SELF-TAUGHT SELF-CORRECTION FOR SMALL LANGUAGE MODELS

Viktor Moskvoretskii
Skoltech, HSE University
vvmoskvoretskii@gmail.com

Chris Biemann
Universität Hamburg

Irina Nikishina
Universität Hamburg

ABSTRACT

Although small language models (SLMs) are efficient and perform well across a wide range of tasks, they are still prone to errors. A critical and highly sought-after capability is their ability to self-correct. While prior research has often depended on external tools or large proprietary models, this work explores self-correction in SLMs through iterative fine-tuning using exclusively self-generated data. We propose the Self-Taught Self-Correction (STaSC) algorithm and its generalized variant, G-STaSC. Experimental results on a question-answering task highlight the effectiveness of STaSC over alternative methods and G-STaSC variations, offering significant insights into the mechanisms of self-correction. To facilitate further research, we provide open access to our user-friendly codebase and lightweight models.

1 INTRODUCTION

Recent advanced language models employ complex reasoning (Guo et al., 2025) and meta-reasoning (Xiang et al., 2025), expanding their capabilities. However, even the most advanced models are prone to errors, including hallucinations (Huang et al., 2025) and logical inconsistencies (Ghosh et al., 2024), requiring symbolic or human verification. To address those problems, self-correction — the ability to revise their own outputs — has been evolved (Madaan et al., 2023).

The existing approaches mostly use zero-shot prompting (Madaan et al., 2023; Shinn et al., 2024), external evaluators for correction or feedback (Zhang et al., 2024) or apply large proprietary models and focus specifically on mathematical tasks (Kumar et al., 2024). In this study, we focus on the self-correction of small language models (SLMs) through iterative fine-tuning on self-generated data. This approach is efficient as it avoids reliance on stronger external models or tools during inference, revealing the model's intrinsic self-correction abilities without external influence.

We introduce the Self-Taught Self-Correction (STaSC) algorithm, which adapts the core idea of STaR (Zelikman et al., 2022) for self-correction. Additionally, we present its generalized version, G-STaSC, which unifies and extends both STaSC and the Self-Correction (SC) algorithms (Welleck et al., 2022). Unlike prior methods, G-STaSC offers flexible control over initial answer exploration, correction filtering, and iterative fine-tuning, effectively encompassing both STaSC and SC as special cases, and showing how different algorithmic choices affect self-correction performance.

Our results on the *Natural Questions* dataset (Kwiatkowski et al., 2019) show that SLMs can learn to self-correct using self-synthesized data, while also improving their initial answer quality despite being trained solely for corrections. We release easy-to-use and adaptable code for self-correction and self-improvement algorithms at `https://github.com/VityaVitalich/STASC/`.

The contributions of the paper are as follows:

- We propose the Self-Taught Self-Correction (STaSC) algorithm and its generalized version, G-STaSC, which unifies and extends existing self-correction methods.
- We conduct extensive experiments on a purely Natural Language Processing (NLP) task — Question Answering — using an open-source SLM, demonstrating its ability to learn self-correction with self-synthesized data.
- We release open-source, easily adaptable code for self-correction, along with efficient SLMs with fewer than 2B parameters, making self-correction practical and accessible.

