification

Tasks and Datasets. To get the corresponding dataset for discourse relation classification, we use an automatically rule-based method (Ma et al., 2019) to label the WikiSplit++ (Tsukagoshi et al., 2024) dataset, obtaining a dataset with discourse relation labels, which we called DiscoWiki. DiscoWiki contains four types of discourse relation following PDTB3.0(Webber et al., 2019) and we have selected an equal number of samples for each type of discourse relations. To validate the effectiveness of the knowledge verification method, we introduce three additional types of knowledge and determine whether the LLM lacks them based on existing research:

- **Discourse relation:** DiscoWiki is used for 4-class classification of discourse relations. According to the evaluation by Chan et al.

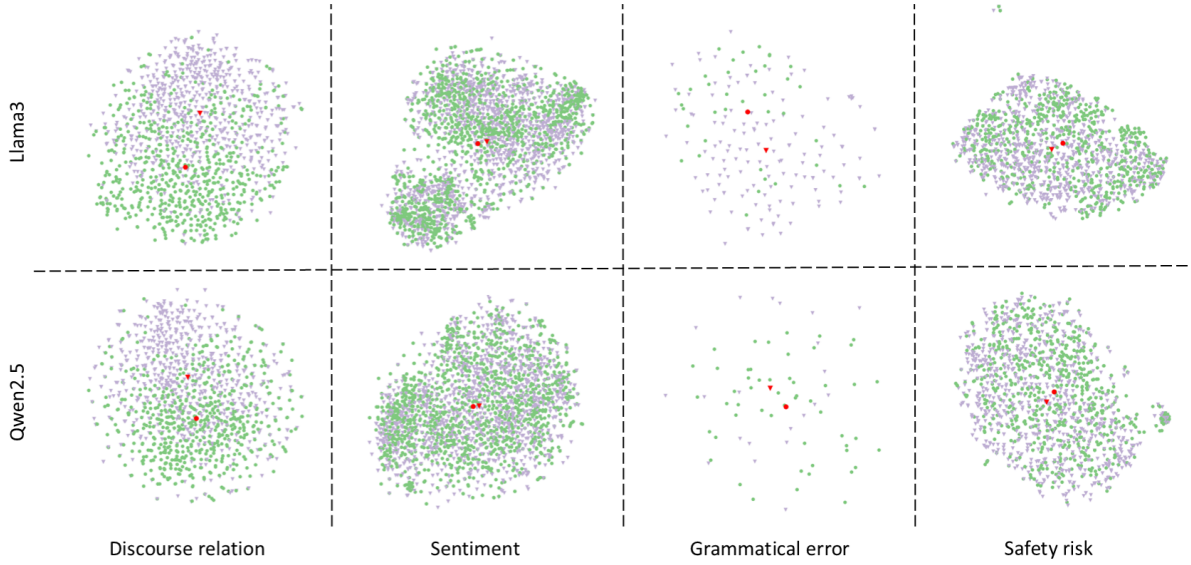

Figure 3: T-SNE (Van der Maaten and Hinton, 2008) visualization of the embedding distribution and each color represents a category within categorical knowledge based on dataset labels. The embedding is the last token embedding from B matrix of LoRA on test set.

(2024), LLMs still struggle to classify implicit discourse relations. Therefore, we conclude that this knowledge is missing in the LLM.

- **Sentiment:** Stanford Sentiment Treebank binary (SST-2) is used for 2-class sentiment classification. The binary sentiment classification is typically well mastered by LLMs and LLMs do not lack it.

- **Grammatical error:** EXPECT (Fei et al., 2023) is used for 15-class classification of grammatical error types. The error types can be used to enhance the performance of LLMs on Grammatical Error Correction (GEC) task(Fei et al., 2023), where LLMs often underperform task-specific models in this task(Davis et al., 2024). The LLMs lack this knowledge so that supplementing LLMs with it allows for performance improvement on the GEC task.

- **Safety risk:** AEGIS2.0 (Ghosh et al.) is used for 2-class classification of safety risks. Zheng et al. (2024) have found that LLMs are naturally capable of distinguishing harmful and harmless queries without safety prompts. Hence, the safety risks knowledge is not missing in LLMs.

