# Tandem Training for Language Models

**Robert West,[1]* Ashton Anderson,[2]* Ece Kamar,[3] Eric Horvitz[3]**
[1]EPFL, [2]University of Toronto, [3]Microsoft
robert.west@epfl.ch, ashton@cs.toronto.edu, eckamar@microsoft.com, horvitz@microsoft.com

## Abstract

As language models continue to rapidly improve, we can expect their actions and reasoning to become difficult or impossible for weaker agents and humans to follow, undermining interpretability and oversight. With an eye on long-term futures, we pursue methods that encourage models to produce solutions that remain intelligible to weaker collaborators. We formalize intelligibility as *handoff robustness:* a strong model's solution is intelligible to a weaker model if randomly handing off control to the weaker model along the solution path does not cause failure. Building on this criterion, we introduce *tandem training* for language models, a reinforcement learning (RL) paradigm in which rollout tokens are intermittently and randomly sampled from a frozen weak model rather than the strong model being trained. Because rollouts succeed only when the strong model's actions and reasoning process can be continued by the weak model—when the two can co-construct a successful solution—optimizing standard RL objectives with tandem training implicitly incentivizes both correctness and intelligibility. In the GSM8K math reasoning task, tandem training reliably teaches models to abandon jargon and adapt their language to weaker partners while keeping task accuracy high. Our results demonstrate a promising route to building AI systems that remain auditable by weaker agents, with implications for human–AI collaboration and multi-agent communication.

## 1 Introduction

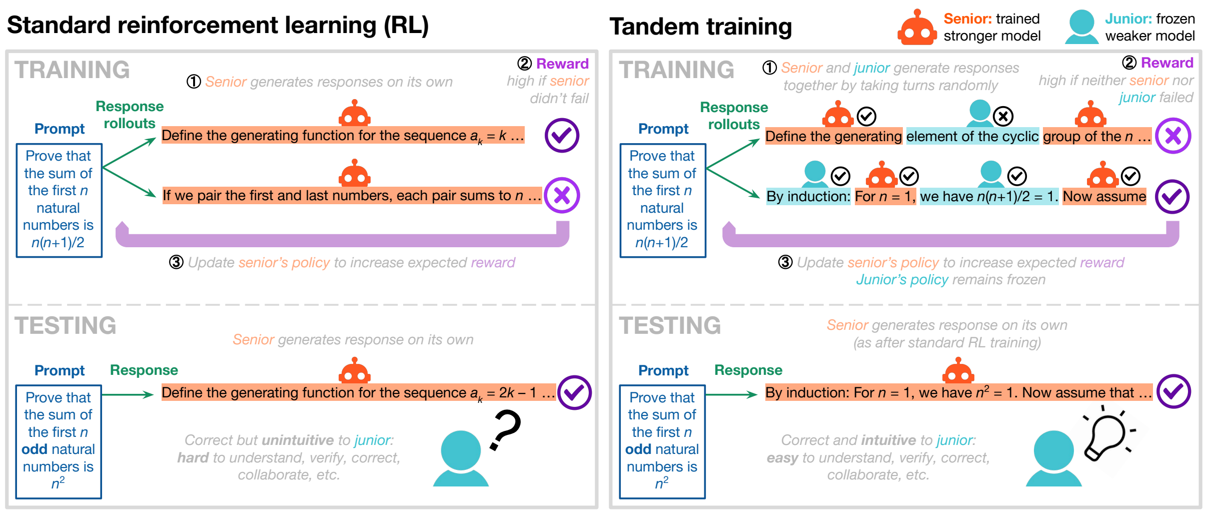
Artificial intelligence (AI) is rapidly becoming more capable. AI models have already matched or surpassed human capabilities in several milestone domains, and most researchers expect this trend to continue. As AI improves, however, its actions and reasoning will often become difficult or impossible for weaker agents and humans to follow because AI improvement is often driven by autonomous learning loops, synthetic data, and proxy rewards, causing model behavior to potentially stray from what humans or other agents might find familiar or intuitive.

This poses serious risks and challenges. Highly capable but unintelligible models may be useful in isolation or in highly modular environments. But when collaboration is useful or necessary, or when oversight is required, or in any scenario where interpretability and control are important, unintelligibility is a major problem. An agent with hard-to-follow reasoning, uninterpretable actions, or unpredictable decision-making increases the risk of detrimental outcomes in any system in which it plays a part. Without being intelligible, AI models and agents are limited in the extent to which they can collaborate with people, integrate into multi-agent systems, and be broadly beneficial.

With long-term futures in mind, we pursue methods that encourage models to produce high-quality solutions while remaining intelligible to weaker collaborators. We focus on language models, given their ubiquity and ability. One way to approach the problem of intelligibility would be to design elaborate system prompts, provide supervised examples, or construct custom rewards, but these bespoke solutions would be fraught with challenges. They would likely be brittle, easily gamed, and limited in scope.

We adopt another approach, where intelligibility is defined in a pragmatic, outcome-based manner: a (stronger) senior model $M_{\text{sen}}$ is considered intelligible to a (weaker) junior model $M_{\text{jun}}$ if $M_{\text{jun}}$ can take over during randomly assigned portions of a solution path without causing task failure. Imagine a model in the process of solving a problem, when randomly and unpredictably, it must hand off control to a weaker collaborator to continue solving the problem, and this handoff may

*Work done as visiting researchers at Microsoft Research.



Standard reinforcement learning (RL)
Tandem training
Senior: trained stronger model
Junior: frozen weaker model
TRAINING
① Senior generates responses on its own
② Reward high if senior didn't fail
Prompt
Response rollouts
Prove that the sum of the first n natural numbers is n(n+1)/2
Define the generating function for the sequence a_k = k …
If we pair the first and last numbers, each pair sums to n …
③ Update senior's policy to increase expected reward
① Senior and junior generate responses together by taking turns randomly
② Reward high if neither senior nor junior failed
Define the generating element of the cyclic group of the n …
By induction: For n = 1, we have n(n+1)/2 = 1. Now assume
③ Update senior's policy to increase expected reward
Junior's policy remains frozen
TESTING
Senior generates response on its own
Response
Prove that the sum of the first n odd natural numbers is n²
Define the generating function for the sequence a_k = 2k – 1 …
Correct but unintuitive to junior: hard to understand, verify, correct, collaborate, etc.
Senior generates response on its own (as after standard RL training)
By induction: For n = 1, we have n² = 1. Now assume that …
Correct and intuitive to junior: easy to understand, verify, correct, collaborate, etc.

Figure 1: **Tandem training vs. standard RL.** Standard RL ***(left)*** only encourages correct solutions, regardless of whether they are intelligible to other models or humans, which raises concerns about interpretability and oversight. We define intelligibility via *handoff robustness,* positing that a partial solution is intelligible to another agent if that agent could continue the solution, at least for a few steps, without derailing it into failure. *Tandem training* ***(right)*** encourages handoff robustness, and thus intelligibility, by letting the trained model ("senior") generate solutions together with a frozen (typically weaker) "junior" model, taking turns randomly. Since rewards are high only if neither senior nor junior made a critical mistake, the senior is encouraged to generate solutions that can be correctly continued (and thus, by our definition, understood) by the junior.

occur several times on the solution path. If $M_{\text{sen}}$ and $M_{\text{jun}}$ can jointly construct successful task solutions, then $M_{\text{jun}}$ must be able to understand $M_{\text{sen}}$'s behavior and actions well enough to continue successfully. We consider this a strong form of intelligibility, as the weaker collaborator is not simply interpreting the stronger model's actions *post hoc,* but actively co-constructs successful solutions.

