Self-training Language Models for Arithmetic Reasoning

Marek Kadlčík* and **Michal Štefánik***
Faculty of Informatics, Masaryk University, Czech Republic
{kadlcik,stefanik.m}@mail.muni.cz

Abstract

Recent language models achieve impressive results in tasks involving complex multistep reasoning, but scaling these capabilities further traditionally requires expensive collection of more annotated data. In this work, we explore the potential of improving models' reasoning capabilities without new data, merely using automated feedback to the validity of their predictions in arithmetic reasoning (*self-training*).

In systematic experimentation across six different arithmetic reasoning datasets, we find that models can substantially improve in both single-round (offline) and online self-training, reaching a correct result in +13.9% and +25.9% more cases, respectively, underlining the importance of *actuality* of self-training feedback. We further find that in the single-round, *offline* self-training, traditional supervised training can deliver gains comparable to preference optimization, but in *online* self-training, preference optimization methods largely outperform supervised training thanks to their superior stability and robustness on unseen *types* of problems.

1 Introduction

Despite recent improvements in the practical usability of language models (LMs) attributed to preference alignment methods (Wang et al., 2023), these models often struggle with tasks requiring *reasoning*, i.e., a process of inferring a conclusion or decision logically and systematically (Huang and Chang, 2023). Previous work improves the reasoning capabilities of language models by *scaling* training data to more diverse (Kadlčík et al., 2023) or complex (Hendrycks et al., 2021) collections, but reaching further improvements in this direction becomes exceedingly expensive.

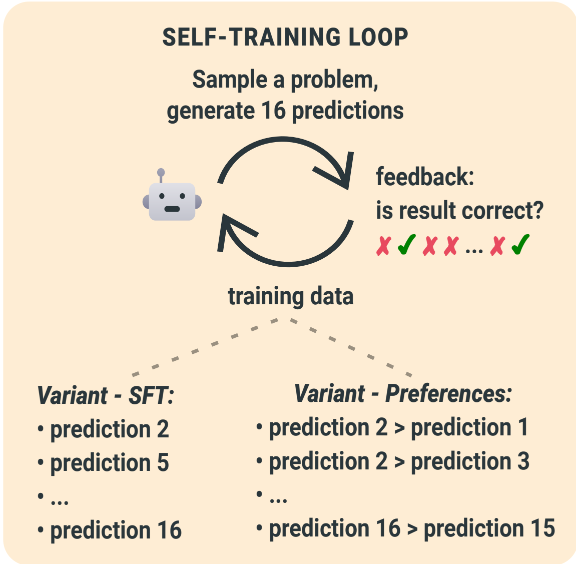

Figure 1: Schema of self-training that we apply to provide the model with training feedback to its predictions. In the offline variant, the model generates all predictions in a single round. In the online variant, the training data is continuously generated.

In this work, we evaluate the potential of improving models' capabilities by training from implicit, automated feedback to models' responses. *Arithmetic reasoning* tasks present a challenge that reflects heavily on the model's reasoning capabilities, while the quality of the model's responses can be automatically assessed against the annotated *correct results* rather than expensive and possibly subjective judgments of model outputs (Hu et al., 2023). Thus, we choose the arithmetic reasoning to address our two main research questions:

RQ1: To what extent can we improve the reasoning abilities of language models without additional data?

RQ2: Can the preference optimization bring further improvements to models' capabilities over traditional supervised fine-tuning?

*Equal contribution; Authors ordered alphabetically

Findings of the Association for Computational Linguistics: EACL 2024, pages 12378–12386
November 12-16, 2024 ©2024 Association for Computational Linguistics

We address these questions by implementing two variants of self-training: (1) an *offline* variant, where the training feedback to the model responses is constructed in a *single* iteration (§3.1), and (2) an *online* variant, where the model obtains and trains on the feedback to its *current* predictions (§3.2).