2 GENERALIZED SELF-TAUGHT SELF-CORRECTION

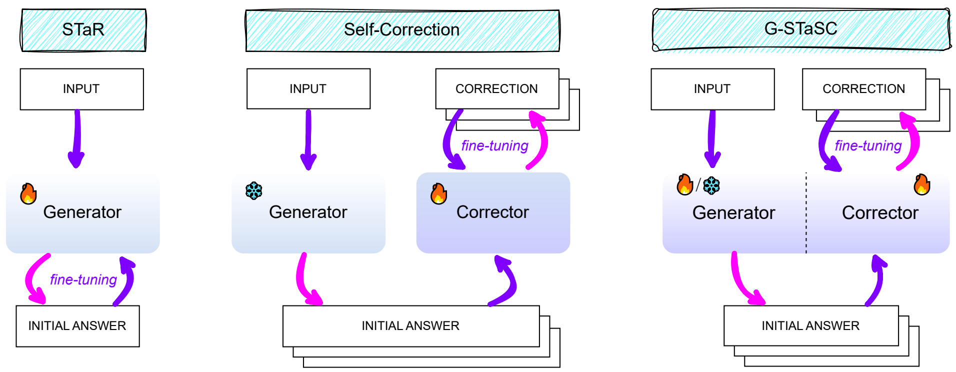

Figure 1: Illustration of the self-improvement method **STaR** (left) Zelikman et al. (2022), self-correction method **SC** (center) Welleck et al. (2022), and our method, G-STaSC (right). STaCS offers flexible control over initial answer exploration, correction filtering, and iterative fine-tuning. It is inspired by STaR and effectively encompasses SC as a special case. SC and G-STaSC allow several initial answers and corrections. The dotted line in the G-STaSC denotes two possible setups: fine-tuning the model and generating from it at the next iteration (Evolving Fine-Tuning) and keeping the Generator frozen and fine-tuning the Corrector model only (Fixed Fine-Tuning).

In this Section, we introduce the Self-Taught Self-Correction (STaSC) algorithm, an adaptation of STaR (Zelikman et al., 2022) for self-correction through iterative fine-tuning on self-synthesized data. We also present its generalized version, G-STaSC, which encompasses STaSC, the Self-Correction (SC) algorithm (Welleck et al., 2022), and their possible multiple variations.

Formally, Algorithm 1 begins with an initial language model state, M_0, and an initial dataset, D_0, consisting of input-output pairs (x, y). Additionally, we define a number of improvement iterations T, the number of sampled initial generations N_{init}, the number of sampled corrections N_{corr}, and a reward function r, which evaluates the quality of model-generated outputs.

The algorithm follows the structure of *STaR* for self-correction when using $N_{\text{init}} = 1$, $N_{\text{corr}} = 1$, sampling initial answers with M_{t-1}, forming strictly improving pairs, and training from the initial model M_0. It also adopts the form of SC when configured with $N_{\text{init}} = 25$, $N_{\text{corr}} = 3$, sampling initial answers with M_0, forming strictly improving pairs, and training from the latest model M_{t-1}.

Step 1: Sample Initial Answers. In the first step, we sample N_{init} initial answers $\hat{y}^1$ for each input x in the dataset D_0. A key design choice at this stage is whether to sample with Fixed Initialization M_0, as done in the SC paper, or with Evolving Initialization from the previous iteration M_{t-1}, as in STaSC. The former approach ensures robustness to variations in the self-improvement process, while the latter allows for greater exploration, potentially leading to more diverse refinements.

Step 2: Sample Corrections. At the second step, we sample N_{corr} corrections $\hat{y}^2$ for each output $\hat{y}^1$ in dataset D_0 using the model from the last iteration M_{t-1}.

Step 3: Filter Corrections. At the third step, we filter the corrections using the reward function $r(\hat{y}^2)$ to form the fine-tuning dataset D_t. The design choice here is whether to use Improving Filter, selecting corrections that are strictly improving reward $r(\hat{y}^2) > r(\hat{y}^1)$, as done in STaSC and SC, or Non-Decreasing Filter, selecting those may also remain unchanged if the initial answer was already correct, satisfying $(r(\hat{y}^2) = r(\hat{y}^1)) \cap (r(\hat{y}^1) \geq \tau)$, as proposed in SCoRE (Kumar et al., 2024). The former approach enforces strict improvement for every input, while the latter allows the model to retain an answer if it was already correct.