Such *handoff robustness* is not straightforward to achieve. We tackle the challenge by introducing a novel paradigm for training language models, which we call *tandem training,* inspired by a proof of concept in chess (Hamade et al., 2024). Tandem training (cf. Fig. 1) can be combined with any reinforcement learning (RL) algorithm used for language model training. In RL, solution rollouts are sampled from the language model being trained, with the goal of training being to make successful rollouts more likely. Tandem training modifies the rollout phase by sometimes sampling next tokens from a frozen junior model $M_{\text{jun}}$ rather than the senior model $M_{\text{sen}}$ being trained. Whether the next turn's tokens are sampled from the junior or the senior model is determined randomly. As rollouts are more likely to be successful when $M_{\text{sen}}$ generates text that can be continued by $M_{\text{jun}}$ without introducing errors that lead to task failure, optimizing $M_{\text{sen}}$ via tandem training incentivizes not only correct, but also intelligible, solutions.

We conduct experiments in the domain of mathematical reasoning across three settings and find that tandem training reliably teaches strong models to abandon jargon and adapt their language to weaker models while keeping task accuracy high. In settings where senior and junior differ in their math skills, senior-specific notation drops from 99% to 0% within 20 gradient updates—indicating adaptation to the frozen junior model, which does not understand this notation—yet senior accuracy remains above the junior baseline. In settings where senior and junior use different languages before training, the senior adapts its language to the junior's language, again without senior accuracy falling below the junior baseline. These results demonstrate a practical path to aligning capable models with weaker collaborators, with the potential to improve AI model intelligibility and enable safer, more reliable, and more performant AI–human and AI–AI interaction.

## 2 Related work

**Tandem training.** Our work applies tandem training, an RL paradigm first introduced by Hamade et al. (2024) in chess, to language models. Their work showed that optimizing for partner compatibility (winning a team chess game) is distinct from optimizing for raw ability (winning a chess game on one's own). We adopt their framework and setup (e.g., "senior" and "junior" models; randomized handoffs between them) and significantly expand on this work by applying tandem training to language models, solving reasoning problems, and moving beyond the adversarial game environment.

**RL with verifiable rewards.** Recent work demonstrates that pure RL with automatically verifiable outcome rewards can produce strong reasoners (Wen et al., 2025), but with the undesirable pathology of degraded intelligibility (Li et al., 2025; Guo et al., 2025; Wu et al., 2025). Reasoning chains often become less reliable, grounded, and interpretable with length and complexity (Cheng et al., 2025; Hassid et al., 2025).

**Scalable oversight.** There has been a recent push to enable scalable oversight, where strong models can be overseen by weaker agents, e.g., via decomposition (Christiano et al., 2018) or debate (Irving et al., 2018). Complementary lines replace scarce human labels with AI feedback or process-level signals, showing that supervising intermediate reasoning can improve correctness and auditability at scale (Bai et al., 2022; Paul et al., 2023; Lightman et al., 2023). Research on weak-to-strong generalization (Burns et al., 2024) demonstrates that training with weak supervision can nevertheless surface strong capabilities, formalizing conditions under which a weaker teacher or signal suffices. Our tandem training approach operationalizes scalable oversight inside the solution trajectory: instead of relying on external judges, decompositions, or teacher labels, we impose random handoffs to a weaker collaborator during RL and reward success only when the strong model's reasoning is continuable by weaker agents, incentivizing intelligible solutions that are more amenable to oversight.

## 3 Method: tandem training

There are many domains of practical interest in which we would prefer language models to generate intelligible, intuitive solution paths, including, e.g., medicine (where human doctors may need to make final decisions based on AI-generated diagnoses or treatment plans), law (where human judges may need to rule in cases presented by AI lawyers), computer use (where human users may hand off control over their computers to AI agents and *vice versa*), or mathematical reasoning (where human mathematicians may collaborate with AI models to prove new theorems).

As a concrete example from the math domain, consider a human collaborating with a reasoning language model to "Prove that the sum of the first $n$ natural numbers is $n(n+1)/2$." A large and well-trained reasoning model might have deep knowledge of advanced mathematical techniques and might wield them precisely; e.g., it might begin its solution by writing: "Define the generating function for the sequence $a_k = k$: $A(x) = \sum_{k=1}^{\infty} kx^k$." Although this *ansatz* is in principle correct and might be entirely sensible from the model's point of view, a human might not be able to follow along, which would complicate collaboration with, and verification by, human partners. A more desirable solution might start by stating: "By induction: For $n = 1$, we have $n(n+1)/2 = 1$."

We ask: How might one incentivize language models to produce such intelligible solutions? Potential ways forward include test-time approaches, such as system prompts describing the nature of intelligible, intuitive solutions, and training-time approaches, such as supervised finetuning on ground-truth intelligible solutions or RL with intelligibility rewards. In practice, such approaches are, however, difficult to implement, as they require explicitly defining intelligibility *a priori*—a hard-to-codify notion that may differ across settings, agents/users, and times.

We seek more viable methods for encouraging intelligible outputs without the need to explicitly define intelligibility, relying instead on the notion of *handoff robustness,* which pragmatically and implicitly *defines a partial solution as intelligible to another agent (a model or human) if that agent could continue the solution—at least for a few steps—without derailing it into failure.* We operationalize this idea via *tandem training,* in which two models—called *senior* and *junior*—take turns randomly during output generation, without coordinating. The (typically weaker) junior model remains frozen, whereas the (typically stronger) senior model is trained[1] based on the quality of the output that the two models co-created. In rollouts that concluded successfully despite the junior's participation, the senior acted in a way that enabled the junior to not make critical mistakes (or else the rollout would have failed), which, per our

[1]Note the difference from model distillation (Hinton et al., 2015; Sanh et al., 2019), where, in a reversal of roles, the stronger model is frozen while the weaker model is trained.

definition, means that the senior acted in a way that was intelligible to the junior. Reinforcing the senior's behavior observed in successful rollouts thus achieves the dual objective of making the senior more intelligible to the junior and keeping the senior's performance high. The stochasticity of turn-taking not only provides a simple rule for when to switch between models, but also encourages handoff robustness and intelligible outputs in any situation, and prevents the senior model from acquiring tricks and reward hacks.

Tandem training can be viewed as a form of regularization: by injecting noise into the training process, it encourages simpler, more generalizable behavior. The approach is especially similar in spirit to dropout in neural-network training (Srivastava et al., 2014), where neurons of the network being trained are randomly "muted" so the network learns not to over-rely on specific activation patterns. Similarly, in tandem training, the senior model is randomly "muted" (and replaced by the "noisier" junior) so the senior learns not to over-rely on specific reasoning and speaking patterns. Akin to other regularization methods, including dropout, noise injection (here via handoffs to the junior model) is performed only during training; at test time, the tandem-trained senior model generates solutions on its own.

