LMATE: LEARNING NEW TASKS WITHOUT FORGETTING THROUGH TEACHER-GUIDED REVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual instruction tuning seeks to fine-tune an *instruction tuned model* to a new target task while avoiding to lose its *general capabilities*. *Supervised Fine-Tuning (SFT)* force the model to match the ground-truth one-hot distribution from the dataset that may differ substantially from their existing response distribution, causing unnecessary changes and leading to catastrophic forgetting.

We present *LLM as a Teacher* (LMATE), which adopts a *teacher/student* approach that constructs task-informed training targets from the student's own answers. For each prompt, the student generates an initial response, and a stronger teacher (i.e. a larger model that also has access to ground-truth demonstration) provides targeted feedback. Conditioned on its initial response and this feedback, the student produces a revised response through *in-context learning*. We then fine-tune the student on these revised answers. To limit learning from suffixes that may drift after an early correction, we apply the training loss only to the revised prefix ending at the k_{diff}-th token-level difference. This concentrates the update on the earliest teacher-guided corrections while excluding more extensively revised suffixes. We show that corrections focused on early mistakes from the student is beneficial for adaptation and retention performance.

We evaluate LMATE on scientific question answering, tool use, and factual knowledge acquisition, measuring retention with IFEval and MMLU-Pro. LMATE achieves a competitive adaptation–retention trade-off across these tasks. On Tool-Use it reaches 66.19% accuracy compared with 61.03% for SFT, and for Timely Events it reaches 22.44 F1-score compared with 17.19 for SFT, while retaining higher IFEval and MMLU-Pro accuracy on both datasets.

1 INTRODUCTION

Despite their broad capabilities, pretrained LLMs often require *supervised fine-tuning (SFT)* on task-specific data. Such adaptation can degrade knowledge and instruction-following behaviors acquired during pretraining and alignment, a phenomenon known as *catastrophic forgetting* Ibrahim et al. (2024); Wu et al. (2024). *Continual instruction tuning (CIT)* therefore seeks to balance plasticity on the target task with retention of previously acquired capabilities Shenfeld et al. (2025; 2026); Wu et al. (2026); Chen et al. (2026); Lu & Lab (2025); Marek et al. (2026).

At the high-plasticity end of this trade-off, standard SFT minimizes the token-level negative log-likelihood of ground-truth demonstrations. Because it uses one-hot next-token targets and contains no explicit mechanism for preserving prior behavior, SFT can move the model away from its original predictive distribution Wu et al. (2026); Zhu et al. (2026); Shenfeld et al. (2026).

To study this trade-off, we introduce SELFTEACHSFT, a simple baseline that replaces the ground-truth target with a completion sampled from the frozen reference model with probability γ. Its training objective combines the negative log-likelihood of the ground-truth demonstrations with the cross-entropy between the reference and fine-tuned model distributions. Up to a constant, the latter term corresponds to the forward Kullback-Leibler divergence from the reference model to the fine-tuned model Marek et al. (2026); He et al. (2025). Consequently, γ provides a direct control over the adaptation-retention trade-off: increasing it strengthens the reference-model anchor while reducing the amount of target-task supervision.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Despite its simplicity, SELFTEACHSFT is competitive across the three adaptation tasks and achieves the highest F1-score on Timely Events (32.09), while exhibiting less degradation than SFT and SDFT on the retention benchmarks. However, for examples trained on reference-model completions, SELFTEACHSFT does not use the corresponding ground-truth demonstration, leaving potentially useful target-task information unexploited. This limitation motivates LMATE.

LMATE addresses this limitation by combining both *learning signals*, i.e. task-specific information from the ground-truth demonstrations, and responses generated by the student. For each prompt, the student first generates an initial answer. A stronger teacher with a larger model and privileged access to the ground-truth demonstration then provides targeted textual feedback, which the student uses in context to produce a revised answer. LMATE subsequently fine-tunes the student on these revised answers.

Although teacher feedback improves the correctness of the student's answer, the resulting revision can progressively depart from the student's original generation. This is especially important for autoregressive models: correcting an early decision may alter the entire subsequent trajectory, causing later tokens to reflect a substantially shifted distribution rather than a localized correction. To control this drift, LMATE uses an *edit-bounded prefix loss*. We align the initial and revised answers at the token level and apply the training loss only through the k_{diff}-th token-level difference. If the revised answer contains fewer than k_{diff} differences, we train on the complete answer. Thus, k_{diff} controls how much of the revised trajectory contributes to the update: smaller values emphasize early, localized corrections, whereas larger values incorporate more of the teacher-guided revision.

Using targeted feedback and edit-bounded prefix fine-tuning, LMATE is designed to concentrate learning on task-relevant corrections while limiting unnecessary changes to the student's response, preserving tone, personality and vocabulary. Across the three adaptation tasks, LMATE achieves competitive target-task performance while generally reducing degradation relative to SFT. It attains higher mean performance on Tool-Use (66.19% versus 61.03% accuracy) and Timely Events (22.44 versus 17.19 F1), while remaining competitive on Science Q&A (63.91% versus 67.85% accuracy). Across the six retention evaluations — IFEval and MMLU-Pro following adaptation to each target dataset — LMATE exhibits less performance degradation than SFT in five cases.

Contributions. As contributions of our work, we:

- Propose LMATE, a teacher-student continual instruction tuning framework that uses ground-truth-informed feedback to transform student-generated responses into task-informed training targets;
- Introduce an edit-bounded prefix objective that uses token-level differences to control how much of each revised trajectory contributes to training;
- Formulate SELFTEACHSFT, a simple reference-sampling baseline that exposes a controllable adaptation-retention trade-off and connects sample mixing to KL-regularized fine-tuning; and
- Evaluate these methods across three complementary adaptation settings, jointly measuring target-task learning and retention of general capabilities, and analyze how successive feedback rounds affect the quality of the revised responses.

2 RELATED WORKS

Continual Instruction Tuning One line of works on CIT trains the model on the new task, while using a KL divergence constraint with a reference copy of the pre-trained model to preserve the general knowledge capabilities He et al. (2025); Chen et al. (2023); Li & Hoiem (2018); Jaques et al. (2017); Ziegler et al. (2019). Although effective, this strategy is very memory-expensive since it requires storing a costly copy of the reference model in memory, unless PEFT techniques are used like LoRA. Another recent research trend are some works advocating against the one-hot training objective in SFT, that causes the model to generalize poorly Wu et al. (2026); Zhu et al. (2026); Sun et al. (2025). Some of these works use a RL based view of SFT, and show it represents an ill-defined reward function. DFT Wu et al. (2026) argues that SFT ends-up multiplying the token-level gradients by a large value when the token is unlikely for the model. They correct this using an importance sampling factor, that converts the training procedure with off-policy trajectories into on-policy. ASFT

builds upon DFT, showing that it can be generalized as a *reward re-weighted regression* problem Zhu et al. (2026). TALR Lin et al. (2026) is motivated from an optimization perspective in which weights for harder tokens should be small, yet the entropy of weights should be minimized. This motivates a similar solution to DFT, in which the token-level cross-entropy is multiplied by the model probability scaled by a τ factor. All these works support the claim that training on surprising tokens is harmful for forgetting. In LMATE we minimize unexpected tokens for the model by building upon its own answer, including targeted changes to correct mistakes.

Answer revision methods Previous methods in the literature have explored the idea of improving an answer generated by the model being trained. In CGD Kapusuzoglu et al. (2026), the student answer revised by a teacher, that creates a critique for the answer and also creates a new version. However, the teacher does not have access to the ground-truth demonstration, but it is able to critique the student answer since it a much larger model. Finally, the student is trained using SFT (i.e. forward KL), conditioned on all the information: prompt, initial answer and critique. In STaR Zelikman et al. (2022), the model is asked to create a reasoning trace and answer to each question in the dataset. Then, they select the correct rationale/answer pairs and build a reasoning dataset to train the model on. For the questions with a wrong answer, the authors perform what they call as "rationalization", which consists of providing hints to the model, and asking it to create a new rationale for the question. In SPIN Chen et al. (2024), the model is trained using an iterative procedure, in which the model learns to generate answers that increasingly resemble the training data. At every iteration of the algorithm, the generated answers (rejected option) are refined to be sampled from the latest updated model.

