Step-Tagging Early-Stopping: Toward controlling the generation of Language Reasoning Models through black-box step monitoring

Yannis Belkhiter [1 2] **Seshu Tirupathi** [1] **Giulio Zizzo** [1] **John D. Kelleher** [3 2]

Abstract

Language Reasoning Models (LRMs) have shown impressive performance on solving complex problems requiring multi-steps. However, a growing body of studies show that LRMs are still inefficient, over-generating verification and self-reflection steps. To address this challenge, we introduce the Step-Tagging Early-Stopping (ST-ES) framework, a lightweight sentence-classifier enabling real-time annotation of the type of reasoning steps that an LRM is generating. We show that limiting the count of specific step-type - especially verification and self-reflection steps - yields a more accurate and token-efficient early-stopping criterion than token-count baseline, and that each step-types yield to a different efficiency trade-off. Unlike prior dynamic early-stopping methods, ST-ES operates in a full black-box setting, and offers interpretable early-stopping criteria. We evaluate ST-ES on three mathematical reasoning benchmarks, namely, MATH500, GSM8K, AIME and two knowledge and reasoning benchmarks, MMLU and GPQA respectively. We achieve 20 to 50% token reduction while maintaining comparable accuracy to standard generation.

1. Introduction

For the past few years, the field of Language Reasoning Models (LRMs) has experienced significant growth in terms of capabilities. Inference Time Scaling, with work on model prompting has emerged as a popular field with the goal of making models more accurate at reasoning (Wei et al., 2023; Wang et al., 2023). At the same time, fundamental work on Training Time Scaling has led to the release of strong native reasoning models (Xu et al., 2025a).

[1]IBM Research Europe, Dublin, Ireland [2]Trinity College Dublin, Ireland [3]ADAPT Research Centre, Dublin, Ireland. Correspondence to: Yannis Belkhiter <yannis.belkhiter@ibm.com>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

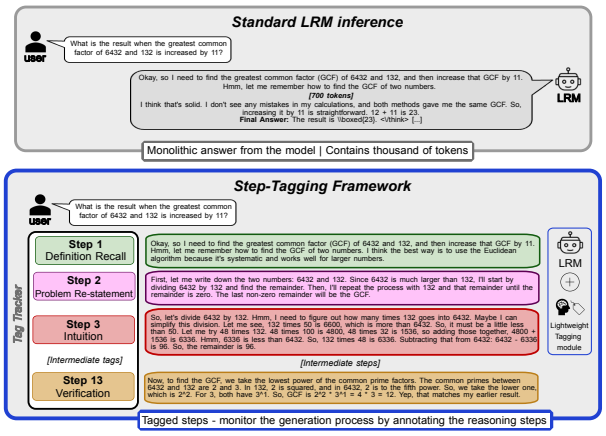


Figure 1. Step-Tagging: a framework for monitoring the generation of LRMs - example on sample 39 from MATH500 test with DS-Qwen14B, using the ReasonType taxonomy - seed 42

However, recent surveys have shown that LRMs need to generate a very large number of tokens—several thousands—in order to generate an accurate answer on challenging questions (Qu et al., 2025; Chen et al., 2025b; Sui et al., 2025). This behavior makes reasoning models extremely inefficient - scaling in both compute resources and inference time. Although recent works have suggested solutions to this problem, most approaches overlook the possibility of monitoring the reasoning as it is being generated, and using this information to improve the inference such as its efficiency. Indeed, efficient inference-time scaling methods are either static - fixed budgets/black-box - or dynamic, but they requires access the model's internals and do not exploit the text generated by LRMs to guide early stopping. To address this challenge, this paper aims to offer a new perspective on the efficiency of LRMs by focusing on dynamic monitoring of model's outputs. Our contributions are as follows:

- **Step-Tagging module:** We introduce the *Step-Tagging* module (see Figure 1), an online lightweight sentence classifier capable of identifying the nature of each step generated by LRMs. This novel framework offers a tool to systematically monitor the generation of LRMs.
- **Step-Tagging Early-Stopping (ST-ES):** Leveraging the *Step-Tagging* module, we found that the type of reasoning steps plays a role in determining the early-stopping

condition. Based on these observations, we built an interpretable early-stopping framework that dynamically stops token generation based on reasoning steps types and counts, calibrated on both models and problem complexity. Tested on three open-source LRMs across five reasoning datasets, our framework reduced token generation by 20-50% while maintaining a comparable accuracy.