At a lower level, tandem training alternates between two phases: (1) generating tandem rollouts, and (2) updating the senior's policy based on them.

**Tandem rollout generation.** To generate tandem rollouts, we devised a decoding algorithm where two language models $M_{\text{sen}}$ and $M_{\text{jun}}$ work together to co-create an output. The granularity of stochastic turn-taking is a design parameter that determines the atomic *units* of text between which a handoff from one model to the other can occur, such as tokens, words, sentences, paragraphs, reasoning steps, etc. The tandem decoder keeps both models in GPU memory. Abstractly, the same input $x$ is fed to both models, but as the models may use different prompting modalities (e.g., language, system prompt, demonstrations, chat template), the concrete text sequences $x_{\text{sen}}$ and $x_{\text{jun}}$ seen by the two models may differ. To co-create a shared response $y$, whenever a new token $y_{t+1}$ is to be generated to continue the partial response $y_{1:t}$, each model $m \in \{\text{sen}, \text{jun}\}$ independently samples a token $y^m_{t+1} \sim M_m(x_m\, y_{1:t})$ given the shared context. Let $m_t$ be the currently active model (where $m_1$ is chosen randomly). If appending $y^{m_t}_{t+1}$ to $y_{1:t}$ would begin a new unit (e.g., word or sentence), we toss a coin (we use $p = 0.5$) to determine the new active model $m_{t+1}$; else, $m_{t+1} = m_t$ (since the current unit has not concluded yet). Last, we extend the shared partial solution by $m_{t+1}$'s proposal: $y_{1:t+1} = y_{1:t}\, y^{m_{t+1}}_{t+1}$.

**Senior policy update.** In order to update the senior model based on tandem rollouts, tandem training can leverage any RL method for language modeling, including REINFORCE (Williams, 1992), PPO (Schulman et al., 2017), GRPO (Shao et al., 2024), etc., which perform gradient descent to maximize the expected reward of rollouts, where rewards may be obtained from programmatic verifiers, trained reward models, humans, etc. We emphasize that tandem training does not require any explicit information about the differences between the senior and junior models, such as skill level, expected formatting, domain-specific jargon, etc.; updates are entirely based on the success of tandem rollouts. This is important as the differences between senior and junior might be subtle, difficult to express, or unknown.

## 4 Experimental setup

To explore and evaluate the paradigm of tandem training for language models, we conduct experiments in the domain of mathematical reasoning, which offers multiple benefits as a first testing ground for tandem training. First, researchers have shown that strong math reasoning models can be trained using RL without human supervision, relying solely on rewards obtained by automatically verifying answer correctness (Kaliszyk et al., 2018; Wang et al., 2025; Wen et al., 2025). In the absence of auxiliary rewards encouraging human-readable reasoning traces, such models have no incentive to reason in intelligible ways, which has already led to concerning outcomes, such as the "poor readability and language mixing" demonstrated by DeepSeek-R1-Zero (Guo et al., 2025). There is an immediate need to prevent reasoning models from becoming unintelligible to humans, and tandem training holds promise to directly address this pressing challenge.

In addition to its practical relevance, math reasoning has the advantage that the wide difficulty range of math problems allows us to begin by working with smaller models. Although tandem training is inspired by the challenge of making superhuman AI behave intelligibly to humans, we may begin with less capable model pairs. Instead of considering a superhuman AI as senior and a human as junior, we may use a stronger, but still subhuman, model as senior and a weaker model as junior—shifting the performance levels downward while maintaining a capability differential.

We work with the GSM8K (Grade School Math 8K) benchmark and construct tandem pairs from differently trained or prompted variants of the Llama-2 model family, as described next.[2]

### 4.1 Data

GSM8K consists of 8,792 math word problems (7,473 for training, 1,319 for testing) that typically require two to eight steps of elementary arithmetic operations to solve (Cobbe et al., 2021). The dataset consists of English question–answer pairs, e.g.,

Q: Julie is making Caesar salad for a family picnic. At the market, she spends \$8 on green lettuce and \$6 on red lettuce. If each type of lettuce costs \$2 per pound, how many total pounds of lettuce did she buy?

A: The total cost of the green and red lettuce is \$8 + \$6 = \$≪8+6=14≫14. Julie bought \$14 / \$2 = ≪14/2=7≫7 pounds of lettuce. #### 7

In the ground-truth answer, the final numerical solution is always given after "####". We also point out the special notation used for performing arithmetic: whenever an equality sign appears in the text (e.g., "\$14 / \$2 ="), it is followed by a "clean" version (e.g., without dollar signs) of the preceding calculation, enclosed in double angle brackets (≪≫). Presumably, this notation was introduced to facilitate parsing out arithmetic operations as "programs". In our setting, we may consider it domain-specific "jargon" that a generalist model would not use by default.

### 4.2 Models

We construct tandem pairs from various language models derived from Llama-2-7b:[3]

**Specialist model.**[4] Obtained from Llama-2-7b via supervised finetuning on the training portion of GSM8K, this model demonstrates increased performance on GSM8K (39% test accuracy, vs. 24% for vanilla Llama-2-7b-chat). It is prompted with the question only (no system prompt, no chat template) and, per its training data, uses ≪≫ jargon (cf. Sec. 4.1) in its answers.

**Base models.** Here we prompt the chat version of Llama-2-7b[5] in one of five languages $L$ (English, German, French, Bulgarian, Serbian), using a system prompt and two in-context question–answer demonstrations to explain the task and the expected output format and language (see Appendix A), followed by the GSM8K question. The entire prompt is provided in language $L$ (questions were translated ahead of time using GPT-4.1-mini), which effectively switches the model's default language from English to $L$. As the base models were not specifically tuned for GSM8K, they do not produce ≪≫ jargon. GSM8K test accuracy ranges between 12% (Serbian) and 24% (English).

### 4.3 Training and testing

We consider three settings, each of which combines the above models in a way that allows us to observe whether tandem training produces the desired effects (input languages in parentheses):

1. **Skill disparity:** $M_{\text{sen}}$ = specialist (English); $M_{\text{jun}}$ = base (English); jargon = $\{\ll\gg\}$
2. **Skill & language disparity:** $M_{\text{sen}}$ = specialist (English); $M_{\text{jun}}$ = base ($L_{\text{jun}} \neq$ English); jargon = $\{\ll\gg$, English$\}$
3. **Language disparity:** $M_{\text{sen}}$ = base ($L_{\text{sen}}$); $M_{\text{jun}}$ = base ($L_{\text{jun}} \neq L_{\text{sen}}$); jargon = $\{L_{\text{sen}}\}$

In the *skill disparity* setting, we should expect successful tandem training of the senior model to decrease usage of the GSM8K-specific ≪≫ jargon, while maintaining task accuracy well above that of the junior model. In the *skill and language disparity* setting, the senior additionally differs from the junior with respect to its input language, so in addition to ≪≫ jargon, we should expect successful tandem training to also make the senior respond in the junior's input language $L_{\text{jun}}$, rather than in English (the senior's input language). Finally, in the *language disparity* setting, neither the senior nor the junior was specifically tuned for GSM8K, so ≪≫ jargon plays no role here. Instead, the senior's input language $L_{\text{sen}}$ can be considered jargon, and we should expect successful tandem training to make the senior respond in the junior's, rather than its own, input language.