Step 4: Fine-Tuning. At the fourth step, we fine-tune the model on the dataset D_t formed in Step 3 to obtain the improved model M_t. The design choice here is whether to use Fixed Fine-Tuning with the initial model M_0, as done in STaSC, or Evloving Fine-Tuning with the model from the previous iteration M_{t-1}, as in Self-Correction.

3 EXPERIMENTAL SETUP

We evaluate our algorithm on the QA task using the Natural Questions dataset (Kwiatkowski et al., 2019), which consists of factual simple questions. To ensure computational efficiency and consistency with previous studies (Moskvoretskii et al., 2025; Trivedi et al., 2022; Jeong et al., 2024), we use a representative subset.

Algorithm 1 Generalized Self-Taught Self-Correction (G-STaSC)

Require: Initial model M_0, dataset D_0, number of iterations T, initial samples N_{init}, correction samples N_{corr}, reward function r

for $t = 1$ to T **do**
  Step 1: Sample Initial Answers
  $\hat{Y}_i^1 = \{\hat{y}_{ij}^1\}_{j=1}^{N_{\text{init}}} \sim M(x_i), \quad \forall x_i \in D_0$
    Option 1: $M = M_{t-1}$ (Evolving Initialization)
    Option 2: $M = M_0$ (Fixed Initialization)
  Step 2: Sample Corrections
  $\hat{Y}_i^2 = \{\hat{y}_{ijk}^2\}_{k=1}^{N_{\text{corr}}} \sim M_{t-1}(x_i, \hat{y}_{ij}^1), \quad \forall \hat{y}_{ij}^1 \in \hat{Y}_i^1$
  Step 3: Filter Corrections
  $D_t^+ = \{(x_i, \hat{y}_{ij}^1, \hat{y}_{ijk}^2) \mid r(\hat{y}_{ijk}^2) > r(\hat{y}_{ij}^1)\}$
  $D_t^= = \{(x_i, \hat{y}_{ij}^1, \hat{y}_{ijk}^2) \mid r(\hat{y}_{ijk}^2) = r(\hat{y}_{ij}^1) \wedge r(\hat{y}_{ij}^1) \geq \tau\}$
    Option 1: $D_t = D_t^+$ (Improving Filter)
    Option 2: $D_t = D_t^+ \cup D_t^=$ (Non-Decreasing Filter)
  Step 4: Fine-Tuning
  $M_t = \text{train}(M, \{\hat{y}_{ijk}^2 \mid (x_i, \hat{y}_{ij}^1, \hat{y}_{ijk}^2) \in D_t\})$
    Option 1: $M = M_0$ (Fixed Fine-Tuning)
    Option 2: $M = M_{t-1}$ (Evolvong Fine-Tuning)
end for

Table 1: Maximum reward r over iterations for initial answer $r(\hat{Y}^1)$ and for correction $r(\hat{Y}^2)$ for different number of samples and initial generations. Bold corresponds to the best performance.

N_{init}	N_{corr}	$max\{r(\hat{Y}^1)\}$	$max\{r(\hat{Y}^2)\}$
1	1	-	-
	3	0.248 ± 0.011	0.208 ± 0.011
	5	0.230 ± 0.011	0.228 ± 0.021
	10	0.240 ± 0.010	0.236 ± 0.022
3	1	0.236 ± 0.007	0.232 ± 0.018
	3	0.264 ± 0.015	0.238 ± 0.018
	5	0.273 ± 0.017	0.236 ± 0.019
	10	0.283 ± 0.018	0.242 ± 0.023
5	1	0.273 ± 0.012	$\mathbf{0.250 \pm 0.024}$
	3	0.295 ± 0.019	0.244 ± 0.023
	5	0.300 ± 0.020	0.248 ± 0.029
	10	0.308 ± 0.023	0.244 ± 0.023
10	1	0.326 ± 0.030	0.246 ± 0.023
	3	0.324 ± 0.029	0.236 ± 0.023
	5	$\mathbf{0.328 \pm 0.062}$	0.154 ± 0.029