The detailed statistics of four datasets have listed in Table 2.

Dataset	Llama3	Qwen2
DiscoWiki	0.0423	0.0233
SST-2	0.0098	0.0027
EXCEPT	0.0478	0.0663
AEGIS2.0	0.0125	0.0108

Table 3: SC for embeddings clustering.

Experimental Setup. To learn knowledge with the selected dataset, we first tune the final classification linear layer itself with backbone parameters frozen. After tuning the classification linear layer, we learning knowledge with LoRA. Specifically, we set the init scalar η for LoRA to 0, and both A and B are initialized with Gaussian initialization. More details of training can be found in Appendix B.

3.5 RQ1: How to verify whether LLMs lack specific knowledge?

To ensure the comparability of categorical knowledge with different numbers of types, we select two types of data with the most distinct embedding distribution as representatives of this categorical knowledge. To assessing the clustering characteristics, we visualize the embeddings of LoRA trained on different datasets and models in Figure 3 and calculate the Silhouette Coefficient (SC) score with knowledge category label to evaluate the cluster characteristics in Table 3.

Model	General Language Tasks					Sentence Fuse	GEC
	Drop	Squad	ARC	HellaSwag	Avg.	DiscoFuse	CoNLL14
LLaMA3-8B	47.25	71.81	79.44	79.52	69.51	43.89	37.14
LLaMA3-8B-DR	47.16	71.74	79.35	79.66	69.48	45.13	36.49
LLaMA3-8B-GE	46.88	71.62	79.61	79.70	69.45	44.65	37.74
LLaMA3-8B-DR+GE	46.13	71.24	79.35	79.80	69.13	45.31	36.84
Qwen2-7B	37.93	56.98	89.85	77.84	65.65	43.41	31.03
Qwen2-7B-DR	40.27	57.92	90.02	77.91	66.53	43.81	30.97
Qwen2-7B-GE	38.95	57.63	89.76	78.08	66.11	44.04	31.34
Qwen2-7B-DR+GE	41.31	58.61	89.85	78.21	67.00	44.07	30.58

Table 4: Comparison of evaluation results of knowledge supplement among several benchmarks. DR refers to discourse relation and GE refers to grammatical error.

Based the visualization of embeddings distribution of knowledge for four datasets, it it obvious that the embeddings distribution of discourse relation and grammatical error exhibits characteristics corresponding to knowledge categories. From the SC calculation results in the Table 3, we can also reach the same conclusion that the embeddings distribution of knowledge module learned on DiscoWiki and EXCEPT exhibits clustering characteristics matching knowledge categories, while the knowledge modules learned on a dataset like SST-2 that contains knowledge already mastered by LLMs do not exhibit such characteristics. The experimental results empirically validate the effectiveness of our knowledge verification method.

4 Effect of knowledge-based SFT

4.1 Knowledge supplement

A *task vector* is built by the difference between the weights of a pre-trained model and the weights of the same model after fine-tuning on a task which specifies the direction and stride of fine-tuning. More importantly, simple arithmetic on *task vector* can be used to control the behavior of the resulting model(Ilharco et al., 2022). Inspired by *task vector*, we propose *knowledge vector*, which can be built simply use the weights of knowledge module that has been verified during knowledge verification stage. In this way, LLMs can learn specific knowledge through the addition of the corresponding *knowledge vector*.

Compared with *task vector*, *knowledge vector* decouples task and knowledge. The *task vector* learns knowledge of a specific task, simultaneously, influences the original task instructions following ability to utilize corresponding task-specific knowledge, leading to the performance declines on other tasks (Kotha et al., 2023; Jiang et al., 2024b; Sun and Gao, 2024). Conversely, the *knowledge vector* learns knowledge in a task-agnostic manner, exerting less impact on the general capabilities of LLMs.