[2] All code, models, and data are made publicly available at https://github.com/epfl-dlab/lm-tandem-training

[3] https://hf.co/meta-llama/Llama-2-7b

[4] https://hf.co/RedHatAI/Llama-2-7b-gsm8k

[5] https://hf.co/meta-llama/Llama-2-7b-chat-hf

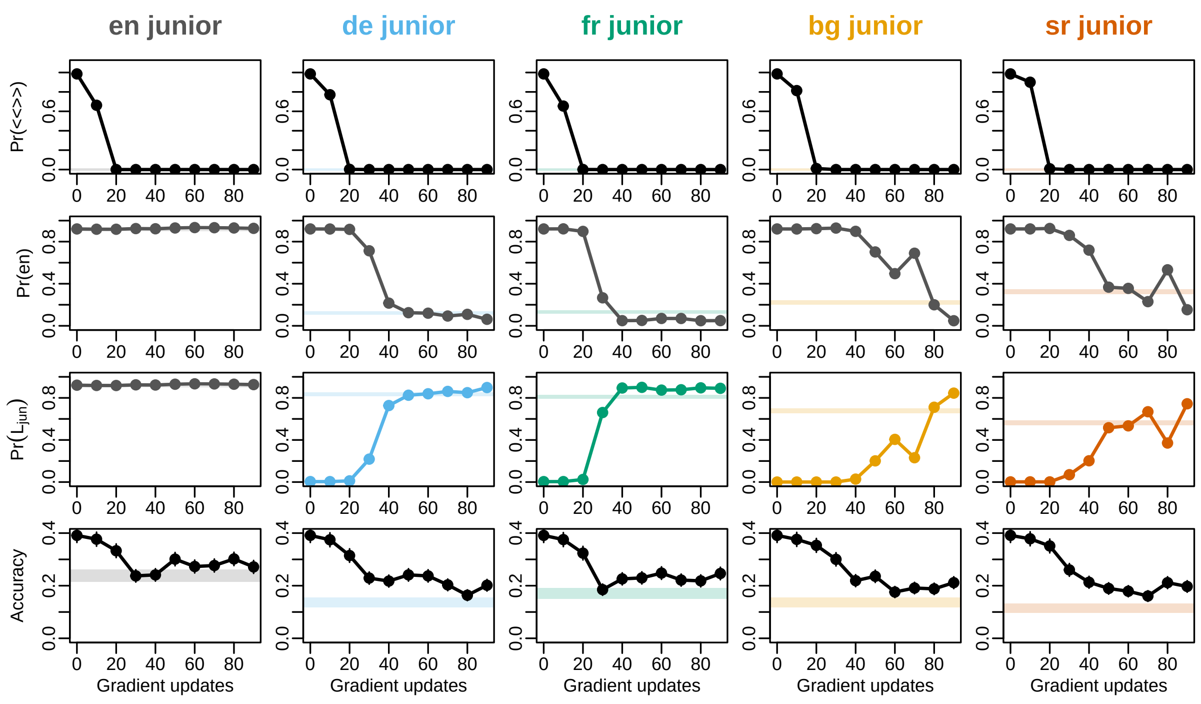


Figure 2: ***Column 1:*** Results for *skill disparity* setting, where tandem-trained senior is GSM8K specialist, and frozen junior is Llama-2 prompted in English. ***Columns 2–4:*** Results for *skill and language disparity* setting, where junior is Llama-2 prompted in non-English languages (one per column: German, French, Bulgarian, Serbian). ***Row 1:*** Use of notational jargon ($\ll\gg$). ***Row 2:*** Use of linguistic jargon (English). ***Row 3:*** Use of junior language. ***Row 4:*** Accuracy. ***Curves:*** Tandem-trained senior (with 95% CIs). ***Shaded bands:*** Frozen junior (95% CIs). ***Takeaway:*** Tandem training reduces senior model jargon while keeping senior accuracy above junior level.

In all settings, we tandem-train the senior for one epoch on the GSM8K training portion and evaluate it (on its own, without the junior) on the testing portion. Training is done using a variant of the REINFORCE (Williams, 1992) RL algorithm with binary rewards, where the following steps are iterated: (1) Given a batch of input questions, sample a set of (in our case, two) tandem rollouts for each question (cf. Sec. 3). (2) Perform an SGD update to increase the log-likelihood of the correct rollouts, while discarding incorrect rollouts.

When computing the log-likelihood of a rollout, one may also mask tokens produced by the junior, as they were not produced by the (senior) model being trained, but for simplicity we did not perform masking in the main experiments reported here. Turn-taking in tandem rollouts was carried out uniformly at random at the word level (cf. Sec. 3). The senior model was tuned using low-rank adapters on all linear layers (Hu et al., 2022). For hyperparameters, see Appendix B.

We emphasize that our goal is not to develop new RL algorithms, but to showcase the novel tandem training paradigm, which can, in principle, be used with any RL algorithm. Hence, it is advantageous if tandem training works even with a simple RL algorithm, such as the one presented here. (See Appendix E for an exploration of more complex variants, allowing for soft-masking of junior tokens and for not discarding incorrect rollouts.)

## 5 Results

Since the two-fold goal of tandem training is for the trained senior model to stop using jargon while keeping task accuracy high, we track accuracy as well as jargon over the course of training. We do so by storing checkpoints after every 10 gradient updates and using them to generate senior answers for the GSM8K test set. **Accuracy** is measured as the fraction of answers that provide the correct numerical solution. Jargon is measured in two ways: **notational jargon** is the fraction of answers that contain $\ll$ or $\gg$, whereas **linguistic jargon** is the probability of the senior's input language $L_{\text{sen}}$ in its generated output, according to a language identification method based on a fastText model (see Appendix C). Next, we discuss results for each of the three settings described in Sec. 4.3.

**Skill disparity.** The leftmost column of Fig. 2 shows results for the *skill disparity* setting, where, during tandem training, an (English) GSM8K-specialist senior is paired with a base-Llama-2 junior prompted in English. We observe that, whereas before tandem training the senior uses notational jargon ($\ll\gg$) in nearly every answer (99%), it entirely stops doing so (0%) within 20 gradient updates.[6] At this point, accuracy is 33%, slightly down from the 39% achieved by the senior before tandem training, and considerably higher than the 24% achieved by the frozen junior. Thereafter, accuracy first decreases and then remains stable and largely above junior level. We discuss and mitigate the accuracy decrease in the final paragraph of this section.

**Skill and language disparity.** The four right columns of Fig. 2 show results for the *skill and language disparity* setting, where tandem training pairs the (English) GSM8K-specialist junior with a base-Llama-2 junior prompted in a non-English language $L_{\text{jun}}$. Notational jargon ($\ll\gg$) vanishes as quickly as in the above-discussed skill disparity setting (within 20 gradient updates). Linguistic jargon also disappears as a consequence of tandem training: for three of the four junior languages $L_{\text{jun}}$, the senior has entirely abandoned English in favor of $L_{\text{jun}}$ within 50 gradient updates; for the fourth junior language (Bulgarian), within 80 updates. At the same time, senior accuracy remains significantly above junior level throughout training. Overall, tandem training again has the desired effect: to make senior jargon disappear while keeping accuracy above junior level.