Our experiments reveal that both self-training variants present an efficient method for improving LMs' capabilities with *implicit* training signal; both variants allow to significantly improve the initial model without *any* new data. In the offline variant, similar improvements can be achieved by both supervised and preference optimization methods. However, the online variant reveals crucial issues in scaling the supervised training to *autonomous* settings. On the contrary, preference optimization methods can robustly persist the original capabilities even in autonomous self-training while reaching further improvements.

Finally, the difference in average improvement between our best-performing offline (+13.9%) and online method (+25.9%) indicates that the *actuality* of self-training feedback is a crucial factor of self-training effectivity. Our results motivate future research towards exploring new sources of implicit feedback able to provide language models with immediate feedback to their current predictions.

2 Related Work

We build upon a line of previous work that experiments with providing feedback to language models in arithmetical reasoning. Notably, Luo et al. (2023) train models with PPO (Schulman et al., 2017) against feedback on individual steps given by ChatGPT 3.5. Uesato et al. (2022) apply variants of self-training on GMS8K and compare the effectiveness of giving outcome-based (per solution) or process-based (per each step in solution) feedback, concluding that the two approaches result in comparable accuracy, but outcome-based feedback delivers a higher error rate in the rationales. Lightman et al. (2023) also focus on a comparison of process-based and outcome-based feedback on a larger scale and conclude that process-based feedback outperforms outcome-based at end-result accuracy.

Our work is closest to Parisi et al. (2022) and Zelikman et al. (2022). Parisi et al. (2022) apply self-training with a traditional supervised objective: they train the model on a small set of seed data and continuously use the trained model to generate solutions for a larger set, from which correct solutions are used in another training epoch. They show that three such subsequent epochs can improve the accuracy with diminishing returns. Zelikman et al. (2022) experiment with self-training with supervised fine-tuning on commonsense and math reasoning. They report positive results of self-training on the model's reasoning capabilities under specific conditions: (1) the initial model must be capable enough to be able to achieve improvements, and (2) training tasks must hold a negligible chance of random success (unlike, e.g., binary classification).

Our work builds upon these findings but differs from previous work in our objectives and data setting; We provide a systematic comparison of *different* training objectives in *both* online and offline settings, including the most recent preference optimization methods and show that training objective indeed plays a crucial role, especially in the online setting. Our data setting is more ambitious than of previous work: we show that self-training can deliver substantial improvements also by using *only* problems already *seen* in previous training. Finally, contrary to previous self-training work, we make our code[1] and models[2] freely available to accelerate future work in self-training.

3 Experiments

Our experiments build upon the 3-billion-parameter FLAN models fine-tuned specifically for arithmetic reasoning in previous work of Kadlčík et al. (2023). These relatively compact calculator-assisted models called CALCFORMERS were shown to perform noticeably well on multi-step reasoning, while even on single-step and two-step problems perform compared to 70B Llama-2 (Touvron et al., 2023). Another desiderata of these models is the *transparency* of their training data. In our experiments, this allows us to opt for a more challenging yet realistic self-training setting where we do *not* train the models on *any new* data, but only on the problems that CALCFORMERS have *already* seen in the training, merely with a complementary training signal.

Specifically, we self-train these models with the prompts from Ape210K (Zhao et al., 2020), to our knowledge the largest available arithmetical reasoning dataset of over 200,000 math problems. In addi-