2. Related Work

To render models less verbose and more efficient, Train and Test Time Scaling approaches have been explored (Qu et al., 2025; Li et al., 2025; Chen et al., 2025a). Also, recent work has explored monitoring the generation of LRMs.

Efficient Reasoning during Inference. Researchers have explored Inference Time Scaling technique to increase the efficiency of models (Qu et al., 2025). *Model Switch* uses a router module to select small or large models for inference depending on the complexity of the problem (Ong et al., 2025). Similarly, *System Switch* looked at dynamically selecting inference settings based on the problem (Aytes et al., 2025). *Length Budgeting* aims to reduce the budget allocated to the generation of answers. Works such as Lee et al. (2025); Han et al. (2025); Xu et al. (2025b) showed that careful prompt engineering can lead to more efficient generation compared to standard inference. In addition, Pu et al. (2025) demonstrated that calibration experiments can be performed to estimate the optimal number of tokens to solve particular problems. However, these techniques rely on fixed budgets or template constraints defined before inference and therefore do not adapt to the evolving reasoning traces of the model. As a result, they cannot exploit the information contained in the model's own generated reasoning during inference. In contrast, Yang et al. (2025) and Wang et al. (2026) proposes dynamic early stopping criteria based on the model's confidence (DEER), or entropy (EAT), respectively. Even though these methods are dynamic based on the model's generation, they requires full or partial access to the internals, operating in a white-box (DEER), or gray-box (EAT) settings. We address this by suggesting a dynamic black-box early-exit criteria, operating only on the text generated (no use of internals or proxy models).

Monitoring LRM generation. We observe an emerging theme of research on monitoring LRM generation at a step level. Specifically, Lee & Hockenmaier (2025) proposes a taxonomy of reasoning traces evaluators, but acknowledge that existing monitoring approaches are not adapted to complex reasoning traces. Similarly, Golovneva et al. (2023) prompts a model to generate step-by-step reasoning, and defines an error-type taxonomy to evaluate reasoning steps. Nevertheless, this method is post-hoc and focuses on measuring the quality of reasoning. Closer to our work, LCoT2Tree converts long CoT into hierarchical tree structures to analyse the reasoning patterns of LRMs (Jiang et al., 2025). The framework classifies thoughts into types of reasoning, but their framework is applied after the generation, and is not applied dynamically. Further, Venhoff et al. (2025) derive a taxonomy of reasoning behaviors through trained Sparse Auto-Encoders on step activations, clustering them to identify common behaviors. While their approach results in a rich taxonomy, their technique requires full-access to the model's internals. Yu et al. (2026) also note that *"the high-level semantic roles of reasoning steps and their transitions remain underexplored."*. As a result, existing works often overlook the question of how to dynamically monitor LRMs reasoning during single inferences from text alone, and apply such monitoring for early-stopping purposes.

3. Step-Tagging module

In this section, we introduce *Step-Tagging*, a lightweight module capable of identifying, and tagging reasoning steps in real-time during inference.

Step-by-step generation. To enable fine-grained monitoring of LRMs, we decompose the reasoning traces into step sequences. Let $y = y_{1:n} \in V^n$ be the output token sequence generated by the model over the vocabulary V. We segmented LRM's outputs as follows:

$$S = \{s_1, \ldots, s_i, \ldots, s_T\}, \text{ such as } s_i = y_{r_{i-1}:r_i} \quad (1)$$

where r_i corresponds to the delimiters of the step segments. In this paper, we segmented the reasoning traces based on the delimiter $\alpha =$ `"\n\n"` (Zhang et al., 2025).