**Language disparity.** Finally, we consider the *language disparity* setting, where both the senior and junior are base-Llama-2 models, but prompted in different languages $L_{\text{sen}}$ and $L_{\text{jun}}$, respectively. Here, the senior's input language $L_{\text{sen}}$ is considered jargon. As observed in Fig. 3, tandem training leads the senior to abandon its jargon fast and adopt the junior's language instead, typically within 20 gradient updates. In terms of accuracy (Fig. 4), we find that, when the junior beats the senior before training, the senior catches up; when they do equally well before training, it remains this way; and when the senior beats the junior before training, senior accuracy tends to decrease (see discussion in next paragraph), but never below junior accuracy. The tenor is again that tandem training makes jargon disappear without driving accuracy below junior level. Note in Fig. 3 that the switch from $L_{\text{sen}}$ to $L_{\text{jun}}$ is never direct, but always passes through an intermediate phase (around gradient update 10) where the senior outputs English—which is neither $L_{\text{sen}}$ nor $L_{\text{jun}}$. In other words, English appears as a transitory *lingua franca*. Inspection of tandem rollouts from training shows that, in early training, rollouts often begin with a mix of $L_{\text{sen}}$ and $L_{\text{jun}}$, which seems to confuse the models and lead them to switch to English amidst rollouts. As such English completions tend to be more successful, the trained senior adopts this behavior, but the frozen junior maintains its tendency to produce $L_{\text{jun}}$, especially early on in rollouts. This, in turn, lets the senior eventually abandon English in favor of $L_{\text{jun}}$.

**Mitigating accuracy decrease.** In the above results, senior accuracy tends to decrease towards junior accuracy over the course of tandem training. We hypothesize that the decrease is due to a gradual distribution shift: easier problems are more likely to be solved correctly, and since our simple RL method discards incorrect rollouts, the senior adapts to easier problems over the course of training. However, since the test set still follows the overall difficulty distribution, test accuracy suffers. This hypothesis is supported by a supplementary experiment where RL was performed with rollouts from the senior alone, rather than tandem rollouts (cf. footnote 6): here, too, test accuracy decreased with more training (from 40% to 34%), pointing to the RL method, rather than tandem rollouts, as the cause of accuracy deterioration. We thus experimented with two modifications to the RL method: (1) instead of discarding incorrect rollouts, include them in the log-likelihood objective with negative weight $c < 0$; (2) instead of weighing senior and junior tokens equally in the objective, "soft-mask" junior tokens via a multiplicative factor $j < 1$. Modification 1 enables learning from failure; modification 2 lets the senior focus more on its own than on the junior's behavior. (Note that the simpler RL method used in the main experiments above corresponds to $c = 0, j = 1$.) In Appendix E we show that, by optimizing the hy-

[6] To confirm that jargon indeed disappears due to tandem training, rather than simply due to RL, we isolated the effects of tandem training and RL in a supplementary experiment where we trained the senior using the same RL method, but on rollouts produced by the senior alone. In this setup, jargon remains at 99% for all checkpoints, confirming tandem training, rather than mere RL, as the cause for vanishing jargon.

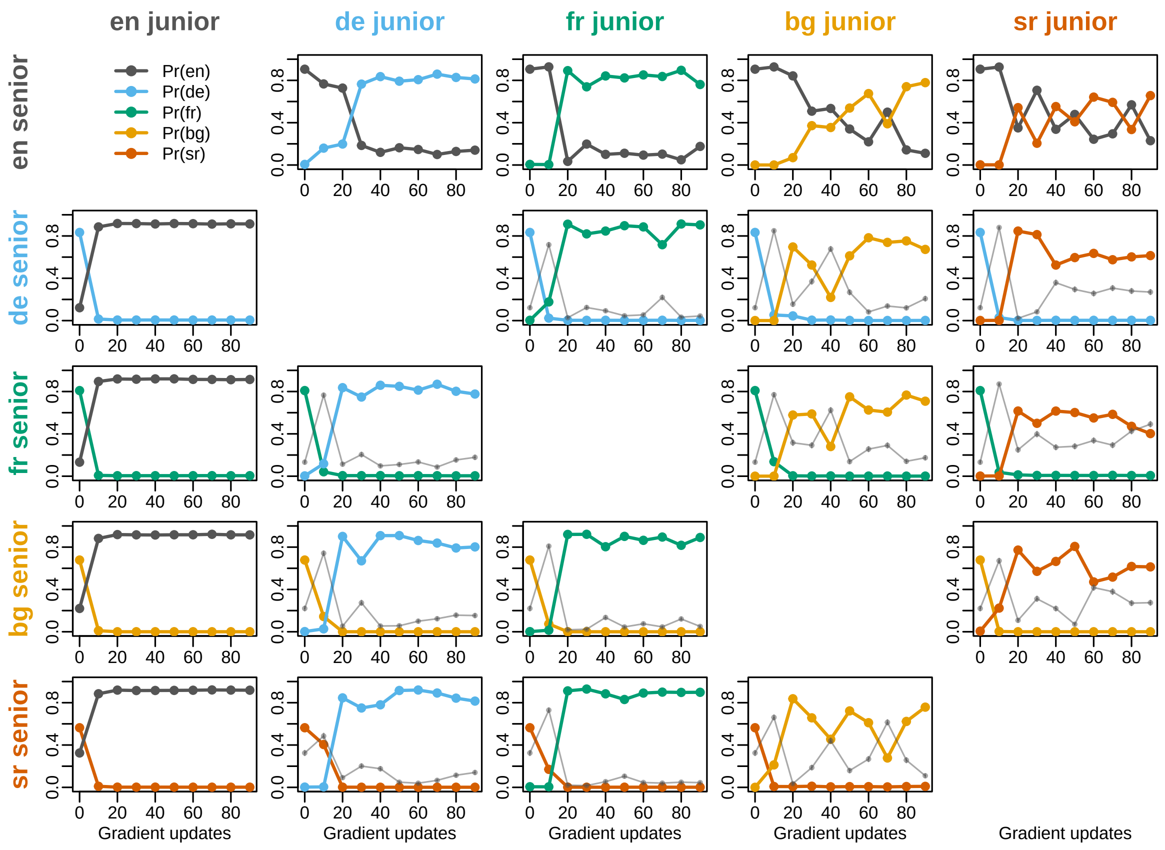


Figure 3: Results for *language disparity* setting, where tandem-trained senior ***(rows)*** and frozen junior ***(columns)*** are Llama-2 models prompted in different languages. Plots show senior's language use when applying checkpoints from tandem training to GSM8K test data. (Senior and junior accuracy shown in Fig. 4.) ***Takeaway:*** Tandem training reduces senior jargon (use of senior's input language) without driving accuracy below junior level.

perparameters $(c, j)$ on a validation set, the accuracy decrease is strongly mitigated, while jargon still vanishes entirely—the best of both worlds.