[1] `https://github.com/prompteus/calc-x`
[2] see our HuggingFace Hub

	GSM8K	AQuA-RAT	Ape210K	MAWPS	SVAMP	ASDiv-A
BASE MODEL	43.2±2.7	37.8±6.1	26.3±2.1	61.9±4.2	51.8±3.2	78.7±2.3
SFT PLAIN	**46.1**±2.7	37.8±5.9	32.9±2.2	70.6±3.8	56.2±3.0	81.9±2.2
SFT PLAIN + LoRA	44.9±2.7	**39.0**±5.9	**37.3**±2.2	**80.8**±3.5	55.8±3.1	**82.8**±2.1
SFT BALANCED	45.8±2.7	37.4±5.9	33.6±2.2	66.7±3.9	**58.4**±3.0	82.0±2.2
SFT WITH NEGATIVES	41.8±2.7	33.1±5.7	28.0±2.1	65.2±4.1	52.2±3.1	75.9±2.4
DPO ($\beta = 0.99$)	45.3±2.7	37.0±5.9	29.2±2.1	69.6±3.9	54.2±3.1	83.1±2.1
DPO ($\beta = 0.9$)	37.2±2.6	40.9±6.1	32.8±2.3	61.2±4.1	52.2±3.1	78.1±2.3
DPO ($\beta = 0.9$) + LoRA	45.9±2.7	**41.3**±6.1	32.4±2.2	64.4±4.0	57.1±3.1	84.7±2.0
KTO ($\beta = 0.3$)	**47.1**±2.7	38.6±6.1	36.4±2.2	**78.3**±3.5	55.8±3.1	85.3±2.0
KTO ($\beta = 0.1$)	47.0±2.7	40.6±6.1	**37.9**±2.3	68.3±3.9	57.2±3.1	86.4±1.9
KTO ($\beta = 0.1$) + LoRA	43.1±2.7	36.2±5.9	37.6±2.2	64.2±4.1	58.5±3.3	87.0±1.9
IPO ($\tau = 0.9$)	38.4±2.7	39.0±5.9	26.9±2.1	71.3±3.8	64.6±3.0	87.4±1.9
IPO ($\tau = 0.99$)	40.7±2.7	36.6±5.9	28.1±2.1	66.3±4.0	64.5±3.0	**87.8**±1.8
IPO ($\tau = 0.99$) + LoRA	36.0±2.6	39.4±5.9	30.2±2.1	66.7±4.0	**65.6**±3.0	**87.8**±1.8

Table 1: Correct results obtained in offline self-training on *Ape210K*. For each preference optimization method, we report results for its two best-performing configurations. **Bold** entries denote the best results among supervised and preference optimization methods per dataset. Confidence intervals are bootstrapped (500 samples, 1,000 repeats).

tion to Ape210K's test set, we evaluate our models on five other math datasets, assessing the robustness of models' capabilities in new *types* of math problems; GSM8K (Cobbe et al., 2021) containing multistep elementary-grade problems requiring on average 3.25 steps to achieve correct result, AQuA-RAT (Ling et al., 2017) with more complex, multiple-choice tasks, and three simpler, one to two-steps datasets: MAWPS (Koncel-Kedziorski et al., 2016), ASDiv-A (Miao et al., 2020), and SVAMP (Patel et al., 2021).

In both self-training variants, we use the trained model to generate training data (see Fig. 1). The generated data consists of the original input prompt (x_i) and associated model predictions (y_i) in the form of a chain-of-thought sequence containing the model's final result at the end. For each prompt, we generate 16 predictions using sampled generation. Annotations of correct results then allow us to automatically annotate each prediction for either being correct (y_i^{OK}), or incorrect (y_i^{NOK}), assigning a set of both correct and incorrect predictions to each input prompt.

For the supervised fine-tuning (SFT) objective, we construct the training dataset from pairs of (x_i, y_i^{OK}). SFT uses a standard next-token prediction with cross-entropy loss and teacher forcing (Bahdanau et al., 2015). All preference optimization (PO) objectives then train on triples (x_i, y_i^{OK}, y_i^{NOK}), with the y_i^{OK} marked as being preferred over y_i^{NOK}. We experiment with three recent preference optimization methods: Direct Preference Optimization; DPO (Rafailov et al., 2023), Kahneman-Tversky Optimization; KTO (Ethayarajh et al., 2024) and Identity Preference Optimization; IPO (Azar et al., 2023). These methods differ in a variety of aspects in the formulation of training loss. For brevity, we direct the reader to the referenced work for further details on preference optimisation methods. Further details of our general training setup can be found in Appendix A.