3 METHODOLOGY — FROM SELFTEACHSFT TO LMATE

3.1 PROBLEM FORMULATION AND MOTIVATION

We consider a continual instruction-tuning scenario in which an instruction-tuned LLM is adapted to a *target task* using a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is an input prompt and y_i is the corresponding ground-truth response. The target task may require a skill or factual knowledge that the initial model does not reliably exhibit.

In addition to improving target-task performance, we seek to preserve capabilities acquired by the reference model during *pretraining, instruction tuning, and alignment*. Directly measuring retention of these capabilities is difficult because the original training data and the details of how they were used are generally unavailable. We therefore estimate forgetting using proxy benchmarks that evaluate *general knowledge* and *instruction-following abilities*. Specifically, we quantify forgetting as the performance degradation of the adapted model relative to the reference model on these benchmarks.

The main goal of training is to fine-tune an instruction tuned model $p_\theta(.)$ to learn how to perform the new task while retaining the general knowledge abilities. The standard approach is to perform training using SFT on the ground-truth demonstrations:

$$\mathcal{L}_{SFT}(\theta) = -\mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}\big[\log p_\theta(y_i \mid x_i)\big] \tag{1}$$

A key limitation of SFT is that its objective focuses exclusively on adaptation to the target task, without explicitly promoting the retention of previously acquired capabilities. The objective function $\mathcal{L}_{SFT}(\theta)$ described above only samples from the new task dataset, and makes the model to mimic the dataset distribution, completely ignoring its own distribution acquired through pretraining, instruction tuning and alignment.

3.2 SELFTEACHSFT

We introduce SELFTEACHSFT, a baseline that uses completions sampled from a frozen reference model to mitigate forgetting during target-task adaptation. For each prompt x, a reference completion $\hat{y} \sim p_{ref}(. \mid x)$ serves as an anchor: training the student model $p_\theta(. \mid x)$ to assign high likelihood to $\hat{y}$ encourages its conditional output distribution to remain close to $p_{ref}(. \mid x)$.

Like standard SFT, SELFTEACHSFT optimizes a next-token prediction objective conditioned on the input prompt x_i. For each training example, however, the target is the ground-truth response y_i with

probability $1-\gamma$, and a completion $\hat{y}_i \sim p_{ref}(. \mid x_i)$ sampled from the frozen reference model with probability γ.

Formally, let $p_\theta(.)$ be the model being trained at every iteration, and denote $p_{ref} = p_{\theta_0}$ as the reference model which is the pre-trained model with initial weights θ_0. Then, we can define the loss function used in SELFTEACHSFT $\mathcal{L}_{\text{SelfTeach}}(\theta;\gamma)$ as:

$$\begin{aligned}\mathcal{L}_{\text{SelfTeach}}(\theta;\gamma) = &-(1-\gamma)\mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}\big[\log p_\theta(y_i \mid x_i)\big] \\ &-\gamma\mathbb{E}_{x_i\sim\mathcal{D}}\mathbb{E}_{\hat{y}_i\sim p_{ref}(.|x_i)}\big[\log p_\theta(\hat{y}_i \mid x_i)\big] \end{aligned} \tag{2}$$

The first term in the equation above is the $(1-\gamma)$-scaled cross entropy loss from SFT (see equation 1), while the second term is a γ-scaled cross-entropy loss between the distribution of the reference model $p_{ref}(. \mid x)$ and the training model $p_\theta(. \mid x)$. The second term discourages the student's conditional output distribution from deviating from that of the reference model, therefore promoting retention of previously acquired capabilities. We can reformulate the loss term above as:

$$\mathcal{L}_{\text{SelfTeach}}(\theta;\gamma) = (1-\gamma)\mathcal{L}_{SFT} + \gamma\mathbb{E}_{x_i\sim\mathcal{D}}D_{\text{KL}}(p_{ref} \parallel p_\theta) + C \tag{3}$$

In which $C = \gamma\mathbb{E}_{x_i\sim\mathcal{D}}\mathbb{E}_{\hat{y}_i\sim p_{ref}(.|x_i)}\big[-\log p_{ref}(\hat{y}_i \mid x_i)\big]$ is a constant w.r.t. θ. We approximate the KL term sampling from $p_{ref}(. \mid x)$, which enables us to minimize both terms using the same next-token prediction objective. In Appendix C, we give more details to the empirical sampling strategy.

Trade-off controlled by γ. The mixture probability γ controls the adaptation-retention trade-off. When $\gamma = 0$, SELFTEACHSFT reduces to standard SFT, placing full weight on target-task supervision and providing no explicit mechanism for retention. When $\gamma = 1$, training uses only reference-model completions, prioritizing the preservation of reference-model behavior while providing no direct supervision from the ground-truth target responses. Appendix E provides a detailed discussion of the effect of γ, together with empirical results for different values.

Limitations of target mixing The data-switching strategy in SELFTEACHSFT selects one of two complementary supervision sources for each training example, leaving the other unused. When the selected target is a reference-model completion $\hat{y} \sim p_{ref}(. \mid x)$, the corresponding ground-truth demonstration y does not contribute to the update. Conversely, when the ground-truth demonstration y is used as the sole training target, the objective provides no explicit anchor to the reference distribution $p_{ref}(.)$. In Section 4, we introduce LMATE, which combines information from both y and $\hat{y}$ by relying on a teacher model, that is able to see the ground-truth demonstration for a given input prompt and provide feedback for a student model on how to improve its own generated completions.

4 LLM AS A TEACHER

We assume a student model $p_\theta(. \mid x)$, parametrized by the weights θ, and a teacher model $p_{teacher}(. \mid x, y)$. We assume the teacher to be an oracle for the task that is able to give valuable feedback for the student. In order to obtain this high-quality teacher, we use an independent stronger model — i.e. a model with $10\times$ the number of the parameters of the student — and also provide the teacher with the ground-truth demonstration in its input.

Given an input prompt, the student samples an answer from its policy $p_\theta(.)$ which is revised given the feedback from the teacher. The student then creates a new version of the answer based on the feedback, and it is trained on the improved version.

LMATE works in the following steps:

1. For each input prompt x_i in the dataset, we first obtain the student's output for that prompt, i.e. we compute $\hat{y}_i \sim p_\theta(. \mid x_i)$;

2. Then we use the teacher to obtain a targeted textual feedback f_i for the student answer $f_i \sim p_{teacher}(. \mid x_i, y_i, \hat{y}_i)$. The teacher is conditioned on the correct ground truth answer, which is leveraged using in-context learning and transforms the teacher into an oracle that is able to give valuable feedback to the student. For more details on the process of generating a high quality feedback, see Section 4.1, and for a presentation of the used prompts see Appendix D;
3. We sample from the student, using the teacher feedback f_i and the student's own previous answer as conditions, prompting it to improve its previous answer into a revised version $\tilde{y}_i \sim p_\theta(. \mid x_i, \hat{y}_i, f_i)$;
4. Finally, we create a training dataset using the original prompts and the revised answers $\widetilde{\mathcal{D}} = \{(x_i, \tilde{y}_i)\}_{i=1}^N$, and use it to update the model using SFT. For more details on our fine-tuning procedure, see Section 4.2.

We repeat steps 2 and 3 in several rounds, in order to obtain increasingly improved versions of the student answer. An example with one round of feedback revision in LMATE can be seen in figure 1. Please see Algorithm 1 for a pseudo-code of LMATE.

Example: One round of feedback from teacher

Input question: A fair coin is tossed three times. What is the probability that the first heads happens in the third toss?

Student answer: <reasoning> Let me think: each toss is independent and the probability of each outcome is $\frac{1}{2}$. [...] Considering that there must be exactly one heads among the three tosses, then there are 3 possibilities out of the 8 possible outcomes. </reasoning> <answer> $\frac{3}{8}$ </answer>

Teacher feedback: Your assumption that the event "heads being tossed only once" is equivalent to "the first heads coming up on the third toss" is wrong. For the first heads to happen in the third toss, the first two tosses have to be tails.