Objective. Our definition of a reasoning step enables users to segment reasoning steps within model outputs. However, this definition alone does not allow the user to annotate the segmented steps with reasoning types. This annotation would enable users to track logical transitions within model outputs. To do this, we must first define a tag dictionary $\mathcal{T}_{\text{tags}}$ (i.e., a label space of reasoning step tags) that covers the types of reasoning steps generated by models. Essentially, given a sequence of reasoning steps S, we wish to label each step s_i with a tag $\tau_i \in \mathcal{T}_{\text{tags}}$. Formally, we are looking to construct a step-tagging function ϕ such as:

$$\forall\, i \in [1, T], \phi(s_i) = \tau_i \quad (2)$$

where $s_i \in S$ is a reasoning step from the full output sequence y, ϕ is the step-tagging function, and $\tau_i \in \mathcal{T}_{tags}$ is the reasoning tag associated to the step s_i.

Taxonomy of the type of steps. To enable fine-grained monitoring of reasoning behavior, we need to know the different types of reasoning steps that are typically generated by LRMs (i.e., we need to define $\mathcal{T}_{\text{tags}}$). To do so, we created a taxonomy based on the outputs of both DeepSeek-R1-Distill-Llama-8B (DeepSeek-AI, 2025) and QwQ-32B (Team, 2025) models.

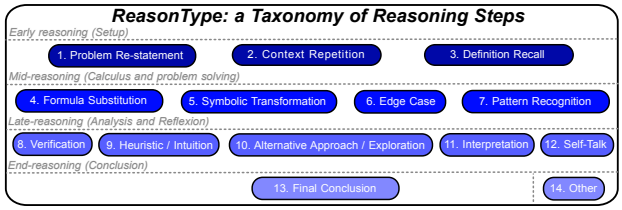


Figure 2. ReasonType - A taxonomy of reasoning step types as per `gpt-4o-mini`

Inspired by prior work on model behavior analysis (Galichin et al., 2025; Kuznetsov et al., 2025), we first created a prompt to identify distinct types of reasoning steps in the traces. We then sampled 100 reasoning traces from the MATH500 train dataset (covering 20 samples per difficulty level for each model) and using our prompt submitted the traces to `GPT-4o-mini` (OpenAI, 2024). The prompt resulted in a series of different step-types. We merged overlapping categories, to construct a taxonomy that reflects the temporal and reasoning progression of model's traces. We refer to this taxonomy as *ReasonType* (Figure 2) encompassing 13 categories, including early-stage behaviors such as *Problem Re-statement*, or later reasoning stages like *Verification* and *Exploration*. We detailed our construction and validation process in Appendix B.

Early-stopping criteria. In the following section, we see that LRMs tend to generate the answer early in the output sequence, with step-types following an ordered pattern. Based on this observation, the central challenge that we address is to determine when to stop the generation of LRMs based on step tags, creating an interpretable stopping criterion. Assuming that our Step-Tagging framework can effectively monitor the steps (Equation 2), we can define a constraint on the frequency of a given step type. Each constraint operates online, over a running sequence of reasoning steps $S_{\text{running}} = \{s_1, \ldots, s_j\}$, where each step s_i is associated with a tag $\tau_i \in \mathcal{T}$. We define the constraint c_{τ^*} as:

$$c_{\tau^*}(S_{\text{running}}, \delta) = \mathbf{1}[f_{\text{freq}}(S_{\text{running}}, \tau^*) \leq \delta] \tag{3}$$

$$\text{with } f_{\text{freq}}(S_{\text{running}}, \tau^*) = \sum_{i=1}^{j} \mathbf{1}[\tau_i = \tau^*]$$

where $c_{\tau^*}(S_{\text{running}}, \delta)$ is the constraint on the tag type τ^* over the step-sequence S_{running} being generated, given the threshold δ. $f_{\text{freq}}(S_{\text{running}}, \tau^*)$ is the occurrence of the type-step τ^* over the running sequence S_{running}. While the constraint c_{τ^*} is satisfied, the generation continues. If the constraint is violated, the generation stops (see Algorithm 1 in Appendix C for more implementation details).