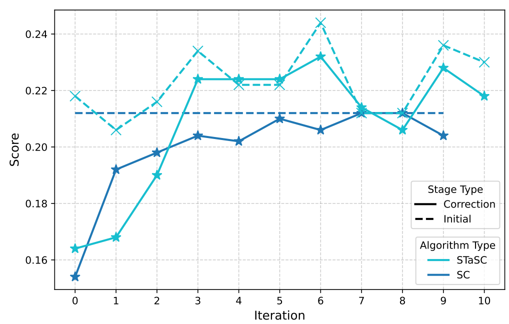

Figure 2: Initial Accuracy and Correction Accuracy for STaSC and SC algorithms. SC Initial Accuracy is shown as dashed line due to fixed initialization. All runs start from different random states.

Following prior work, we adopt In-Accuracy as the primary evaluation metric, which measures whether the generated answer contains the reference answer. We conduct experiments using Qwen-2.5-1.5B (Qwen et al., 2025), employing default generation parameters. The default setup for self-correction is 2-shot. More details are provided in Appendix B.

4 Results & Discussion

In this section, we provide the results and discuss them, inspecting the *STaSC* algorithm design.

4.1 Impact of N_{init} and N_{corr}

Firstly, we examine how the selection of parameters N_{init} and N_{corr} influences algorithm performance. To encourage exploration, we sample initial answers from the model's previous iteration and apply only improving corrections, following SC and STaSC. Training is conducted from the base model to ensure stability. From the Table 1, we can see that a greedy approach for *Qwen-2.5-1.5B* fails to ensure convergence, as no improvement corrections are observed in the first iteration. In contrast, increased exploration yields significantly better results, likely due to the weaker alignment of the initial model. Overall, setting $N_{\text{init}} = 5$ and $N_{\text{corr}} = 5$ is the most stable and decently performing configuration. Therefore, this configuration is adopted for subsequent experiments.

4.2 Impact of G-STaSC Variations

Table 2 in the Appendix B presents the performance of G-STaSC under different algorithmic design choices, and Figures 3 and 4 illustrate the correction performance of STaSC and the SC algorithm with various configurations.

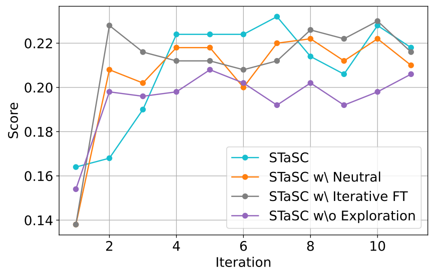

Figure 3: STaSC versions corrections performance.

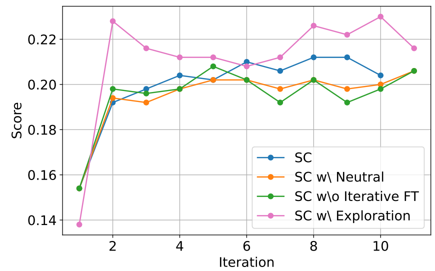

Figure 4: SC versions corrections performance.

Evolving Initialization significantly boosts performance, enhancing SC results while causing a notable drop for STaSC with Fixed Initialization. This is likely due to the model's limited ability to improve initial answers when starting from a fixed state. **Non-Decreasing Filter** slightly degrades performance across both STaSC and SC, affecting both initial answers and corrections. This suggests that allowing non-improving corrections may introduce confusion during fine-tuning. **Evolving Fine-Tuning** in STaSC shows an interesting trend—initial performance gains followed by slight degradation over iterations, with recovery observed in later stages. This fluctuation suggests that evolving fine-tuning requires more careful tuning. For SC, evolving fine-tuning consistently improves performance, likely because it serves as the primary source of exploration when using fixed initialization. Overall, STaSC achieves the best performance when combining Evolving Initialization, Improving Filter, and Fixed Fine-Tuning.