Revised student answer: <reasoning> Let me think: each toss is independent and the probability of each outcome is $\frac{1}{2}$. [...] Considering that there must be exactly two tails in the first two tosses, and a heads in the third one, the probability of this event is $\frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{8}$. </reasoning> <answer> $\frac{1}{8}$ </answer>

Figure 1: Example: One round of feedback from teacher.

Teacher/student as proxy for data and model distributions By using a teacher/student approach, we provide the student with rich textual feedback that makes it improve its own answers. The teacher is a proxy between the student and the ground truth demonstrations, acting like a selective mechanism to guide the student into correcting the important parts of the answer, and ignoring irrelevant parts like tone and style. Yet, by using y_i as conditioning on the teacher, and by making the student to generate $\tilde{y}$ as an improvement from its initial answer, we effectively combine the relevant information from both the training data demonstrations and the reference model distribution.

Theoretical connections for LMATE LMATE has some interesting theoretical connections with some concepts in Reinforcement Learning and with self-distillation. In Appendix A.1 we show how to interpret LMATE as a self-distillation from an enhanced student; and in Appendix A.2 we further show how this can be used to derive a *near-policy* interpretation for LMATE.

4.1 WHAT MAKES A GOOD TEACHER FEEDBACK

Given an initial answer of the student $\hat{y} \sim p_\theta(. \mid x)$, the main goal of the teacher is to create a feedback f that addresses the key decisions taken by the student that led to an incorrect conclusion. Ideally, correcting the earliest wrong decisions is better, since the early tokens produced by the student condition the generation of all the next tokens in the completion. Yet, the feedback should not focus on nuisance factors, like style, personality and tone. In the example in figure 1 the feedback from the teacher addresses the first incorrect decision from the student, and it preserves the answer

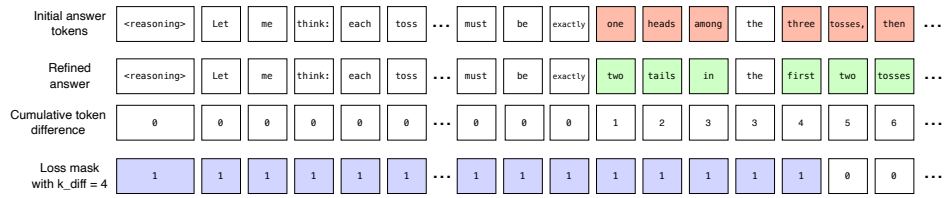


Figure 2: Visualization for the token-level difference set. First, we calculate the token-level difference between the initial student answer and the revised answer (first and second rows). Then we check, for each token in the revised answer, which tokens belong to the difference set, and calculate a cumulative count of the tokens in the `diffset` for the entire sequence (third row). We create a mask that is 1 for the shortest prefix $\tilde{y}_{1:j}$ such that it has at least k_{diff} tokens in the `diffset`. If the sequence has less than k_{diff} difference tokens (or even zero), then we train on the whole sequence.

(i.e. the student *trajectory*) up until that point, and makes the student to follow a correct path from that point onward. The teacher does not critique aspects that are irrelevant for answering the question, like the stylistic format taken by the student to start with *"Let me think: [...]"*. This is different from what happens in other methods like SFT, in which the student is trained to mimic *the entire* ground-truth demonstration, including its nuisance factors. Training to mimic the nuisance factors hurts generalization due to the exposure bias problem (see Appendix B), and updates the model more than necessary, also hurting knowledge retention.

4.2 Edit-bounded prefix fine-tuning

By using empirical samples, it is possible to compare the revised answers $\tilde{y}$ with the initial student answers $\hat{y}$. We calculate the token-level differences between $\hat{y}$ and $\tilde{y}$ by tokenizing both sequences, and checking for each token in $\tilde{y}$ if it was changed from $\hat{y}$ — similar to how `diffsets` can be calculated from two snapshots of the same code or text. This information is important since it represents the tokens in the revised answer that were modified after the student had access to the teacher feedback. See figure 2 for an example of `diffset` calculation in LMATE.

When fine-tuning the student, we constrain the loss to be calculated in a prefix of $\tilde{y}$, until the first k_{diff} differences are observed in the revised answer. This constrains updates to the completion until a point in the revised answer where it is too different from the original answer. This is important to avoid updating the model on samples that were drawn from a distribution completely different from the student. Limiting the model updates helps LMATE to follow the requirement mentioned in Section 4.1 to make targeted changes to key decisions *as early as possible* in the student answer. In Appendix H, we show results comparing different values of k_{diff}.

5 Experimental Setup

Our experimental setup is designed to evaluate the model performance across two main dimensions: *new task learning* and *general knowledge retention*. We adopt a similar evaluation setup as Shenfeld et al. (2026).

To assess retention of previously acquired capabilities, we compare the model's performance before and after target-task adaptation on two proxy benchmarks: IFEval Zhou et al. (2023), which evaluates instruction following, and MMLU-Pro Wang et al. (2024), which evaluates broad knowledge and reasoning. We interpret a decrease relative to the reference model's performance as empirical evidence of forgetting.

For new task learning, we measure the accuracy and F1-score on the test set of the new task. The training datasets we use are the following:

- **Tool-Use dataset**, which is a subset of the ToolAlpaca dataset Tang et al. (2023), consisting of documentations of public APIs. The input prompts ask the model to make a correct tool call given one question;

- **Science Q&A**, which consists of chemistry questions from the `L3` level of the SciKnowEval dataset Feng et al. (2025). These questions require the model to reason about the question before picking one of the multiple-choice answers;
- **Timely Events Benchmark (TiEBe)** Almeida et al. (2025), which evaluates knowledge of recent real-world events through questions associated with different geographic regions. We use the subset corresponding to regions in which English is the predominant language.

For Tool-Use and Science Q&A, which we treat as skill-learning tasks, we evaluate target-task performance using answer accuracy. For Timely Events, which assesses the acquisition of recent factual knowledge through open-ended generation, we report token-level F1 between the generated response and the ground-truth answer.

We use models from the `Qwen2.5` family in all experiments unless stated otherwise, with `Qwen2.5-3B-Instruct` as the student and `Qwen2.5-32B-Instruct` as the teacher. During the LMATE data-construction stage, both models remain fixed: the student generates an initial response, the teacher provides feedback, and the student produces a revised response. Parameter updates are performed only after the final revised training dataset has been constructed. We use `vLLM` Kwon et al. (2023) for efficient batched generation and KV-cache management through PagedAttention. Because the teacher is used only for inference, it requires neither gradient storage nor optimizer states, making it feasible to use a substantially larger teacher than the student.

As the baselines, we test several different models, ranging from classic standard baselines to methods that also leverage on-policy ideas on their design:

- **Supervised Fine-Tuning (SFT)** is the standard approach for post-training, in which the student model is directly trained to replicate the ground-truth demonstrations from the dataset;
- **SELFTEACHSFT** is a simple baseline similar to SFT, but with probability γ we inject demonstrations in the training dataset that are generated by the student model. By comparing this baseline with LMATE, we can evaluate the effect of the teacher feedback on the student samples and consequently in the adaptation versus retention tradeoff;
- **Self-Distillation Fine-Tuning (SDFT)** Shenfeld et al. (2026) is also an on-policy teacher/student approach, in which the teacher provides feedback through self-distillation to the student;
- **Dynamic Fine-Tuning (DFT)** Wu et al. (2026) is a post-training method in which the objective function is modified by a token-level importance sampling ratio, that bias cross entropy towards the model distribution, effectively simulating on-policy training with off-policy samples.

We present the hyperparameters used for training and model generation in Appendix I.

6 RESULTS

Next we present *skill-learning* results on Tool-Use and Science Q&A, and *knowledge-acquisition* results on Timely Events. Target task performance represents the accuracy or F1-score of the model on the test set of each dataset. *Knwoledge retention* is measured in IFEval and MMLU-Pro. For IFEval, we report the `prompt-strict` accuracy, and for MMLU-Pro the Exact Match. All reported values are the average $\pm$ standard deviation of accuracy/F1-score calculated over 5 runs with different random seeds.