4.2 Experiment for knowledge supplement

Tasks and Datasets. The evaluation is performed on four key general benchmarks using the LLM-Box (Tang et al., 2024), a comprehensive library for implementing LLMs, including a unified training pipeline and comprehensive model evaluation. We evaluate the LLM with knowledge vector with four benchmarks for general language tasks. Furthermore, we incorporate benchmarks requiring the verified knowledge, including an augmented version of DiscoFuse with multi-reference(Ben-David et al., 2020) for discourse relation knowledge and CoNLL14 (Ng et al., 2014) for grammatical error knowledge.

4.3 RQ2: What are the effects of knowledge-based SFT on tasks that require such knowledge and those that do not?

We compare the performance of pretrained LLM, LLM with single knowledge vector and LLM with combination of different knowledge vectors. The results are presented in Table 4.

In terms of the results using discourse relation knowledge vector, both LLaMA and Qwen show significant improvements on the Sentence Fusion task, while exhibits a slight performance decline on the GEC task. On general tasks, LLaMA's performance remains largely unaffected, whereas Qwen

demonstrates a notable improvement.

Regarding the results using grammatical error knowledge vector, LLaMA and Qwen achieve improvements on both the Sentence Fusion and GEC tasks. However, LLaMA exhibits a slight performance decline on general tasks, whereas Qwen shows an improvement.

The results using the combined knowledge vectors of discourse relation and grammatical error exhibit a more complex pattern. Both LLaMA and Qwen achieve significant improvements on the Sentence Fusion task, reaching optimal performance, as both types of knowledge vectors contribute positively to this task. However, for the GEC task, the combined knowledge vectors lead to a slight performance decline. On general tasks, LLaMA experiences a slight decrease in performance, whereas Qwen demonstrates a significant improvement.

In summary, whether used individually or in combination, knowledge vectors can enhance the performance of LLMs on tasks that require such knowledge while not leading to a significant decline in other tasks.

The results highlight that LLM with single knowledge vector effectively balances general capabilities and knowledge-related capabilities.

5 Related Work

Learning from mistakes Humans can learn from mistakes to improve their capabilities and correct mistakes. Inspired by this, researchers have explored leveraging mistakes to enhance the performance of LLMs (Tong et al., 2024; An et al., 2023; Li et al., 2024; Wang et al., 2024). The LEMA (LEarning from MistAkes) method proposed by An et al. (2023) fine-tuning LLMs on pairs consisting of errors and their respective corrections generated by GPT-4. Similarly, Tong et al. (2024) fine-tuning LLMs on CoTErrorSet, a benchmark constructed by having the LLM prompted to correct its own errors based on the correct reference and the incorrect response generated by itself.

However, rather than fine-tuning on datasets constructed based on error responds across various tasks, we analyze the causes of errors from the perspective of knowledge deficiencies and correct errors by fine-tuning the model to learn the required knowledge from a curated knowledge dataset.

Self-correction Self-correction typically involves three stages: a LLM generates initial outputs, a feedback model generates feedback given the input and initial output and a refinement model generates a refined output considering the input, initial output and feedback. In the context of self-correction, LLMs refine their own responds based on the feedback from either themselves (Madaan et al., 2024) or external tools or knowledge (Shinn et al., 2024; Gou et al., 2023). Self-correction focus on the utilization of feedback to refine the outputs of LLM while our KSOD framework aims to improve the LLM itself from the perspective of knowledge.

6 Limitations

Although our study presents a promising framework for supplementing LLMs with desired knowledge on demand, its scope is limited to knowledge of categories which can be formalized as a classification task. Future research could explore the KSOD framework to other knowledge, such as knowledge of theories and knowledge of algorithms that cannot be formalized as a classification task.

7 Conclusion

In this study, we introduce a novel knowledge-based SFT framework, KSOD, to supplement knowledge missing in LLMs that causes errors in outputs of LLMs. We propose a knowledge verification method and validate its effectiveness. Our framework effectively balances the LLMs' performance across both general and knowledge-related tasks. We demonstrated the effectiveness of KSOD through LLMs with both single and combination of knowledge vectors, which outperformed pretrained LLMs on comprehensive benchmarks.

References

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2023. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*.