To facilitate the evaluation of early-exit answers, we prompted the models right after early-stopping, and allowed an additional budget of 100 tokens. We borrowed this approach from Muennighoff et al. (2025), who showed that this intervention helped the model to provide its current best answer.

4. Experimental Setting

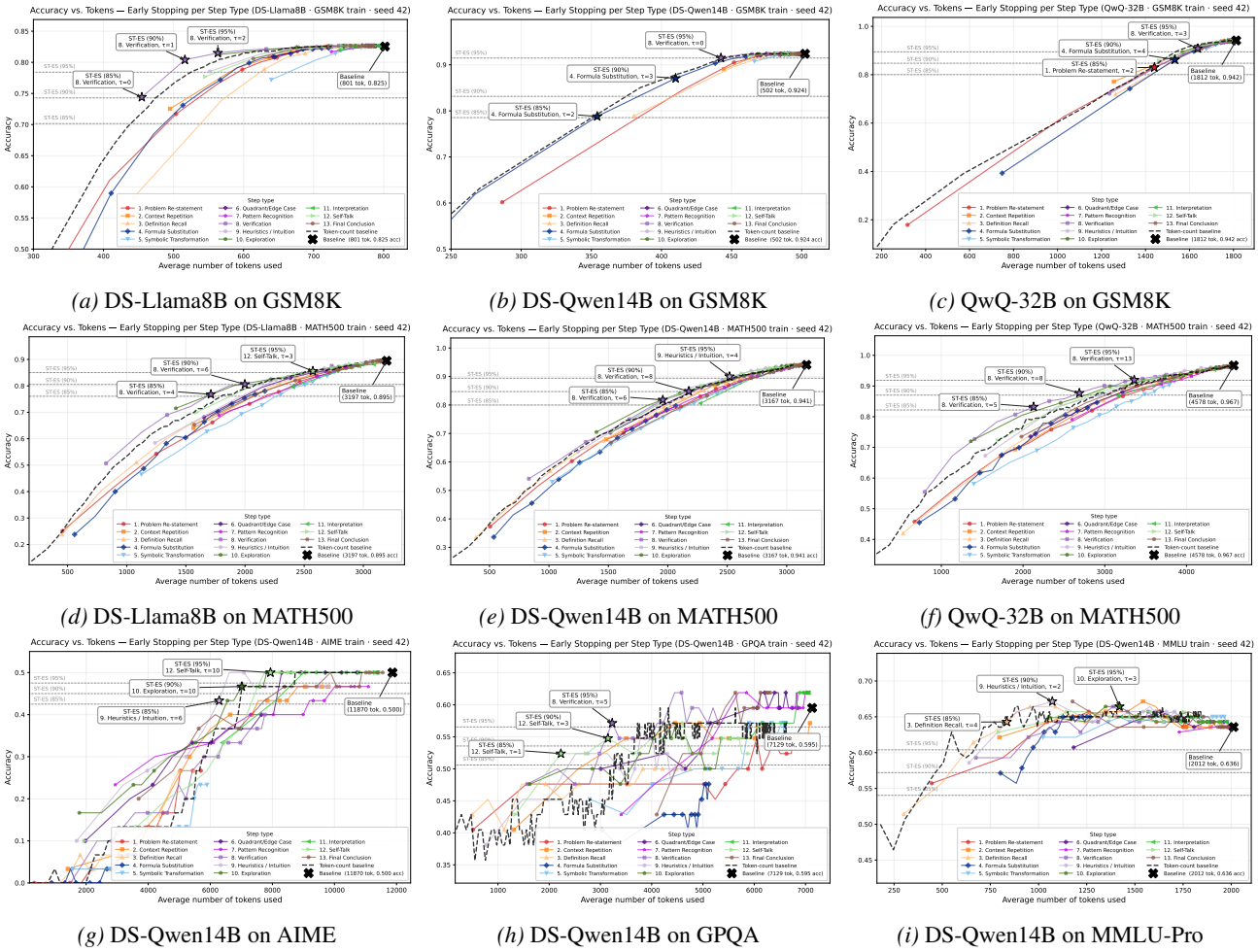
Our paper contains two objectives. First, our goal is to compare the contribution of each step-types to the efficiency of LRMs, and identify the most reliable early-stopping configurations. Second, we assess whether the ST-ES framework can be applied during deployment. In this section, we will first motivate our selection of models, datasets, and metrics, followed by our baseline implementation.