4.3 STaSC vs SC

Figure 2 compares STaSC and SC in terms of both initial answers and corrections over training iterations. Starting from roughly the same baseline, STaSC consistently outperforms SC, achieving higher quality in both initial answers and corrections, with corrections often matching or slightly surpassing the quality of initial answers. SC demonstrates greater stability with fewer fluctuations, and its corrections gradually improve to match the quality of initial answers in the later iterations.

5 Related Works

Self-correction is a growing research area, defined as the ability of language models (LMs) to refine their outputs during inference, sometimes using external tools or knowledge (Kamoi et al., 2024). Most existing work focuses on external feedback from knowledge bases or verification tools (Jiang et al., 2023; Gou et al., 2024; Pan et al., 2023; Xu et al., 2023), which, while effective, are often unavailable, costly, and do not challenge LMs to develop intrinsic reasoning.

An alternative is intrinsic self-correction, where models refine their outputs without external critics. This can be done in zero-shot settings (Madaan et al., 2023) or via external models trained on synthetic errors (Paul et al., 2024) or self-generated data (Welleck et al., 2022). However, these approaches still rely on external verification, limiting true self-correction.

The only work addressing genuine self-correction is the SCoRE framework (Kumar et al., 2024), which introduces multi-turn reinforcement learning within the same model. Yet, SCoRE lacks formal theoretical grounding, overlooks baseline algorithm adaptations, and relies on large proprietary models without open-source code, limiting its reproducibility and impact.

6 Conclusion

In this study, we presented the Self-Taught Self-Correction (STaSC) algorithm, inspired by STaR (Zelikman et al., 2022), along with its generalized version, G-STaSC, which unifies STaSC and the previously developed Self-Correction (SC) algorithm (Welleck et al., 2022). Our experiments on a QA task with SLM demonstrated the superiority of STaSC over SC and its variations, while also providing new insights into self-correction. To support future research, we have open-sourced our code and lightweight models.

REFERENCES

Bishwamittra Ghosh, Sarah Hasan, Naheed Anjum Arafat, and Arijit Khan. Logical consistency of large language models in fact-checking, 2024. URL https://arxiv.org/abs/2412.16100.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing, 2024. URL https://arxiv.org/abs/2305.11738.

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

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2): 1–55, January 2025. ISSN 1558-2868. doi: 10.1145/3703155. URL http://dx.doi.org/10.1145/3703155.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*, 2024.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation, 2023. URL https://arxiv.org/abs/2305.06983.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Viktor Moskvoretskii, Maria Lysyuk, Mikhail Salnikov, Nikolay Ivanov, Sergey Pletenev, Daria Galimzianova, Nikita Krayko, Vasily Konovalov, Irina Nikishina, and Alexander Panchenko. Adaptive retrieval without self-knowledge? bringing uncertainty back home. *arXiv preprint arXiv:2501.12835*, 2025.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. REFINER: Reasoning feedback on intermediate representations. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1100–1126, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-long.67/.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*, 2022.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, et al. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-though. *arXiv preprint arXiv:2501.04682*, 2025.

Wenda Xu, Danqing Wang, Liangming Pan, Zhenqiao Song, Markus Freitag, William Yang Wang, and Lei Li. Instructscore: Explainable text generation evaluation with finegrained feedback. *arXiv preprint arXiv:2305.14282*, 2023.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning, 2022. URL `https://arxiv.org/abs/2203.14465`.

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. Small language models need strong verifiers to self-correct reasoning. *arXiv preprint arXiv:2404.17140*, 2024.