3.1 Offline Self-training

In the offline variant, we perform a single iteration of collecting predictions with prompts from Ape210K, resulting in over 24,000 prompts with at least one positive and one negative prediction.

All PO methods rely on a crucial parameter β or τ that weights the KL regularization of the trained model according to the original "reference" model. We perform a hyperparameter tuning of this parameter with $\beta \in (0.01, 0.1, 0.3, 0.6, 0.9, 0.99)$ according to in-domain validation accuracy separately for each method and report the results for the best two configurations.

For SFT, we experiment with 3 variants. SFT PLAIN is trained on pairs (x_i, y_i^{OK}). In SFT BALANCED and SFT WITH NEGATIVES, we aim to *compensate* for the potential data disadvantage of SFT PLAIN compared to PO methods exhibiting the trained model to *two* solutions (y_i^{OK}, y_i^{NOK}) per problem: (i) In SFT BALANCED, we use *two differ-*

ent correct predictions y_i^{OK} for one x_i. (ii) In SFT WITH NEGATIVES, we use both positive y_i^{OK} and negative y_i^{NOK} as targets for each x_i. In the training data constructed from y_i^{NOK}, we prefix x_i with a phrase "Write incorrect solution for the following problem". This exposes the model to both correct and incorrect solutions, conceivably helping it to differentiate between the two within SFT training.

Finally, we re-train the best-performing run of each method with a low-rank adaptation (LoRA) (Hu et al., 2021), a commonly used fine-tuning regularization technique that restricts the fine-tuning update of each weight to have a specific low rank. We apply LoRA with a rank of 32 on all linear projections in the model.

Results Table 1 compares the accuracy achieved in offline self-training with each method. A comparison of supervised and more complex preference optimization methods reveals a relatively small difference between the best-performing configurations of both categories. Especially thanks to LoRA regularization, SFT shows the ability to reach results comparable in most datasets. Similar to SFT, LoRA regularization also has a positive effect on DPO, evidencing DPO's inclination to overfitting, as also evidenced by previous work (Azar et al., 2023). Among all supervised methods, the SFT WITH NEGATIVES performs the worst, showing that using negative feedback in supervised training analogically to preference optimization is non-trivial.

On the practical side, we note that PO methods converge much faster than *SFT* methods, achieving the best validation scores on average after around 2,400 training steps compared to 16,600 steps in supervised setups. A detailed comparison of training steps and time can be found in Table 3.

3.2 Online Self-training

In the online self-training, we generate the training data on the fly. Therefore, throughout the whole training, both the positive and negative predictions used for conditioning the updates can realistically be generated by the trained model. Previous work showed that exposing the model to its own outputs might itself improve its robustness (Štefánik et al., 2023). In our online self-training experiments, we additionally evaluate the LM's capability to autonomously improve its *reasoning* capability based on the up-to-date feedback to its own predictions.

A methodology of constructing training samples from the model's predictions for both SFT and PO methods remains identical to the offline variant. Details of data processing can be found in Appendix A.1. As the generation process in online training substantially slows down updates, we restrain the scale of experiments to the best-performing configurations from the offline variant.

Results Table 2 shows the accuracy of training methods in online self-training. This setting reveals much larger differences between methods. Supervised fine-tuning (SFT) improves accuracy on simple one-step and two-step datasets (MAWPS, SVAMP, and ASDiv-A) but substantially degrades performance on out-of-distribution GSM8K and AQuA-RAT. Manual inspection (Appendix B) reveals that the degradation on AQuA-RAT is caused by the model's forgetting of the response format of multiple-choice questions, well-preserved by all PO methods.