6.1 SKILL LEARNING DATASETS

First we present the results for the Skill learning datasets, namely Science Q&A and Tool-Use, in table 1. Considering the target task accuracy, we observe that SFT and LMATE obtain the best results. For Science Q&A, SFT improves upon the baseline from 31.08% to 67.85%, while LMATE comes in second place with 63.91%. For Tool-Use, still SFT and LMATE get the best results, but this time LMATE has the highest score, improving the base model from 35.67% to 66.19%, while SFT achieves 61.03%. It is expected for SFT to achieve high target task performance, since it is aimed only at

Table 1: Skill-learning results on Tool-Use and Science Q&A. All methods use learning rate 5×10^{-5}, and for LMATE we used $k_{\text{diff}} = 64$.

	Science Q&A			Tool-Use		
Method	Target ↑	IFEval ↑	MMLU-Pro ↑	Target ↑	IFEval ↑	MMLU-Pro ↑
Base *(Qwen2.5-3B-Instruct)*	31.08 ± 0.18	59.74 ± 0.72	42.86 ± 0.10	35.67 ± 0.56	59.74 ± 0.72	42.86 ± 0.10
SFT	$\mathbf{67.85} \pm 1.27$	$\mathbf{56.30} \pm 0.40$	30.99 ± 2.30	$\underline{61.03} \pm 2.95$	45.95 ± 1.45	29.84 ± 0.63
DFT	56.69 ± 4.01	$\underline{50.68} \pm 4.99$	34.18 ± 1.50	52.99 ± 2.59	41.44 ± 2.21	23.41 ± 6.96
SDFT	62.84 ± 1.49	44.55 ± 2.20	$\mathbf{40.50} \pm 0.50$	56.29 ± 4.41	$\mathbf{51.05} \pm 0.81$	$\mathbf{41.87} \pm 0.45$
LMATE (ours)	$\underline{63.91} \pm 2.56$	50.20 ± 1.46	$\underline{36.47} \pm 1.15$	$\mathbf{66.19} \pm 1.69$	$\underline{48.80} \pm 2.03$	$\underline{32.79} \pm 0.64$

Table 2: Skill-learning results comparing with SELFTEACHSFT. We pick two values for $\gamma \in \{0.1, 0.5\}$, that represent well two extremes of more adaptation (with $\gamma = 0.1$) and more knowledge retention (with $\gamma = 0.5$). All methods use learning rate 5×10^{-5}, and for LMATE we used $k_{\text{diff}} = 64$.

	Science Q&A			Tool-Use		
Method	Target ↑	IFEval ↑	MMLU-Pro ↑	Target ↑	IFEval ↑	MMLU-Pro ↑
Base *(Qwen2.5-3B-Instruct)*	31.08 ± 0.18	59.74 ± 0.72	42.86 ± 0.10	35.67 ± 0.56	59.74 ± 0.72	42.86 ± 0.10
SELFTEACHSFT ($\gamma = 0.1$)	$\mathbf{66.55} \pm 1.01$	$\underline{50.61} \pm 2.13$	$\underline{39.28} \pm 0.52$	$\underline{65.15} \pm 3.82$	48.10 ± 2.20	28.81 ± 1.34
SELFTEACHSFT ($\gamma = 0.5$)	52.27 ± 2.35	$\mathbf{54.23} \pm 2.00$	$\mathbf{40.87} \pm 0.44$	61.24 ± 1.56	$\mathbf{53.16} \pm 1.14$	$\mathbf{34.16} \pm 1.84$
LMATE	$\underline{63.91} \pm 2.56$	50.20 ± 1.46	36.47 ± 1.15	$\mathbf{66.19} \pm 1.69$	$\underline{48.80} \pm 2.03$	$\underline{32.79} \pm 0.64$

adaptation with no mechanism to prevent forgetting. Still, LMATE achieves competitive results in Science Q&A, and even outperforms SFT on Tool-Use.

Although SFT has better results for target task accuracy, it tends to suffer more from forgetting. In fact, in our experiments we observe that SFT has large accuracy drops for the forgetting benchmarks, with the exception of IFEval for Science Q&A, in which SFT has the least forgetting with 56.30% final accuracy. DFT comes in second place with 50.68%, and LMATE as a close third with 50.20%.

For the other forgetting metrics — i.e. MMLU-Pro on Science Q&A, and both MMLU-Pro and IFEval on Tool-Use — SDFT produces the best metrics, but this comes with a cost of sacrificing adaptation to the target task. SDFT uses samples from the student policy in order to learn from the teacher. When the new task is significantly novel to the student, it might be challenging to create roll-outs that cover all possible ways in which the teacher might provide useful information. We argue that the LMATE approach of making targeted changes is better, since the trajectories might be "adjusted" from the teacher feedback. Moreover, LMATE has the second best results for these forgetting metrics (excluding IFEval on Scince Q&A), but still is competitive with SFT on the target task accuracy.

DFT struggles to achieve competitive results, except for IFEval on Science Q&A. Similar to SDFT, this might also be caused by the degree of surprise from the model when facing a new task not related to pre-trained knowledge. Low probability tokens caused by the new domain make the updates from DFT have less impact on learning.

Overall, LMATE achieves a competitive adaptation-retention trade-off, maintaining strong target-task performance while limiting degradation on the retention benchmarks.

Finally, we also compare LMATE with SELFTEACHSFT, in table 2. Clearly SELFTEACHSFT with $\gamma = 0.5$, is more focused on knowledge retention, and it retains the best accuracy values for all forgetting benchmarks. On the other hand, it has the worst target task accuracies. LMATE and SELFTEACHSFT with $\gamma = 0.1$ have similar results, with SELFTEACHSFT being better overall for the Science Q&A dataset, with target accuracy of 66.55% compared with 63.91% from LMATE. Conversely, in Tool-Use LMATE has a higher accuracy of 66.19% compared with 65.15% from SELFTEACHSFT with $\gamma = 0.1$.

6.2 KNOWLEDGE ACQUISITION DATASET

We report the results for the Knowledge acquisition task in table 3. LMATE has a good compromise between new knowledge learning and preserving general capabilities. While some methods have

better score on the target task and others have better accuracy in forgetting tasks, LMATE stands out as a competitive method in both adaptation and knowledge retention.

Surprisingly, we observe that SELFTEACHSFT ($\gamma = 0.1$) achieves the best target task F1-score of 32.09, surpassing even SFT (17.19) which is purely focused on adaptation. LMATE is the second best with 22.44. For Timely Events, we observe that preserving the instruction following abilities of the model has more impact on the evaluation of the model *even for target task score*. This is because, when the model loses instruction following abilities, it has a higher chance of producing an output that is not well-formatted, which prevents properly evaluating the model predictions. Therefore, SELFTEACHSFT ($\gamma = 0.1$) is able to both focus on adaptation to the new task, and still preserve instruction following abilities due to training on the 10% reference samples, which leads to higher target F1-score.

For the forgetting benchmarks, DFT got the best results for both IFEval (59.70%) and MMLU-Pro (42.84%), but it loses too much on the adaptation side, since it achieves an F1-score of only 15.68. Similar to the skill learning results, we hypothesize this could be caused by training on several tokens that are unexpected by the model, which DFT down-weights using the importance sampling term.

LMATE is again competitive, reaching the second best result on IFEval (58.63%) and having a close gap with the second best in MMLU-Pro (42.05% vs. 42.65% from SELFTEACHSFT with $\gamma = 0.5$).

7 CONCLUSION

In this paper, we presented LMATE, which is a *continual instruction tuning* method. LMATE works by combining the information from the model base distribution and the information from the ground-truth dataset demonstrations. We combine them by creating textual feedback that informs a student model how its own answers can be improved. The feedback is created by a powerful independent teacher model, and we rely on the *in-context learning abilities* from the student to correctly implement these changes. LMATE is able to achieve competitive results in adaptation tasks and retain general knowledge from the model.

Table 3: Knowledge-acquisition results on Timely Events. All methods use learning rate 2×10^{-5}, and for LMATE we used $k_{\text{diff}} = 16$.