Eyal Ben-David, Orgad Keller, Eric Malmi, Idan Szpektor, and Roi Reichart. 2020. Semantically driven sentence fusion: Modeling and evaluation. *arXiv preprint arXiv:2010.02592*.

Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Chunkit Chan, Cheng Jiayang, Weiqi Wang, Yuxin Jiang, Tianqing Fang, Xin Liu, and Yangqiu Song. 2024. Exploring the potential of ChatGPT on sentence level relations: A focus on temporal, causal, and discourse relations. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 684–721, St. Julian's, Malta. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572.

Christopher Davis, Andrew Caines, Øistein Andersen, Shiva Taslimipoor, Helen Yannakoudakis, Zheng Yuan, Christopher Bryant, Marek Rei, and Paula Buttery. 2024. Prompting open-source and commercial language models for grammatical error correction of english learner text. *arXiv preprint arXiv:2401.07702*.

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

Yuejiao Fei, Leyang Cui, Sen Yang, Wai Lam, Zhenzhong Lan, and Shuming Shi. 2023. Enhancing grammatical error correction systems with explanations. *arXiv preprint arXiv:2305.15676*.

Mor Geva, Eric Malmi, Idan Szpektor, and Jonathan Berant. 2019. Discofuse: A large-scale dataset for discourse-based sentence fusion. *arXiv preprint arXiv:1902.10526*.

Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. Aegis2. 0: A diverse ai safety dataset and risks taxonomy for alignment of llm guardrails. In *Neurips Safe Generative AI Workshop 2024*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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*.

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024a. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Gangwei Jiang, Caigao Jiang, Zhaoyi Li, Siqiao Xue, Jun Zhou, Linqi Song, Defu Lian, and Ying Wei. 2024b. Interpretable catastrophic forgetting of large language model fine-tuning via instruction vector. *arXiv preprint arXiv:2406.12227*.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. 2023. Understanding catastrophic forgetting in language models via implicit inference. *arXiv preprint arXiv:2309.10105*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Bin Sun, Xinglin Wang, Heda Wang, and Kan Li. 2024. Turning dust into gold: Distilling complex reasoning capabilities from llms by leveraging negative data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18591–18599.

Wei Liu, Ying Qin, Zhiyuan Peng, and Tan Lee. 2024. Sparsely shared lora on whisper for child speech recognition. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11751–11755. IEEE.

Mingyu Derek Ma, Kevin Bowden, Jiaqi Wu, Wen Cui, and Marilyn Walker. 2019. Implicit discourse relation identification for open-domain dialogues. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 666–672, Florence, Italy. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the eighteenth conference on computational natural language learning: shared task*, pages 1–14.

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

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

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Huashan Sun and Yang Gao. 2024. Reviving dormant memories: Investigating catastrophic forgetting in language models through rationale-guidance difficulty. *arXiv preprint arXiv:2411.11932*.

Tianyi Tang, Hu Yiwen, Bingqian Li, Wenyang Luo, ZiJing Qin, Haoxiang Sun, Jiapeng Wang, Shiyi Xu, Xiaoxue Cheng, Geyang Guo, et al. 2024. Llmbox: A comprehensive library for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 388–399.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can llms learn from previous mistakes? investigating llms' errors to boost for reasoning. *arXiv preprint arXiv:2403.20046*.

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

Hayato Tsukagoshi, Tsutomu Hirao, Makoto Morishita, Katsuki Chousa, Ryohei Sasano, and Koichi Takeda. 2024. Wikisplit++: Easy data refinement for split and rephrase. *arXiv preprint arXiv:2404.09002*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. 2024. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. *arXiv preprint arXiv:2402.11651*.

Bonnie Webber, Rashmi Prasad, Alan Lee, and Aravind Joshi. 2019. The penn discourse treebank 3.0 annotation manual. *Philadelphia, University of Pennsylvania*, 35:108.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. 2024. Pride and prejudice: Llm amplifies self-bias in self-refinement. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15474–15492.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Jiahao Ying, Mingbao Lin, Yixin Cao, Wei Tang, Bo Wang, Qianru Sun, Xuanjing Huang, and Shuicheng Yan. 2024. LLMs-as-instructors: Learning from errors toward automating model improvement. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11185–11208, Miami, Florida, USA. Association for Computational Linguistics.

Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. 2024. On prompt-driven safeguarding for large language models. In *Forty-first International Conference on Machine Learning*.

Task Prompt →	Sentence fusion is the task of joining several independent sentences into a single coherent text. Please analyze the errors that arise in output of sentence fusion task in the given examples.
Samples →	**Example 1**: Input: In a family tiff in 1865, his remains were exhumed and reburied in the family burial plot at Cobham Park, the family home in Warren County, Pennsylvania. Cobham's wife and in-laws sued the Cobham family for their return. Target: In a family tiff in 1865, his remains were exhumed and reburied in the family burial plot at Cobham Park, the family home in Warren County, Pennsylvania. As a result, cobham's wife and in-laws sued the Cobham family for their return. Output: The remains of Charles Sanders Peirce, an American philosopher and the founder of pragmatism, were exhumed and reburied in the family burial plot at Cobham Park, the family home in Warren County, Pennsylvania. Peirce's wife and in-laws sued the Cobham family for their return. **Example 2**: Input: He retired to Boulogne-sur-Mer, so poor that the East India Company had to vote him a pension, but later returned to England and managed eventually to pay his creditors in full. Some inheritance was left for his descendants. Target: He retired to Boulogne-sur-Mer, so poor that the East India Company had to vote him a pension, but later returned to england and managed eventually to pay his creditors in full so that some inheritance was left for his descendants. Output: He retired to Boulogne-sur-Mer, so poor that the East India Company had to vote him a pension, but later returned to England and managed eventually to pay his creditors in full. Some inheritance was left for his descendants. **Example 3**: Input: They finished third among the league's eight teams, with Gore as their starting center fielder. O'Rourke had moved to left due the departure of Slattery. Target: They finished third among the league's eight teams, with Gore as their starting center fielder, while O'Rourke had moved to left due the departure of Slattery. Output: They finished third among the league's eight teams, with Gore as their starting center fielder. O'Rourke had moved to left due to the departure of Slattery. **Example 4**: Input: The Post Office committee was a regular recipient of complaints from southern states concerning the transmission of abolitionist mailings, which were seen there as incendiary; the matter was of some controversy. Southern legislators sought to have these types of mailings banned. Target: The Post Office committee was a regular recipient of complaints from southern states concerning the transmission of abolitionist mailings, which were seen there as incendiary; the matter was of some controversy because southern legislators sought to have these types of mailings banned. Output: The Post Office committee, which was a regular recipient of complaints from southern states concerning the transmission of abolitionist mailings, which were seen there as incendiary, and the matter was of some controversy. Southern legislators sought to have these types of mailings banned.
`Chain-of-thought prompt` →	Firstly, provide a step-by-step analysis for the common characteristics of the errors from all examples. Next, identify the potential knowledge lacking in LLM that may have led to these errors.

Table 5: Complement prompt of knowledge identification.

A Prompts for Knowledge Identification

The templates used for knowledge identification in the knowledge identification stage are presented in Table 5.

B Knowledge verification Settings

B.1 Details of training knowledge module.

The learning rate for both classification linear layer and LoRA is set to 5e-5. For the rank of LoRA, we select the value that yields the best model performance from 8, 16, 32, 64 for the final experiments. The target module of LoRA is the output matrix in the last self-attention layer.

B.2 Evaluation on benchmarks.

Metrics. For four general benchmarks, we use the default settings in LLMBox. For Disco Fuse task, we use SARI (Xu et al., 2016) to evaluate LLMs' performance. For GEC, we calculate F0.5 of M^2 score (Dahlmeier and Ng, 2012) to evaluate LLMs' performance.

Inference settings. For four general benchmarks, we use the default settings in LLMBox. For Disco Fuse task, we set temperature to 1.0. top-P to 0.9. For GEC task, we set temperature to 0.5, top-K to 50.