## 6 Discussion

**Result summary.** Across all three settings, tandem training rapidly suppresses "jargon", indicating the senior model learning to adapt to become compatible with the junior model, while maintaining above-junior accuracy. This is the core promise of tandem training: we can incentivize an AI model to adapt its behavior and actions to be compatible with a given weaker collaborator without an overly negative impact on its capabilities. Taken together, the results show that handoff robustness, our method of encouraging intelligibility, can be induced directly inside the RL trajectory by randomizing which partner generates next, creating pressure for the senior to produce continuable reasoning traces rather than idiosyncratic ones.

**Learning signals.** Our current sequence-level objective attributes rewards to entire rollouts, which is simple and effective but coarse. Three complementary refinements target sharper, lower-variance credit signals. First, masking junior-authored tokens during likelihood and policy-gradient computations can prevent spurious credit from leaking onto text that the trainable senior did not produce, aligning gradients with the senior's actual contributions. Second, learning from negative examples can leverage failed rollouts as informative counterfactuals: contrastive or penalty terms could downweight the specific trajectories and specific segments most predictive of handoff failure. Third, token- and span-level preference learning (Deng and Mineiro, 2024) can localize blame and credit to the precise tokens where handoffs break, rather than pooling signal uniformly across the trajectory. Our work shows that tandem training is effective even when combined with a simple RL method, and we expect the benefits to increase further with more sophisticated RL methods. This expectation

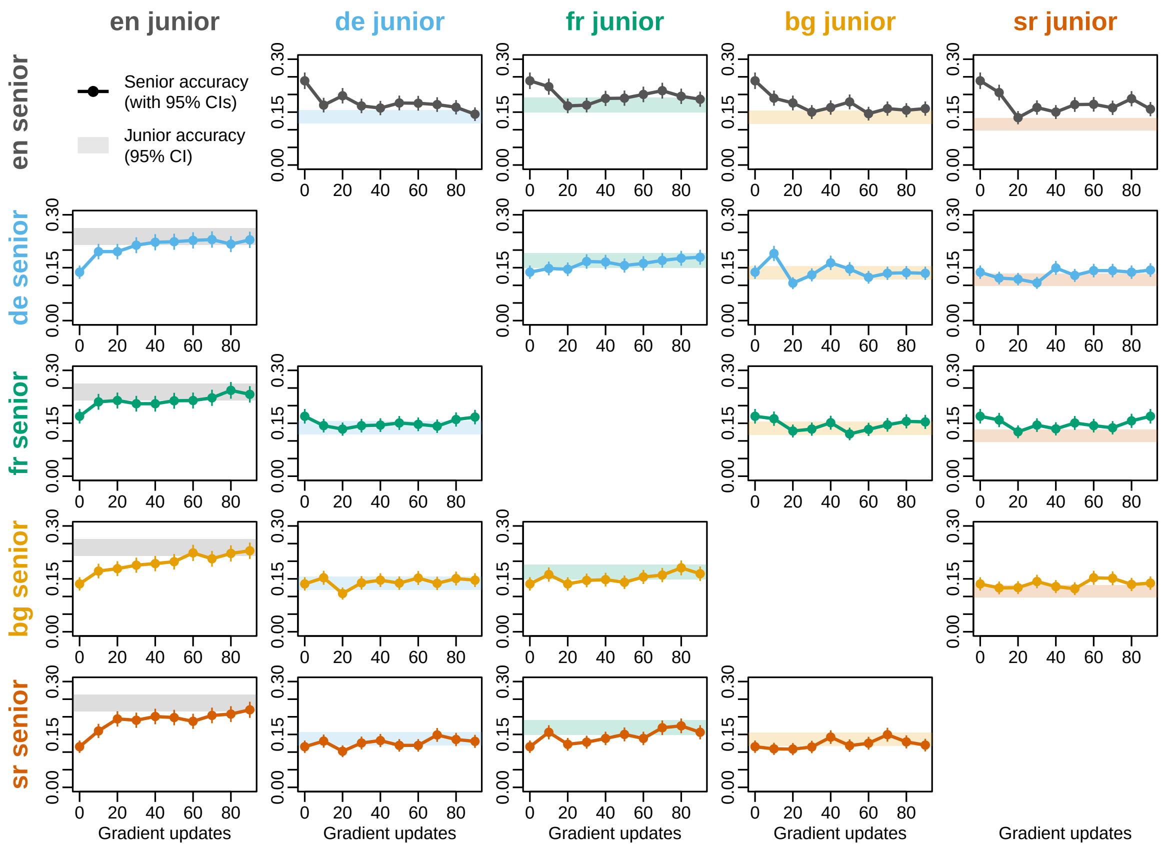


Figure 4: Senior and junior accuracy for *language disparity* setting, where tandem-trained senior ***(rows)*** and frozen junior ***(columns)*** are Llama-2 models prompted in different languages.

is supported by the initial experiments laid out at the end of Sec. 5 and in Appendix E, which show that the first two refinements listed above improve senior task accuracy.

**Extending the framework.** We present an atomic setting with a frozen junior and a trained senior, but there are many natural extensions of this basic tandem training framework. Co-adaptation would be possible by training both models, either simultaneously or by alternating which is trained and which is frozen, perhaps with additional anchoring constraints to prevent private codes from emerging. The "junior model" role can be expanded to design and control the type of compatibility we want to produce in the senior model; one could imagine varying junior competence, style, language, tool use, etc., to regularize the senior toward broadly intelligible behavior. An automated framework could maintain a pool of juniors that are swapped in via a bandit algorithm or curriculum policy where juniors that expose failure modes are prioritized and juniors that the senior has already mastered are gradually retired. It is also possible to apply tandem training earlier in the stack (e.g., late pretraining or supervised finetuning) to encourage intelligibility as a more fundamental attribute. Finally, we limited our attention to fixed i.i.d. handoffs, which could likely be improved with a handoff schedule that optimizes robustness.

## 7 Conclusion

This work introduces tandem training for language models, a novel RL paradigm that operationalizes and encourages intelligibility via handoff robustness. The approach is lightweight, architecture-agnostic, and complementary to existing RL pipelines. Our results show that by making intelligibility a prerequisite for reward, tandem training aligns AI models toward behavior that partners can pick up, audit, and extend, with benefits for scalable oversight and practical collaboration. We see great promise in harnessing tandem training in multi-agent systems and human–AI collaboration.

## 8 Limitations

Our principal goal is to investigate the viability of tandem training for encouraging language models to produce more intuitive and intelligible output. The settings and methods we consider in this first paper are hence limited in scope:

- We consider a single application domain (mathematical reasoning) and evaluate on a single benchmark (GSM8K). An important avenue for future work is to investigate whether the paradigm works equally well in other domains, and what makes a domain more or less amenable to tandem training.
- Although tandem training can in principle be combined with any RL algorithm, we tested it only with REINFORCE with binary rewards. While future work should improve the RL method (cf. discussion in Sec. 6), we consider it an advantage that tandem training works even with simple RL, and we expect it to be even more effective when combined with more advanced RL.
- The tandem decoder implementation described in Sec. 3 assumes that both models (junior and senior) use the same tokenizer. An extension to different tokenizers would be straightforward.

## 9 Ethical considerations

This work investigates tandem training as a way to operationalize intelligibility by requiring that a strong model's partial solutions be continuable by a weaker collaborator under randomized handoffs during RL. Our experiments use public math word-problem data (GSM8K) and off-the-shelf Llama-2-7b variants; no human subjects or personal data are involved.