Method	F1 ↑	IFEval ↑	MMLU-Pro ↑
Base *(Qwen2.5-3B-Instruct)*	7.70 ± 0.74	59.74 ± 0.72	42.86 ± 0.10
SFT	17.19 ± 1.51	55.60 ± 0.73	40.84 ± 0.44
DFT	15.68 ± 0.59	$\mathbf{59.70} \pm 1.05$	$\mathbf{42.84} \pm 0.43$
SDFT	13.57 ± 1.01	54.31 ± 1.07	40.76 ± 0.24
SELFTEACHSFT ($\gamma = 0.1$)	$\mathbf{32.09} \pm 1.68$	56.23 ± 0.53	41.87 ± 0.41
SELFTEACHSFT ($\gamma = 0.5$)	18.56 ± 3.38	57.12 ± 0.57	$\underline{42.65} \pm 0.33$
LMATE (ours)	$\underline{22.44} \pm 1.38$	$\underline{58.63} \pm 1.28$	42.05 ± 0.27

The method presented in this paper has some limitations, which suggests several interesting future research directions. The iterative answer refinement from LMATE depends on the ability of the student to follow teacher's instructions, and it is worth investigating approaches for improving these instruction following abilities from the student. Another limitation is regarding the quality of the teacher feedback for more challenging datasets in which few feedback rounds are not enough to correct the student answer, for example when the input question is very complex or requires a lot of context to be answered.

AI USE STATEMENT

In this work, we used generative AI tools for writing part of the implementation, improving the readability of the paper, creating figures and tables, formatting results, proposing the title, and suggesting experimental parameters. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Thales Sales Almeida, Giovana Kerche Bonás, João Guilherme Alves Santos, Hugo Abonizio, and Rodrigo Nogueira. Tiebe: Tracking language model recall of notable worldwide events through time, 2025. URL https://arxiv.org/abs/2501.07482.

Howard Chen, Noam Razin, Karthik R Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ODTM64azGa.

Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. Lifelong language pretraining with distribution-specialized experts. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning convertsweak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. Sciknoweval: Evaluating multi-level scientific knowledge of large language models, 2025. URL https://arxiv.org/abs/2406.09098.

Yongquan He, Wenyuan Zhang, Xuancheng Huang, Peng Zhang, Lingxun Meng, Xiang Zhou, Ke Zeng, and Xunliang Cai. Don't half-listen: Capturing key-part information in continual instruction tuning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23649–23668, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1153. URL https://aclanthology.org/2025.acl-long.1153/.

Jonas Hübotter, Frederike Lübeck, Lejs Deen Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats Leon Richter, Quentin Gregory Anthony, Eugene Belilovsky, Timothée Lesort, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=DimPeeCxKO.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E. Turner, and Douglas Eck. Sequence tutor: conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1645–1654. JMLR.org, 2017.

Berkcan Kapusuzoglu, Supriyo Chakraborty, Zain Sarwar, Chia-Hsuan Lee, and Sambit Sahu. Critique-guided distillation for robust reasoning via refinement. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, December 2018. ISSN 0162-8828. doi: 10.1109/TPAMI.2017.2773081. URL https://doi.org/10.1109/TPAMI.2017.2773081.

Jiacheng Lin, Zhongruo Wang, Kun Qian, Tian Wang, Arvind Srinivasan, Hansi Zeng, Ruochen Jiao, Xie Zhou, Jiri Gesi, Dakuo Wang, Yufan Guo, Kai Zhong, Weiqi Zhang, sujay sanghavi, Changyou Chen, Hyokun Yun, and Lihong Li. SFT doesn't always hurt general capabilities: Revisiting domain-specific fine-tuning in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ru93xpQFi1.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Martin Marek, Dongkyu Cho, Shikai Qiu, Rumi Chunara, Pavel Izmailov, and Andrew Gordon Wilson. Forgetting in language models: Capacity, optimization, and self-generated replay, 2026. URL https://arxiv.org/abs/2605.26097.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less, 2025. URL https://arxiv.org/abs/2509.04259.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Chen Sun, Renat Aksitov, Andrey Zhmoginov, Nolan Andrew Miller, Max Vladymyrov, Ulrich Rueckert, Been Kim, and Mark Sandler. How new data permeates LLM knowledge and how to dilute it. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=NGKQoaqLpo.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases, 2023. URL https://arxiv.org/abs/2306.05301.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey, 2024. URL https://arxiv.org/abs/2402.01364.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of SFT: A reinforcement learning perspective with reward rectification. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Lv7PjbcaMi.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: self-taught reasoner bootstrapping reasoning with reasoning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

He Zhu, Junyou Su, Peng Lai, Ren Ma, Wenjia Zhang, Linyi Yang, and Guanhua Chen. Anchored supervised fine-tuning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=PORko7QT64.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593, 2019. URL http://arxiv.org/abs/1909.08593.

A RL INTERPRETATION FOR LMATE

A.1 LMATE AS AN APPROXIMATION FOR SELF-DISTILLATION

In the revision step of LMATE described in Section 4, we condition the student on the teacher feedback and its previous answer in order to generate the revised answer $\tilde{y} \sim p_\theta(. \mid x, \hat{y}, f)$. We can see the process of conditioning the student using f and $\hat{y}$ as transforming it into an *enhanced student*, that has access to privileged information $(f, \hat{y})$, and uses *in-context learning* to represent a more powerful model. This *enhanced student* $p^\star_\theta(. \mid x) := p_\theta(. \mid x, f, \hat{y})$ is able to create better samples, but is still following a distribution close to the base student $p_\theta(. \mid x)$.

Given a triplet $(x, f, \hat{y})$ of input prompt, teacher feedback and initial student answer, if we were to perform self-distillation from the enhanced student to the base student, we could use a forward KL divergence loss, given by:

$$D_{\mathrm{KL}}(p^{\star}_{\theta}(. \mid x) \,\|\, p_{\theta}(. \mid x)) = \mathbb{E}_{y \sim p^{\star}_{\theta}(.|x)}\big[\log p^{\star}_{\theta}(y \mid x) - \log p_{\theta}(y \mid x)\big] \tag{4}$$

The enhanced student $p^{\star}_{\theta}(. \mid x)$ is not a separate model; it is the same student conditioned additionally on its previous answer and the teacher feedback. During answer revision, we keep the student parameters fixed and use $p^{\star}_{\theta}$ only to generate revised responses. We update the base student $p_{\theta}(\cdot \mid x)$ only afterward, using the generated revisions as training targets. Hence, when minimizing the KL divergence above, the first term in the expectation is constant w.r.t θ and can be ignored, and we are left with $\mathbb{E}_{y \sim p^{\star}_{\theta}(.|x)}\big[-\log p_{\theta}(y \mid x)\big]$, which represents a cross-entropy loss of the base student $p_{\theta}(.)$ using the enhanced student $p^{\star}_{\theta}(.)$ as the target distribution. This expectation can be approximated by using samples $\tilde{y} \sim p^{\star}_{\theta}(. \mid x)$, and training $p_{\theta}(.)$ with next-token prediction, which is equivalent of performing SFT training on the base student $p_{\theta}(.)$ using samples from $p^{\star}_{\theta}(.)$, given we sample enough roll-outs.

A.2 SFT ON ENHANCED STUDENT AS NEAR-POLICY LEARNING

Recent literature has shown that on-policy learning enables training of the model without losing much of its pre-trained capabilities Lu & Lab (2025); Shenfeld et al. (2025); Hübotter et al. (2026); Shenfeld et al. (2026); Chen et al. (2026).

For an on-policy training procedure, we would usually sample from the student conditioned only on the input prompt, but unconditioned on the feedback and previous answer $p_{\theta}(. \mid x)$. In our case, instead we sample from the the student's enhanced version $p^{\star}_{\theta}(. \mid x) \coloneqq p_{\theta}(. \mid x, f, \hat{y}) \neq p_{\theta}(. \mid x)$, and it is uncertain whether in general it follows the same distribution as the unconditioned student, since the model uses in-context learning abilities to calibrate its output distribution.

Therefore, LMATE relies on the following assumptions that help the student stay grounded in its unconditioned version: 1) by conditioning on the previous answer, we bias the student to *partially repeat* its previous answer, since it already represents one syntactically valid response to the input prompt. This process results in a revised answer that is plausible under the original student model; and 2) the textual feedback provides targeted suggestions to the student answer, resulting in a revised answer that is similar to the initial student answer, but with small changes that increase its probability of being correct.