Contrary to the SFT, PO methods deliver significant improvements compared to *both* the base model *and* their offline variants (Table 1). Noticeable is the improvement of DPO on GSM8K (by 11.9% of absolute accuracy, i.e. by 22.0% relative to base model), among other cases, suggesting that self-training can mitigate overfitting of PO methods. The best-performing KTO method also substantially improved compared to the offline variant; by 11.3% of accuracy on in-domain Ape210K, or by 16.9% on simpler, out-of-domain MAWPS. Among all other online methods, KTO performs best on every dataset except for AQuA-RAT, on average improving by 12.9% of absolute accuracy, i.e. by 25.9% relative to the base model.

Appendix B provides a per-sample analysis of differences between outputs of SFT and PO models, with a report from a manual assessment of faithfulness of models' rationales in Table 4. Noticeably, we find that while the SFT also achieves large in-distribution improvements, this comes for the price of faithfulness and usability of its rationales, as the SFT model learns to completely or partially omit most of the rationales.

Figure 2 visualizes the dynamics of online self-training in solving known problems during training. We can see that self-training increases the proportion of problems that it always solves correctly and, more importantly, robustly reduces the proportion of problems that it can *not* solve.

	GSM8K	AQuA-RAT	Ape210K	MAWPS	SVAMP	ASDiv-A
BASE MODEL	43.2±2.7	37.8±6.1	26.3±2.1	61.9±4.2	51.8±3.2	78.7±2.3
SFT	27.4±2.5	7.9±3.3	41.2±2.3	63.8±4.2	59.8±3.1	83.3±2.1
DPO ($\beta = 0.9$)	49.1±2.7	**39.8**±5.9	37.9±2.3	79.6±3.4	57.3±3.1	85.6±2.0
KTO ($\beta = 0.1$)	**52.7**±2.7	36.6±6.1	**49.6**±2.4	**85.2**±3.0	**62.6**±3.1	**90.6**±1.6
IPO ($\tau = 0.99$)	49.1±2.8	35.8±5.9	42.2±2.3	81.5±3.4	56.8±3.0	86.6±1.9

Table 2: Correct results obtained in online self-training on *Ape210K* problems. **Bold** denotes the best result per dataset. Confidence intervals are obtained from bootstrapping (500 samples, 1,000 repeats).

4 Conclusions

This work explores the potential of autonomously improving language models for arithmetic reasoning: a task allowing automated, immediate, and objective feedback based on the correct results. We experiment with two settings: (i) *offline self-training*, collecting the feedback in a single iteration, and (ii) *online self-training*, where the model trains continuously from feedback to its up-to-date predictions. In both settings, we apply and compare recent preference optimization methods (DPO, KTO, IPO) with standard supervised training (SFT).

We find that self-training provides an opportunity to improve models' capabilities without *any* new data, using *exclusively* models' own predictions and automated feedback. In addition to the offline variant, online self-training provides further opportunities for data-free improvements attributed to the enhanced robustness of preference optimization methods.

Our work motivates future work towards seeking other sources of implicit training feedback beyond arithmetic reasoning, exemplified in previous work in a reasoning coherence (Akyürek et al., 2024) or consistency (Štefánik et al., 2024). Presenting language models with novel sources of implicit feedback via self-training can fill the gap of the traditional, largely simplified training objectives and empower models to capture more complex structural dependencies necessary in many real-world applications.

Limitations

Despite the fact that our proposed self-training methods do not require any new human annotation, we acknowledge their limitations in the extensive computational requirements given by generating the data. While the data generation for the offline variant can be parallelized, this is more difficult for the online variant, where the model is trained with its own most recent predictions. As a result, our self-training experiments took between 15 and 30 days to converge on a single Nvidia A100 GPU.

The time-demanding character of online self-training experiments is a direct cause of another limitation: a constrained diversity of models and datasets that we experiment with. As such, the experiments and conclusions of our work should inspire experiments with self-training in other applications but may not be generalized to claims on the general effectiveness of self-training.