**Potential benefits and misuse.** By design, tandem training increases handoff robustness and reduces jargon, which can strengthen auditability and scalable oversight by weaker agents. However, improved continuability could be misapplied to coordinate more effectively with colluding AIs or to make reasoning traces more persuasive in undesirable contexts. To mitigate this, we recommend anchoring mechanisms, e.g., human-language constraints, verifiers, and audits, especially when both partners are trained, to deter private codes and preserve human understandability.

## 10 Reproducibility statement

Experiments were run on machines with one or two Nvidia A100 GPUs with 80 GB of memory each. Data, models, training and testing procedure, and evaluation methodology are described in the main text. Hyperparameters, prompts, and further method details are listed as appendices. Model artifacts used to instantiate seniors and juniors are referenced in the main text. We release all code, models, and data required to reproduce our results.

## Acknowledgments

This work was performed when Robert West and Ashton Anderson spent their sabbaticals as Visiting Researchers with the AI Frontiers group at Microsoft Research. Thanks to all our Microsoft colleagues for their thoughtful feedback on this work (in particular Saleema Amershi, Siddhartha Sen, and Tim Davidson) and for their warm welcome in Redmond and New York! We also gratefully acknowledge the help of Raghav Singhal and Difan Jiao, who ran additional experiments during the review phase (Appendix E).

## References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Collin Burns, Amanda Askell, Long Chen, and 1 others. 2024. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR.

Jiahao Cheng, Tiancheng Su, Jia Yuan, Guoxiu He, Jiawei Liu, Xinqi Tao, Jingwen Xie, and Huaxia Li. 2025. Chain-of-thought prompting obscures hallucination cues in large language models: An empirical evaluation. *arXiv preprint arXiv:2506.17088*.

Paul Christiano, Buck Shlegeris, and Dario Amodei. 2018. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yihe Deng and Paul Mineiro. 2024. Flow-dpo: Improving llm mathematical reasoning through online multi-agent learning. *arXiv preprint arXiv:2410.22304*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Karim Hamade, Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. 2024. Designing skill-compatible ai: Methodologies and frameworks in chess. *arXiv preprint arXiv:2405.05066*.

Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. 2025. Don't overthink it. preferring shorter thinking chains for improved llm reasoning. *arXiv preprint arXiv:2505.17813*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Geoffrey Irving, Paul Christiano, and Dario Amodei. 2018. Ai safety via debate. *arXiv preprint arXiv:1805.00899*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016a. FastText.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016b. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olšák. 2018. Reinforcement learning of theorem proving. *Advances in Neural Information Processing Systems*, 31.

Yihao Li, Jiayi Xin, Miranda Muqing Miao, Qi Long, and Lyle Ungar. 2025. The impact of language mixing on bilingual llm reasoning. *arXiv preprint arXiv:2507.15849*.

Hadar Lightman and 1 others. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. REFINER: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Lucas Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. 2025. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*.

Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

## A Llama-2 prompts

Llama-2-7b-chat was prompted for GSM8K with a system prompt, followed by two question–answer demonstrations, followed by the input question.

### A.1 System prompt

*You are a math reasoning assistant specialized in solving grade school math word problems. Your task is to provide a clear, step-by-step explanation of the solution and, importantly, to output the final numerical answer on a separate line in exactly the following format:*

*"#### <final answer>"*
*where <final answer> is the numerical result (without any units or extra symbols). Do not include any additional text or formatting on that final line, just a single number.*

When the model was used in a non-English language $L$, the above system prompt was extended by the following instruction and translated into $L$:

*Respond exclusively in L. Never use English or any other language. If you use the English language or any other language, the answer is invalid. Every part of the response must be entirely L.*

### A.2 Demonstrations

Showing demonstrations serves to exemplify the expected output format and language described in the system prompt. The following two question–answer pairs wer used as demonstrations:

[USER] *Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?*

[ASSISTANT] *Natalia sold 48/2 = 24 clips in May. Natalia sold 48+24 = 72 clips altogether in April and May. #### 72*

[USER] *Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?*

[ASSISTANT] *In the beginning, Betty has only 100 / 2 = $50. Betty's grandparents gave her 15 * 2 = $30. This means, Betty needs 100 - 50 - 30 - 15 = $5 more. #### 5*

When the model was used in a non-English language $L$, the demonstrations were also translated to $L$, with the following exception: during development, we found that Llama-2 is less likely to fall back to English (its majority training language), especially during mixed-language tandem rollouts, when one demonstration has a question in English and an answer in $L$, emphasizing that the model should always revert to $L$. We hence always keep the first demonstration's question (but not its response) in English.

## B Hyperparameters

| | |
|---|---|
| Effective batch size | 152 or 156 |
| Learning rate | $10^{-4}$ |
| Training temperature | 0.7 |
| Testing temperature | 0 |
| Max output length | 256 |
| Max input + output length | 512 |
| Training rollouts per prompt | 2 |
| Torch dtype | `bfloat16` |
| QLoRA base model quantization | 4 bits |
| QLoRA rank $r$ | 16 |
| QLoRA scaling factor $\alpha$ | 16 |
| QLoRA target modules | All linear |

With rollout diversity in mind, when performing two rollouts per prompt, the second rollout's coin-toss sequence was an inverted version (i.e., with heads and tails swapped) of the first rollout's coin-toss sequence.

"All linear" QLoRA target models: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`.

## C Language identification

In order to infer the languages present in the model outputs, we rely on the fastText (Joulin et al., 2016a,b) `lid.176.bin` language identification model.[7] Since this model was not trained for mixed-language text (as often produced during tandem training), we do not apply the model directly to the entire text, but use a sliding-window approach instead, as short windows generally contain less language mixing. First, the input text is cleaned of digits and symbols (#, =, +, -, *, <, >, $\times$, $\div$, $) and split into tokens. It is then divided into overlapping windows ("shingles") of eight tokens each. Each shingle is passed to the fastText language identification model, which returns a probability distribution over possible languages. Finally, these per-shingle distributions are averaged across all shingles to produce a global language distribution for the text.

[7] `https://fasttext.cc/docs/en/language-identification.html`

## D AI assistants statement

We used AI assistants like Copilot and GPT models to help us with writing the tandem training code and to assist with writing.

## E Supplemental results

While the main experiments of Sec. 5 used a variant of REINFORCE with binary rewards (see Sec. 4.3), we also explored the following generalization:

1. Instead of discarding incorrect rollouts, include them in the loss with negative weight $c < 0$ (whereas correct rollouts are included with positive weight 1).

2. Instead of weighing senior and junior tokens equally in the loss, "soft-mask" junior tokens via a multiplicative factor $j < 1$.