A LIMITATIONS

- The study is limited to evaluating only one Small Language Model (SLM), which does not prove the generalizability of the findings.
- The chosen SLM may lack the capacity to fully capture complex patterns, potentially limiting the effectiveness of the self-correction process.
- Each experiment was carried out with a single run, which can introduce variability and reduce the robustness of the results.
- The evaluation is confined to a single Question Answering (QA) task, leaving the performance on other tasks or domains unexplored.
- Suboptimal hyperparameter tuning during training may have hindered the model's potential performance and learning efficiency.
- The study does not include a detailed analysis of the types or patterns of corrections made by the model, limiting insights into the self-correction mechanism.
- The reward function used for training may not adequately capture the desired behavior, potentially affecting the quality of the self-correction process.

B TECHNICAL DETAILS

The models are fine-tuned with a learning rate of 1×10^{-5}, for 1 epoch with a batch size of 4. We apply a weight decay of 0.1 and use the Adam optimizer with default betas, along with a cosine annealing scheduler. Training is performed using Fully Sharded Data Parallel on two A100 GPUs, with an estimated total compute of 80 GPU hours.

Initial answer sampling prompt

> You are a helpful reasoning assistant in general domain question answering. Please reason through the question step by step very shortly before giving a final answer.
>
> Generate a short chain-of-thought rationale very shortly, and then provide the final answer.
>
> Step-by-step reasoning:
> Final Answer:
>
> Question: {question text}
>
> Reason step by step very shortly, then conclude with the answer.

Correction sampling prompt

> You are a helpful reasoning assistant in general domain question answering.Your task is to correct the initial response if it is incorrect.
>
> Below is the question and the initial answer. Generate a correction to the initial answer if it is incorrect. Disregard the information you already have, look for other options. Do not use the information that does not match your criteria.
>
> Step-by-step reasoning:
>
> Final Answer:
>
> Question: Which Portuguese soccer player has the most goals?
>
> Step-by-step reasoning: Christiano Ronaldo has scored 900 goals in a career spanning 22 years. I cannot think of anyone else scoring more than him.
>
> Final Answer: Christiano Ronaldo
>
> Step-by-step reasoning: Yes, Christiano Ronaldo is the correct answer. Other details are unimportant.
>
> Final Answer: Christiano Ronaldo
>
> Question: Which actress played in both movies: one about pregnancy and the other about Tracey?
>
> Step-by-step reasoning: 1. Identify movies about pregnancy and about Tracey. 2. Movies about pregnancy include 'Juno' and 'Knocked Up'. 3. Movie about Tracey is likely 'Tracy Chapman' or a documentary, but 'Tracy' could be a reference to 'Tracy Flick' in the movie 'Election'. 4. Considering well-known movies with these themes, actress Reese Witherspoon comes to mind as a possible match.

Final Answer: Reese Witherspoon

Step-by-step reasoning: Let's break down the answer. 'Juno' is indeed a movie about pregnancy, but it starred Ellen Page (now Elliot Page), not Reese Witherspoon. Reese Witherspoon did not star in 'Juno.' Therefore, the association between Reese Witherspoon and 'Juno' is incorrect. Movies About Tracey is likely 'Tracey Fragments (2007)', a movie in which Ellen Page also starred as the lead character, Tracey Berkowitz. Ellen Page (Elliot Page) connects both movies.

Final answer: Ellen Page (now Elliot Page)

Question: {question text}

Initial Answer: {initial answer}
Write a correction if the initial answer is incorrect.