Acknowledgements

We acknowledge the Centre for Biomedical Image Analysis at Masaryk University supported by MEYS CR (LM2023050 and CZ.02.1.01/0.0/0.0/18_046/0016045 Czech-BioImaging) for providing computational resources for training models and collecting evaluations presented in this paper.

References

Afra Feyza Akyürek, Ekin Akyürek, Leshem Choshen, Derry Wijaya, and Jacob Andreas. 2024. Deductive closure training of language models for coherence, accuracy, and updatability. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9802–9818, Bangkok, Thailand and virtual meeting. ACL.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences. *Preprint*, arXiv:2310.12036.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, USA.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *Preprint*, arXiv:2402.01306.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *CoRR*, abs/2103.03874.

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

Yebowen Hu, Kaiqiang Song, Sangwoo Cho, Xiaoyang Wang, Hassan Foroosh, and Fei Liu. 2023. DecipherPref: Analyzing influential factors in human preference judgments via GPT-4. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8344–8357, Singapore. ACL.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. ACL.

Marek Kadlčík, Michal Štefánik, Ondřej Sotolář, and Vlastimil Martinek. 2023. Calc-X and Calcformers: Empowering Arithmetical Chain-of-Thought through Interaction with Symbolic Systems. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12101–12108, Singapore. ACL.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *CoRR*, abs/1705.04146.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *Preprint*, arXiv:2308.09583.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2017. Mixed precision training. *CoRR*, abs/1710.03740.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *Preprint*, arXiv:2205.12255.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235.

Michal Štefánik, Marek Kadlčík, and Petr Sojka. 2023. Soft Alignment Objectives for Robust Adaptation of Language Generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8837–8853, Toronto, Canada. ACL.

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

Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *Preprint*, arXiv:2211.14275.

Shibo Wang and Pankaj Kanwar. 2023. Bfloat16: The secret to high performance on cloud tpus.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pages 15476–15488. Curran Associates, Inc.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Ape210k: A large-scale and template-rich dataset of math word problems. *CoRR*, abs/2009.11506.

Michal Štefánik, Marek Kadlčík, and Petr Sojka. 2024. Concept-aware data construction improves in-context learning of language models.

A Training Details

In every configuration of both preference and supervised training, the model is trained with Adafactor (Shazeer and Stern, 2018) optimizer with an effective batch size of 32, a learning rate of $2 \cdot 10^{-5}$ with 1,000 warmup steps, and a linear decay to 0 in 1 million steps. The models were trained in bfloat16 (Wang and Kanwar, 2023) precision with mixed precision training (Micikevicius et al., 2017). The training terminates after convergence on the in-domain dataset (Ape210K), and then the best checkpoint from the training is selected according to in-domain validations.

Each of our experiments can be reproduced with a single Nvidia A100/A40 graphic card and 32GB of RAM. Note that especially the online self training experiments can take up to 31 days to converge.

A.1 Online self-training

To create new data in online self-training, we sample a random problem from Ape210K and generate predictions with the current model. Next, we label each solution as correct if its result matches the one in the data. The online self-training process is illustrated in Figure 1.

Method	Training steps	Wall Time
SFT PLAIN	16,000	17 h
SFT PLAIN + LoRA	98,000	120 h
SFT BALANCED	14,000	15 h
SFT WITH NEGATIVES	20,000	21 h
DPO $\beta = 0.99$	1,800	2 h
DPO $\beta = 0.9$	1,800	2 h
DPO $\beta = 0.9$ LoRA	2,600	6 h
KTO $\beta = 0.3$	3,800	7 h
KTO $\beta = 0.1$	4,800	8 h
KTO $\beta = 0.1$ LoRA	16,400	35 h
IPO $\tau = 0.9$	1,200	2 h
IPO $\tau = 0.99$	1,200	2 h
IPO $\tau = 0.99$ LoRA	1,600	4 h

Table 3: Number of steps and wall time that different methods take until convergence in offline self-training shows that preference optimization methods converge 5–20 times faster than supervised training. Note that wall time fluctuates based on hardware usage by other programs and should be taken as an approximate measure.