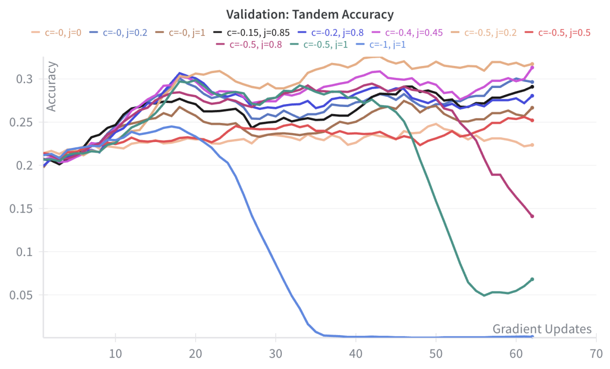
Modification 1 lets the senior learn from failure, whereas modification 2 lets the senior focus more on its own than on the junior's behavior. The simpler RL method used in the main experiments corresponds to $c = 0, j = 1$.

We validated a range of $(c, j)$ combinations in order to investigate whether a judicious choice of these hyperparameters can mitigate or prevent the task-accuracy deterioration observed in Sec. 5 while still teaching the senior to avoid jargon. Importantly, we should not use any predefined notion of jargon in order to choose hyperparameters. One key reason for choosing GSM8K as a domain was precisely that it offers an exactly measurable notion of jargon ($\ll\gg$), which facilitates studying the effects of tandem training; but in real-world settings we generally do not know in what aspects—including jargon—senior and junior might differ.

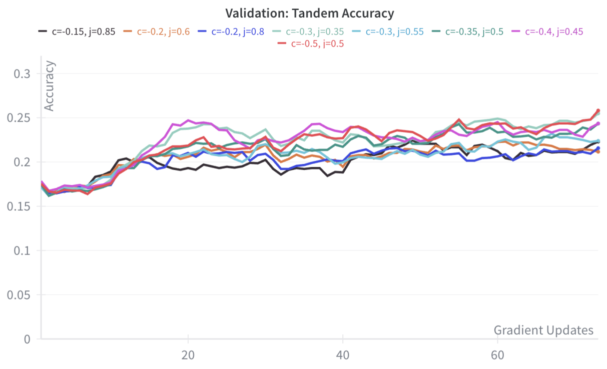
We hence chose $(c, j)$ without referring to jargon and measured jargon only after the fact, to determine whether the corresponding senior model with the chosen hyperparameters meets the criteria of reducing jargon while maintaining high GSM8K task accuracy. Concretely, for each $(c, j)$, we tandem-trained a senior model for one epoch on a random sample containing 90% of the GSM8K training portion and validated the GSM8K task accuracy of the tandem (consisting of the trained senior and the frozen junior) on the remaining 10%. After training, we chose the $(c, j)$ for which the tandem had maximum validation accuracy on average over the last few training checkpoints. Here, the rationale is that, in order for a tandem—where senior and junior take random turns during rollouts—to have high GSM8K accuracy, the senior must have learned to adapt to the junior, which we take as a heuristic for choosing a suitable senior.

In these preliminary experiments, we restricted ourselves to the *skill disparity* setting and to the *skill and language disparity* setting with a French junior. The training runs used the same hyperparameter values listed in Appendix B, with the exception of effective batch size, which, per GPU capacity, was set to 160 and 152, respectively. The tandem model's validation accuracies are shown in Fig. 5a and 5b, respectively, leading us to choose $(c = -0.5, j = 0.2)$ for the skill disparity setting and $(c = -0.3, j = 0.35)$ for the skill and language disparity setting. Evaluating the chosen seniors (on their own, not in a tandem) on the test portion of GSM8K (Fig. 5c) shows that tandem training makes them stop using jargon ($\ll\gg$, and additionally English in the skill and language disparity setting) while keeping task accuracy considerably higher than in the main experiments: whereas in the main experiments, task accuracy in the skill disparity setting noticeably dropped from its initial 39% and stabilized around 30% (Fig. 2, first column), here it remained at around 40% (Fig. 5c, top row); and whereas in the skill and language disparity setting with a French junior (Fig. 2, third column) it stabilized between 22% and 24%, here it remained above 30% (Fig. 5c, bottom row).

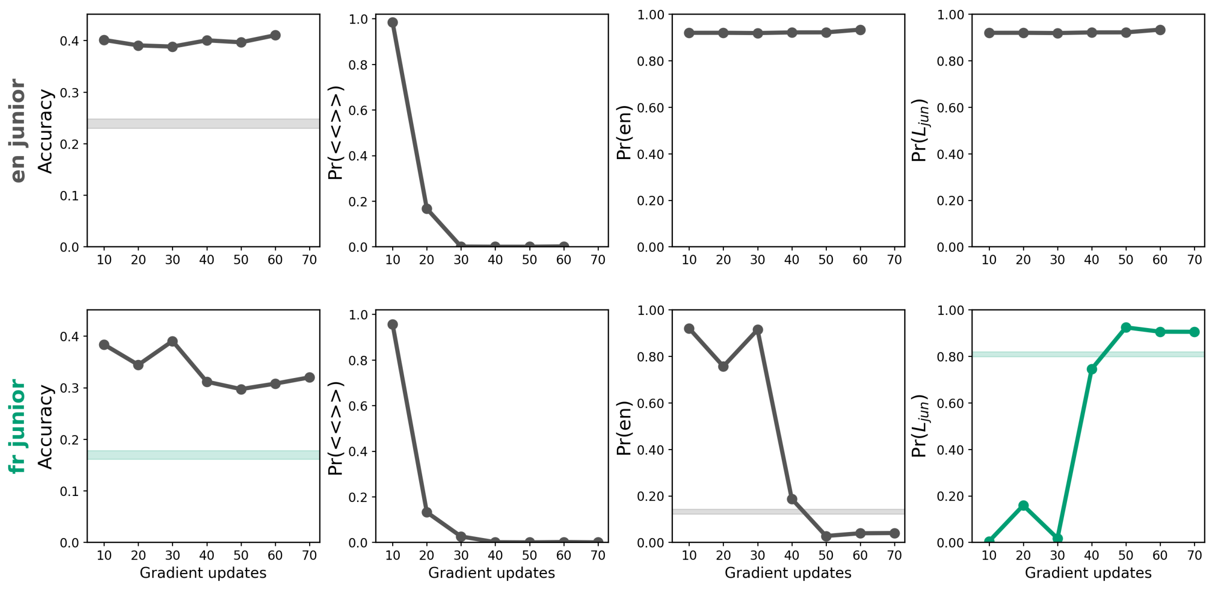
Taken together, these supplemental results show that a slightly more sophisticated variant of the RL method of Sec. 4.3 more closely achieves the key promise of tandem training: to make the senior more intelligible without sacrificing performance. We thus see exploring even more advanced RL algorithms as a promising direction for future work.



(a) Validation accuracy: *skill disparity* setting.



(b) Validation accuracy: *skill and language disparity* setting.



(c) Test results for *skill disparity* setting ***(top)*** and *skill and language disparity* setting ***(bottom)***.

Figure 5: Results for tuning the hyperparameters $(c, j)$ (defined in Appendix E) of the RL algorithm. **(a)** Validation accuracy of the tandem model for a range of hyperparameter choices, in the skill disparity setting. **(b)** *Idem,* in the skill and language disparity setting. **(c)** Test results achieved by the senior models selected based on validation accuracy. In comparison to Fig. 2 (columns 1 and 3), GSM8K accuracy remains higher, while jargon ($\ll\gg$ in the case of skill disparity, $\ll\gg$ and English in the case of skill and language disparity) still disappears.