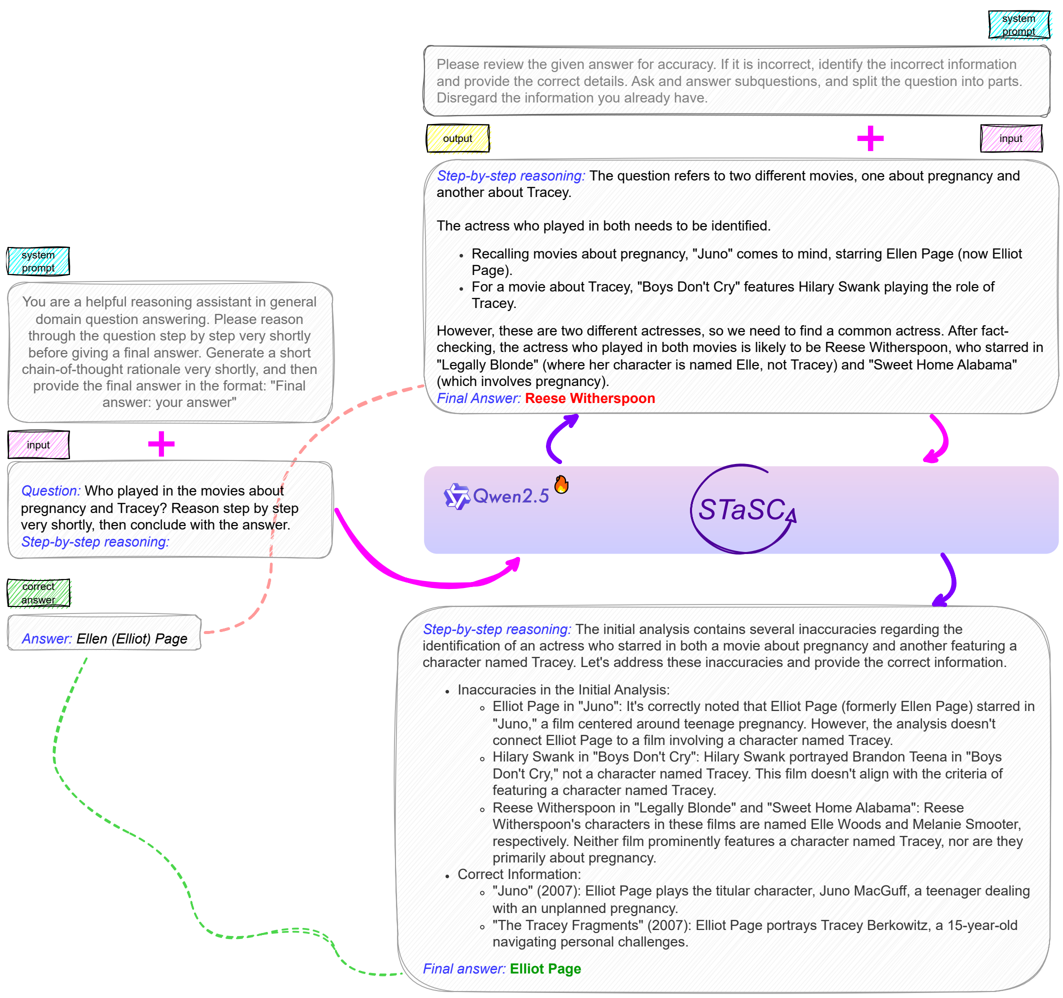

Figure 5: Example of the STaSC pipeline.

Table 2: Maximum reward r over iterations for initial answer $r(\hat{Y}^1)$ and for correction $r(\hat{Y}^2)$ for different settings of STaSC Algorithm. Bold values correspond to the best performance, underlined represent second best.

Step 1 Model	Step 3 Filter	Step 4 Model	$max\{r(\hat{Y}^1)\}$	$max\{r(\hat{Y}^2)\}$
M_0	Improving	M_0	0.212	0.208 ± 0.014
		M_{n-1}	0.212	0.212 ± 0.016
	Non-Decreasing	M_0	0.212	0.198 ± 0.012
		M_{n-1}	0.212	0.206 ± 0.014
M_{n-1}	Improving	M_0	$\mathbf{0.244 \pm 0.011}$	$\mathbf{0.232 \pm 0.023}$
		M_{n-1}	0.236 ± 0.009	$\underline{0.230 \pm 0.024}$
	Non-Decreasing	M_0	$\underline{0.240 \pm 0.009}$	0.222 ± 0.023
		M_{n-1}	0.234 ± 0.013	0.228 ± 0.022