In this experiment, we again compare supervised training and preference optimization. In all variants, we generate 16 solutions per problem with top-k=50 sampling using the latest model, but the subsequent data processing is method-specific.

Supervised training: After generating the solutions, we discard the incorrect ones. The correct solutions are oversampled to generate 32 training examples. Each solution is sampled at most 4 times each, and all solutions are used almost the same number of times (maximal difference of one).

Preference Optimization: After the solutions are generated, we create all possible pairs of solutions where one solution has a correct result and the other one does not. We then sample with repetition from the pairs, such that:

1. every correct solution is used at most 4 times,
2. the number of preference pairs per problem is 32 if possible without violating the condition 1,
3. all correct solutions are used almost the same number of times,
4. all incorrect solutions are used almost the same number of times.

Almost the same number of times means a maximal difference of one.

In both supervised and preference training, the training instances are put into a buffer with 8192 slots, from which they are sampled randomly for training. When a batch of data gets sampled, it is removed from the buffer, and new data are generated with the correct model to fill the empty slots.

During training, we track the proportion of problems that the models consistently solve correctly or fail to solve across 16 trials. Figure 2 shows the progression of the best-performing *online* training run elaborating the preference optimisation with KTO.

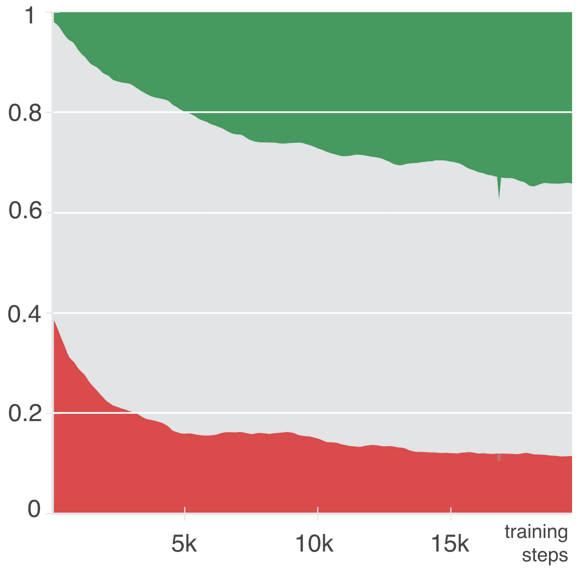

Figure 2: **Training dynamics of online training:** The fraction of training problems for which the model predicted all ■ and none ■ of 16 trials correctly during training of the online KTO with $\beta = 0.1$. The fraction is computed from a sliding window of the last 1000 problems and the chart is smoothed for visual clarity.

B Output analyses

Aiming to better understand the difference between self-training with preference optimization methods and supervised training, we manually analyze a set of randomly chosen rationales generated for prompts of the GSM8K test set. We collect the rationales from (i) the original checkpoint, (ii) the checkpoint trained in online self-training and supervised method (denoted SFT), and (iii) the checkpoint trained on online self-training with the best-performing method (KTO). Due to the time complexity of evaluating long chain-of-thought output sequences, we analyze 20 predictions marked as correct for each checkpoint.

	Original	SFT	KTO
Inconsistency	20%	5%	30%
Missing association	0%	70%	0%
Missing rationale	0%	30%	0%
Missing ratio. part	5%	100%	15%
Not understandable	30%	0%	30%

Table 4: Output analysis: ratio of model predictions containing one of our identified flaws, evaluated on a sample of models' *correct* predictions.