B REVISION AS A MEANS TO REDUCE EXPOSURE BIAS

A well-known problem in deploying machine learning models in production is the mismatch between the data observed during training and the data observed in test time. This distribution shift usually incurs in performance drop since the model was not trained on the input. In the case of LLMs, this problem is yet more important because, by using an autoregressive model, the current context is given by tokens predicted by the model itself in previous iterations. More specifically, when used in production, at every token prediction, the autoregressive model is conditioned in the input prompt created by the user and the beginning of the completion created by the model.

This represents a distribution shift to the training data, in which we use *teacher forcing* with the tokens observed in the training data. Although it is important to expose the model to the completions in the training data — so it learns how to solve the problems posed in the input prompt — mimicking these the ground truth demonstrations force the model to predict and be exposed to tokens that are related to other aspects different than the target task, like tone, personality, and vocabulary.

In LMATE we let the *student* generate an answer with its own words, and have this answer to be evaluated by the *teacher*. The student then uses the teacher feedback in order to make targeted changes to the answer, in order to make it a valid correct solution to the input prompt.

C SAMPLING APPROXIMATION TO KL CONSTRAINT

When training a model for instruction tuning with catastrophic forgetting mitigation, a common strategy is to constrain the model to be close in output space with the reference model, by using a

Kullback-Leibler divergence between the conditional distributions of the reference model $p_{ref}(.)$ and the trained model $p_\theta(.)$. Using an input prompt x, we have:

$$D_{\mathrm{KL}}(p_\theta(. \mid x) \,\|\, p_{ref}(. \mid x)) = \mathbb{E}_{y \sim p_\theta(.|x)} \big[\log p_\theta(y \mid x) - \log p_{ref}(y \mid x) \big] \tag{5}$$

In this standard formulation, we sample completions from the current model being trained $y \sim p_\theta(. \mid x)$, and calculate the logits from the reference model to be distilled to the training model $p_\theta(. \mid x)$. If instead we take the inverse direction on the KL divergence, i.e. $D_{\mathrm{KL}}(p_{ref}(. \mid x) \,\|\, p_\theta(. \mid x))$, we obtain a slightly different formulation, but in the new penalty we sample the completions from the reference model, i.e. $y \sim p_{ref}(. \mid x)$. The KL divergence becomes:

$$\begin{aligned} D_{\mathrm{KL}}(p_{ref}(. \mid x) \,\|\, p_\theta(. \mid x)) = & \\ & \mathbb{E}_{y \sim p_{ref}(.|x)} \big[\log p_{ref}(y \mid x) - \log p_\theta(y \mid x) \big] = \\ & \mathbb{E}_{y \sim p_{ref}(.|x)} \big[\log p_{ref}(y \mid x) \big] \\ & \quad - \mathbb{E}_{y \sim p_{ref}(.|x)} \big[\log p_\theta(y \mid x) \big] \end{aligned} \tag{6}$$

The first term in this formulation $\mathbb{E}_{y \sim p_{ref}(.|x)} \big[\log p_{ref}(y \mid x) \big]$ is not dependent on the parameters θ and therefore can be ignored. Consequently, we have the remaining part $-\mathbb{E}_{y \sim p_{ref}(.|x)} \big[\log p_\theta(y \mid x) \big]$ which is the cross entropy between the training model distribution and the reference model distribution. As is common practice in cross-entropy minimization, we can approximate this penalty by sampling from the reference distribution $p_{ref}(.)$ and optimize the training model using teacher forcing with a next-token prediction probability.

This shows that we can implement a constraint on the training model to be close to the reference model, aiming to mitigate forgetting, by approximating a KL constraint with samples from the reference model. Intuitively, given we sample from the reference model a sufficiently large set of data, by adjusting the model to mimic this dataset using SFT, we are forcing the training model to have an output distribution for those completions close enough with the reference model.

D PROMPTS USED FOR GENERATING ANSWER, FEEDBACK AND REVISION

In this section, we report the prompts we used for generating each of:

1. The student raw initial answer $\hat{y} \sim p_\theta(. \mid x)$;
2. The teacher feedback $f \sim p_{teacher}(. \mid x, y, \hat{y})$; and
3. The revised student answer $\tilde{y} \sim p_\theta(. \mid x, \hat{y}, f)$.

The prompts for each dataset are given below. We opted to use prompts that are dataset specific because there is too much variation between datasets on data format and specific instructions. For example, in the Science Q&A dataset, the tags are `<reasoning> ... </reasoning><answer> ... </answer>`, and the answer is composed by a single letter. In the Timely-Events dataset, the tags are `<explanation> ... </explanation><answer> ... </answer>` and the answer is free-form text. For Tool-Use, the data is composed by blocks of text, followed by `Action:` and `Action Inputs:` fields, that represent the tool call in a JSON format. Due to the heterogeneity on the data format, it would be very challenging (and even harmful) to use a general prompt that suits every dataset.

D.1 PROMPTS FOR TOOLUSE

Prompt: Student initial answer for Tool-Use

System prompt: `You are an expert agent for requesting the desired information from an API.`

User template prompt: `{input_prompt}`

Prompt: Teacher feedback for Tool-Use

System prompt: `You are a concise teacher helping to improve tool-use answers. Your job is to compare a student's answer against the task prompt and a correct demonstration. Give targeted feedback that helps the student revise their answer into a better one. Your feedback should summarize the difference between the student answer and the correct demonstration. Do not rewrite the full answer. Do not be verbose.`

User template prompt:

```
Task prompt:
{input_prompt}

Correct demonstration:
{demonstration}

Student answer:
{student_answer}

Give concise targeted feedback for improving the student answer.

Requirements:
- Mention only the most important issues.
- Prefer 1 to 3 concrete revision instructions.
- Focus on correctness: tool choice, action order, JSON inputs, and
missing required information.
- If the student answer is already good, say what is correct and give
only small improvements.
- If the student answer is correct and the "Action:" and "Action
Input:" fields match the ones in the demonstration, do not suggest
any modification.
- Output only the feedback text.
```

Prompt: Student revision answer for Tool-Use

System prompt: `You are revising your own tool-use answer. Edit the previous answer according to the teacher feedback. Do not write a completely new answer unless the feedback explicitly requires it.`

User template prompt:

```
Task prompt:
{input_prompt}

Previous student answer:
{student_answer}

Teacher feedback:
{teacher_feedback}

Revise the previous student answer.

Rules:
- Follow the teacher feedback strictly.
- Preserve correct parts of the previous answer.
- Make only changes that are supported by the feedback.
```

```
- Do not add extra explanation about the revision.
- Output only the revised answer.
```

D.2 Prompts for Science Q&A

Prompt: Student initial answer for Science Q&A

System prompt: You are a chemistry question-answering assistant. Your task is to answer multiple-choice chemistry questions accurately and in the exact required format. For every question, use your chemistry knowledge to determine the correct option. Output your response using exactly the following structure:

```
<reasoning>
REASONING CONTENT HERE
</reasoning>
<answer>
LETTER
</answer>

Formatting rules:
- The reasoning must appear only between "<reasoning>" and
"</reasoning>".
- The final answer must appear only between "<answer>" and "</answer>".
- The answer tag must contain only one uppercase letter: A, B, C, or
D.
- Do not include the option text or numerical value inside the answer
tag.
- Do not include any text before "<reasoning>" or after "</answer>".
- Do not use Markdown, bullet points, or code fences in the final
output.
- Always choose exactly one option.
```

User template prompt: `{input_prompt}`

Prompt: Teacher feedback for Science Q&A

System prompt: You are a concise teacher helping to improve Chemistry related answers. Your job is to compare a student's answer against the question and a correct solution. Give targeted feedback that helps the student revise their answer into a better one. Your feedback should summarize the difference between the student answer and the correct demonstration. Do not rewrite the full answer. Do not be verbose.