Within the analysis, we encounter 5 types of dominant flaws that models' outcomes exhibit, even when being correct:

1. **Inconsistency:** Within the rationale, the model generates a new reasoning step which is not logically consistent with previous ones.
2. **Missing association:** Model's rationale contains steps that are difficult to assess for consistency, as they lack the associations of units (e.g., of size, distance, or volume) or subjects from input prompt or intermediate computation.
3. **Missing rationale:** Model only generates the result without any rationale associated with it.
4. **Missing rationale part:** Model's rationale is missing a specific segment, making it impossible to fully check the model's computation process.
5. **Not understandable:** Model's rationale contains text that is incomprehensible by the annotator, thus impossible to judge for logical correctness.

The results of this analysis are summarized in Table 4. A set of predictions for identical prompts and responses of SFT and KTO checkpoints can also be found in Appendix B.1.

B.1 Examples of predictions

For an illustration of differences between the supervised and preference optimization checkpoints, below we include a set of predictions of SFT and KTO checkpoints for randomly selected prompts of GSM8K collection. These examples can also be found by associated IDs in the original dataset repository[3]. For better readability, we removed the

[3]https://huggingface.co/datasets/MU-NLPC/Calc-gsm8k

navigation tokens used by the model for interaction with the calculator tool during the generation.

id: gsm8k__aBNdImSK9ddCwuYb

Question: Tobias bought a big pizza with 60 pieces. He ate 2/5 of the pieces on the first day, 10 pieces on the second day, and 7/13 of the remaining pieces on the third day. How many pizza pieces has he eaten so far?

Correct answer: 48

Prediction with SFT:

> 60 * 2/5 = 28 ✗

Prediction with KTO:

> On the first day, Tobias ate 60 * 2/5 = 60*2/5 24 24 pieces. The number of pieces that remained after the second day is 60 - 24 = 60-24 36 36 pieces. On the third day, he ate 36 * 7/13 = 36*7/13 252/13 = around 19.384615 25 pieces. The total number of pieces he has eaten so far is 10 + 252/13 = 10+252/13 382/13 = around 29.384615 38 pieces. Tobias ate a total of 24 + 10 + (382/13) = 24+10+352/13 794/13 = around 61.076923 69 pieces. 794/13 = around 69.076923 ✗

id: gsm8k__qIygQI2WNfRPWoYj

Question: Jennie is helping at her mom's office. She has a pile of 60 letters needing stamps, and a pile of letters already stamped. She puts stamps on one-third of the letters needing stamps. If there are now 30 letters in the pile of already-stamped letters, how many were in that pile when Jennie began?

Correct answer: 10

Prediction with SFT:

> 60 / 3 = 40 ✗

Prediction with KTO:

> Jennie stamps 60 / 3 = 60/3 20 20 letters. Then the number of letters still needing stamps is 60 - 20 = 60-20 40 40. The number of letters already stamped is 30 - 20 = 30-20 10 10. There were 40 + 10 = 40+10 50 50 letters in the pile when Jennie began. 50 ✗

id: gsm8k__0oOjz5Ub66DF4inZ

Question: There are 6 trees in Chris's yard. Ferdinand has half the number of trees that Chris has. Harry has 5 more than twice the number of trees that Ferdinand has. How many more trees are in Harry's yard than Ferdinand's yard?

Correct answer: 8

Prediction with SFT:

> 9 ✗

Prediction with KTO:

> Half the number of trees that Chris has is 6 trees / 2 = 6/2 3 3 trees. Twice the number of trees that Ferdinand has is 3 trees * 2 = 3*2 6 6 trees. Harry has 5 trees + 6 trees = 6+5 11 11 trees. There are 11 trees - 6 trees = 11-6 5 5 more trees in Harry's yard than in Ferdinand's yard. 5 5 5 more trees in Harry's yard than in Ferdinand's yard. 5 5 5 more trees in Harry's yard than in Ferdinand's yard. 11 11 5 more trees than in Ferdinand's yard. 11-6 5 5 more trees in Harry's yard. 5 ✗