User template prompt:

```
Task prompt:
{input_prompt}

Correct solution:
{demonstration}

Student answer:
{student_answer}

Give concise targeted feedback for improving the student answer.

Requirements:
- Mention only the most important issues.
- Prefer 1 to 3 concrete revision instructions.
- Focus on correctness: the right molecule, compound, property,
reaction or any missing required information.
```

```
- If the student answer is already good, say what is correct and give
only small improvements.
- If the student answer is correct and the `<answer>...</answer>` field
match the ones in the solution, do not suggest any modification.
- Output only the feedback text.
```

Prompt: Student revision answer for Science Q&A

System prompt: `You are revising your own answer. Edit the previous answer according to the teacher feedback. Do not write a completely new answer unless the feedback explicitly requires it. Keep the revision concise and **close to the original answer**. Output only the revised answer.`

User template prompt:

```
Task prompt:
{input_prompt}

Previous student answer:
{student_answer}

Teacher feedback:
{teacher_feedback}

Revise the previous student answer.

Rules:
- Follow the teacher feedback strictly.
- Preserve correct parts of the previous answer.
- Make only changes that are supported by the feedback.
- Do not add extra explanation about the revision.
- Output only the revised answer.

Formatting rules:
- The reasoning must appear only between "<reasoning>" and
"</reasoning>".
- The final answer must appear only between "<answer>" and "</answer>".
- The answer tag must contain only one uppercase letter: A, B, C, or
D.
- Do not include the option text or numerical value inside the answer
tag.
- Do not include any text before "<reasoning>" or after "</answer>".
- Do not use Markdown, bullet points, or code fences in the final
output.
- Always choose exactly one option.
```

D.3 Prompts for Timely Events

Prompt: Student initial answer for Timely-Events

System prompt: `You are an expert in general factual knowledge. Answer the question accurately using only your own knowledge about the topic.`

```
Your entire output must follow exactly this format:

<explanation>
A brief explanation identifying the relevant facts needed to answer
the question.
</explanation>
<answer>
The direct answer to the question.
</answer>
```

```
Strict formatting rules:
- Output only the two XML-style blocks shown above.
- The first character of your response must be "<".
- The final characters of your response must be "</answer>".
- Include exactly one <explanation> block and exactly one <answer>
block.
- Do not wrap the response in quotes.
- Do not include Markdown, bullet points, code fences, headings, or
extra commentary.
- The explanation must appear only inside <explanation> and
</explanation>.
- The final answer must appear only inside <answer> and </answer>.
- The final answer must be concise and direct.
- The final answer must not include phrases such as "Based on the
explanation", "The answer is", or similar preambles.
- If the answer cannot be determined from the facts, write "Unknown"
inside <answer> and briefly explain why inside <explanation>.
```

User template prompt: `{input_prompt}`

Prompt: Teacher feedback for Timely-Events

System prompt: `You are a teacher helping the student to improve their answers about factual knowledge. Your job is to compare a student's answer and explanation against the original question and a correct explanation and answer. Give targeted feedback that helps the student revise their answer into a better one. Your feedback should summarize the difference between the student answer and the correct demonstration. Do not rewrite the full answer. Do not be verbose. If new facts need to be introduced in order to improve the answer, inform them to the student. If the student answer is correct but the student explanation contains information that is not consistent with the correct explanation, inform the student. The student does not have access to the correct textual information and explanation provided to you, so all the extra information the student needs to learn to answer correctly will be conveyed through your feedback. The information provided to you in the correct explanation is true and composed by actual events.`

User template prompt:

```
Original question:
{input_prompt}

Correct explanation:
{demonstration}

Student answer:
{student_answer}

Give concise targeted feedback for improving the student answer.

Requirements:
- Mention only the most important issues.
- Prefer 1 to 3 concrete revision instructions.
- Focus on correctness: important facts, logical deductions, and
missing required information.
- If the student explanation is correct and the content of the <answer>
... </answer> tags match the ones in the correct explanation, do not
suggest any modification.
- Output only the feedback text.
```

Prompt: Student revision answer for Timely-Events

System prompt: `You are revising your own answer and explanation to a question. Edit the previous answer according to the teacher feedback. Do not write a completely new answer unless the feedback explicitly requires it. Keep the revision concise and **close to the original answer**. Output only the revised answer.`

User template prompt:

```
Original question:
{input_prompt}

Previous student answer:
{student_answer}

Teacher feedback:
{teacher_feedback}

Revise the previous student answer.

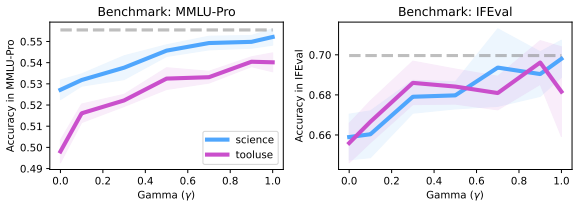
Rules:
- Follow the teacher feedback strictly.
- Preserve correct parts of the previous answer.
- Make only changes that are supported by the feedback.
- Output only the revised answer.

Formatting rules:
- The explanation must appear only between "<explanation>" and
"</explanation>".
- The final answer must appear only between "<answer>" and "</answer>".
- The final answer should contain only the answer, and should not have
phrases like "Based on the explanation ..."
- Do not include any text before "<explanation>" or after "</answer>".
- Do not use Markdown, bullet points, or code fences in the final
output.
```

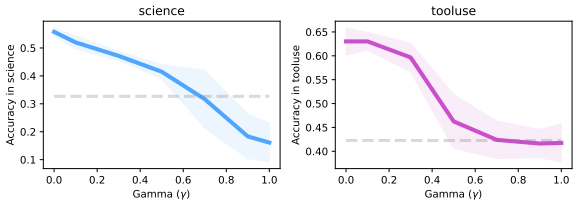
E TRADEOFF IN γ

The rate γ determines an adaptation/retention trade-off. If we set $\gamma = 0$ then all input prompts will have the original ground-truth demonstrations from the dataset as labels, which is standard SFT and represents the extrema of maximum adaptation to the new task and no retention to previous knowledge. However, if we set $\gamma = 1$, we train the model only on the reference samples and the model remains close to the reference model, however no information from the new target task is injected into the training model. In this case we have maximum knowledge retention but weakest new task adaptation.

By picking an intermediate value $0 < \gamma < 1$, we transition between the two extrema of maximum adaptation versus maximum knowledge retention. For the knowledge forgetting benchmarks in figure 3a, we can see that at $\gamma = 1$, the model trained with reference samples only is able to be very close to the reference model. However, as we decrease γ, we observe an increasing degree of forgetting, i.e. a clear decline of the accuracy on the forgetting benchmarks, which is caused by the usage of the ground-truth demonstrations in the dataset, that come from a distribution different than the reference model, and therefore force the training model to adapt sufficiently. The *in-domain performance* is also reported in figure 3b, and it is calculated as the accuracy on the test set of the target training dataset. When $\gamma = 0$, we only use the ground-truth samples as training data, and this results in the model that achieves the maximum accuracy on the target task. By increasing γ, the model is trained on larger fractions of reference model samples, and the accuracy drops for the target task, reaching the minimum achieved by the reference model (or even dropping below this value for the Science Q&A dataset).



(a)



(b)

Figure 3: Results for SELFTEACHSFT. In (a), the forgetting metrics on the IFEval and MMLU-Pro benchmarks is shown as we vary the value for γ. In (b), the *in-domain performance* on two training datasets is shown, i.e. the performance in the test set of the training task. In both (a) and (b), the gray dashed line represents the performance of the reference model. For the experiments used to create this figure, we used a `Qwen2.5-7B-Instruct` model.

F PSEUDO-CODE FOR LMATE

In this section, we present the pseudo-code for LMATE. Please see algorithm 1.

Algorithm 1 LMATE: learning from iteratively refined answers

Require: Adaptation dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$; student p_θ; frozen teacher $p_{teacher}$; number of answer-improvement rounds R; prefix edit limit k_{diff}
Ensure: Adapted student p'_θ

▷ Generate one initial answer from the unadapted student.
for all $(x_i, y_i) \in \mathcal{D}$ **do**
  Sample $a_i^{(0)} \sim p_\theta(\cdot \mid x_i)$
end for
▷ Keep the student parameters fixed during answer improvement.
for $r = 1, \ldots, R$ **do**
  ▷ The teacher uses privileged ground-truth information.
  for all $(x_i, y_i) \in \mathcal{D}$ **do**
    Sample feedback $f_i^{(r)} \sim p_{teacher}(\cdot \mid x_i, y_i, a_i^{(r-1)})$
    Sample a revised answer from student $a_i^{(r)} \sim p_\theta(\cdot \mid x_i, a_i^{(r-1)}, f_i^{(r)})$
  end for
end for
Construct the revised training dataset $\widetilde{\mathcal{D}} \leftarrow \{(x_i, a_i^{(R)})\}_{i=1}^{N}$
Define the LMATE training objective $\mathcal{L}_{\text{LMATE}}(\theta) \leftarrow -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(a_i^{(R)} \mid x_i)$
Update the student using the revised dataset $\widetilde{\mathcal{D}}$ $p'_\theta \leftarrow \text{TRAINSFT}(p_\theta, \mathcal{L}_{\text{LMATE}}, \widetilde{\mathcal{D}}, k_{\text{diff}})$
return p'_θ

G QUALITY OF THE TEACHER FEEDBACK

In this section, we study how effective the feedback of the teacher model is. We first sample the answers from the student, and then we perform three rounds of improvement by collecting the feedback from the teacher and revising the current answer. At every round, we measure the accuracy/F1-score of the answers over the dataset. The results can be seen in figure 4.

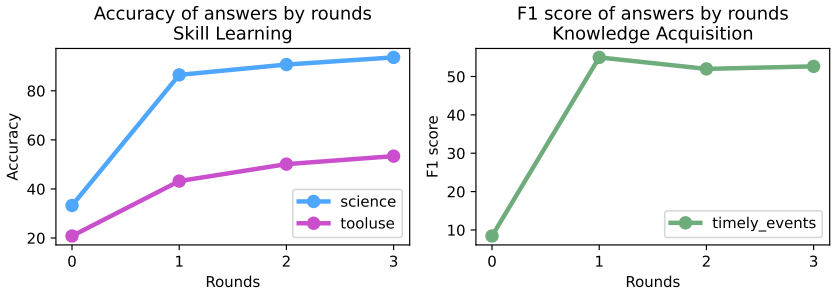


Figure 4: Measure of the teacher feedback effectiveness. At every round of answer improvement, we measure the accuracy of the current answers for the skill learning datasets, and the F1-score for the knowledge acquisition dataset. In this plot, "zero rounds" represents the initial answers sampled by the base model.

The effectiveness of the teacher feedback is considerably dependent on the dataset. For Science Q&A, the accuracy is $\approx 30\%$ for the initial answers generated from the student and it surpasses 93% after several improvement rounds, which shows that the teacher is effective in improving the student answers.

For the Tool-Use dataset, the accuracy barely surpasses 53% after a few rounds. In this dataset, the completions are composed by actions and action inputs, which represent calls to a function described in the input prompt. The accuracy is measured as the match between the actions output by the model compared with the ground-truth actions. We observe that some samples are composed by more challenging action sets, e.g. containing several actions in sequence, or by action inputs that should contain a specific ID that could not be guessed by the model. The teacher has difficulties in creating feedback for the student in these samples, and usually only after several rounds of improvement the student is able to make a meaningful correct modification to the answer. Investigating how the teacher feedback can become more effective for these cases is an interesting direction for future research.

For the Timely Events dataset, the F1-score for the base model is very low ($\approx 8\%$) but it reaches a value of almost 55% after one round of revision. We observe a small decay in F1 for the subsequent iterations. Even if the student latest answer is correct, the teacher might include on its feedback some factual information regarding the context of the news article of the question, in order to make the student answer more complete. By increasing this relevant context, the student answer has more information, but ends up having more text and a smaller precision value when compared with the ground truth answer, which addresses only the facts in the question. Although this phenomenon decreases F1 for training samples, we hypothesize this is beneficial for generalization, since the student is learning more context about those facts.

H HYPERPARAMETER SELECTION FOR k_{diff}

In this section, we present a comparison study for k_{diff} on Timely Events and Science Q&A. We split each dataset into 90% training set in which we train LMATE, then report the accuracy/F1-score on the remaining 10% validation set as the in-domain performance. As before, we also measure accuracy on IFEval and MMLU-Pro.

Results for Science Q&A can be seen in table 4. For this dataset, we observed that using the intermediate value of $k_{\text{diff}} = 64$ yields good results overall, having the best or the second best results in each metric. We choose $k_{\text{diff}} = 64$ for running LMATE in the skill learning datasets (both Science Q&A and Tool-Use) in the main text.

For Timely Events, we report the results in table 5. For this dataset, we observe that using a smaller value of $k_{\text{diff}} = 16$ yields better results for both target task F1 and forgetting. This corroborates with the discussion made in the main text (see section 6.2), in which we argue that preserving pre-trained abilities, in special instruction following, is crucial for good adaptation in this dataset. A small k_{diff} contributes to maintaining previous capabilities by limiting the update of the model to initial tokens in the answers, and yet achieve competitive adaptation scores.

Table 4: Diffmask-k ablation for LMATE on Science Q&A using `Qwen2.5-3B-Instruct` and learning rate 5×10^{-5}. Target-task performance is accuracy; retention is measured using IFEval `prompt-strict` accuracy and MMLU-Pro exact match.

Diffmask k	Target ↑	IFEval ↑	MMLU-Pro ↑
16	56.49 ± 4.16	$\underline{50.06} \pm 1.54$	34.43 ± 1.94
64	$\underline{69.93} \pm 2.46$	$\mathbf{51.50} \pm 2.04$	$\underline{35.90} \pm 0.70$
256	$\mathbf{70.30} \pm 1.51$	49.65 ± 1.51	$\mathbf{37.42} \pm 0.52$

Table 5: Diffmask-k ablation for LMATE on Timely Events using `Qwen2.5-3B-Instruct` and the learning rate of 2×10^{-5}. Target-task performance is F1; retention is measured using IFEval `prompt-strict` accuracy and MMLU-Pro exact match. Values are mean $\pm$ standard deviation, reported as percentages.

Diffmask k	F1 ↑	IFEval ↑	MMLU-Pro ↑
16	$\mathbf{21.64} \pm 1.87$	$\mathbf{58.67} \pm 1.20$	$\mathbf{42.09} \pm 0.29$
64	20.30 ± 0.91	57.56 ± 2.13	41.56 ± 0.34
256	$\underline{20.78} \pm 1.12$	57.56 ± 0.88	$\underline{41.68} \pm 0.23$

I HYPERPARAMETER VALUES USED IN THE EXPERIMENTS

We report the hyperparameters used for training in LMATE in table 6. The used learning rate depends on the type of task: we used 5×10^{-5} for skill-learning tasks (i.e. Science Q&A and Tool-Use), and 2×10^{-5} for Timely Events. We observed that a smaller learning rate works better for Timely Events, which as discussed above, benefits more from retaining instruction following abilities (see section 6.2). We ran all experiments in an NVIDIA B200 GPU.

For LMATE, we used three rounds of revision of the student answers. The values of k_{diff} used were $k_{\text{diff}} \in \{16, 64\}$. Please see Appendix H for more details.

We also report the hyperparameters used for the generation stages of LMATE in table 7.

Table 6: Training arguments used in the reported experiments. Set-valued entries indicate dataset-dependent choices; the value selected for each dataset is reported in the main text.

Training argument	**Value**
Maximum sequence length	2,048 tokens
Macro batch size	8
Learning rate	$\{2 \times 10^{-5}, 5 \times 10^{-5}\}$
Learning-rate scheduler	cosine with warmup
Warm-up ratio	5%
Epochs	2

Table 7: Generation arguments used during the three inference stages of LMATE. GPU memory utilization denotes the fraction of available accelerator memory allocated to `vLLM`.

Generation argument	**Initial response**	**Teacher feedback**	**Student revision**
`top_p`	1.0	0.9	0.9
`temperature`	1.0	0.4	0.2
`max_new_tokens`	1,024	256	1,024
GPU memory utilization	0.90	0.85	0.90