Datasets and metrics. To assess our approach, we selected five state-of-the-art reasoning datasets. First, we evaluated our models on three mathematical datasets, namely: *GSM8K*, *MATH500*, and *AIME*. Second, we evaluated the applicability of our framework on different domains, with *GPQA-Diamond* and *MMLU-Pro*. We evaluated the correctness of the answers using Math-Verify (mathematic tasks) or MCQ-Prompting (knowledge and reasoning benchmarks). Our analysis in Section 5 and the training of our step-tagger (Section 6) is conducted on the *training splits*. We then evaluated ST-ES in Section 7 on the held-out *testing splits*. Table 5 in Appendix D presents the datasets that we selected.

Model selection. To apply our framework a user must have access to the fine-grained reasoning traces of LRMs. Open-source models like DeepSeek-R1 and QwQ consistently provide reasoning traces. For this reason, we focus our analysis exclusively on `DeepSeek-R1-Distill-Llama-8B`, `DeepSeek-R1-Distill-Qwen-14B` and `QwQ-32B`, which offer the granularity needed to monitor the reasoning process. This choice is motivated by their variety in term of size and performance, and diversity in providers.

Inference setting. For the purposes of our experiments, we performed standard inference and applied our *Step-Tagging* and *Early-Stopping* algorithms *offline*. To ensure the robustness and reproducibility of our approach, we generated outputs using fixed random seeds with deterministic decoding. We used five seeds for GSM8K and MATH500 across all three model sizes (8B, 14B, 32B), and we used a single seed on AIME, GPQA and MMLU on the 14B model.

Baselines. To assess the effectiveness of our early-stopping approach, we define two baselines. First, we introduce a *Token-count* baselines, which early-stops the generation for a given count of step (regardless of their step-type) (Pu et al., 2025). Second, we set-up *Prompt-guided* baselines, which explicitly instruct the models to not generate verbose output, or over-verification steps (Lee et al., 2025). We implemented user-prompt and system-prompt variants, with Zero-Shot and Few-Shot prompts that aim to reduce the reasoning computation while retaining accuracy. We selected 4 variants, namely: zero-shot user and system prompt, and few-shot system prompt with 1 and 3 examples: $\mathcal{P}^{(0)}_{\text{user}}$, $\mathcal{P}^{(0)}_{\text{system}}$, $\mathcal{P}^{(1)}_{\text{system}}$, $\mathcal{P}^{(3)}_{\text{system}}$, respectively. The prompts used to establish these baselines are listed in Appendix D.



(a) DS-Llama8B on GSM8K *(b)* DS-Qwen14B on GSM8K *(c)* QwQ-32B on GSM8K

(d) DS-Llama8B on MATH500 *(e)* DS-Qwen14B on MATH500 *(f)* QwQ-32B on MATH500

(g) DS-Qwen14B on AIME *(h)* DS-Qwen14B on GPQA *(i)* DS-Qwen14B on MMLU-Pro

Figure 3. Number of Tokens vs. Pass@1 - ST-ES criteria on train datasets for $\delta \in [0, 20]$ - Each curve represent an ST-ES criteria with annotated step-types by `GPT-4o-mini`. We note that the efficiency curves differs depending on the step type. ST-ES criteria based on verification or self-reflection steps tends to offer the best accuracy trade-offs, almost systematically beating the token-count baseline.

5. Influence of step-types on LRM's efficiency

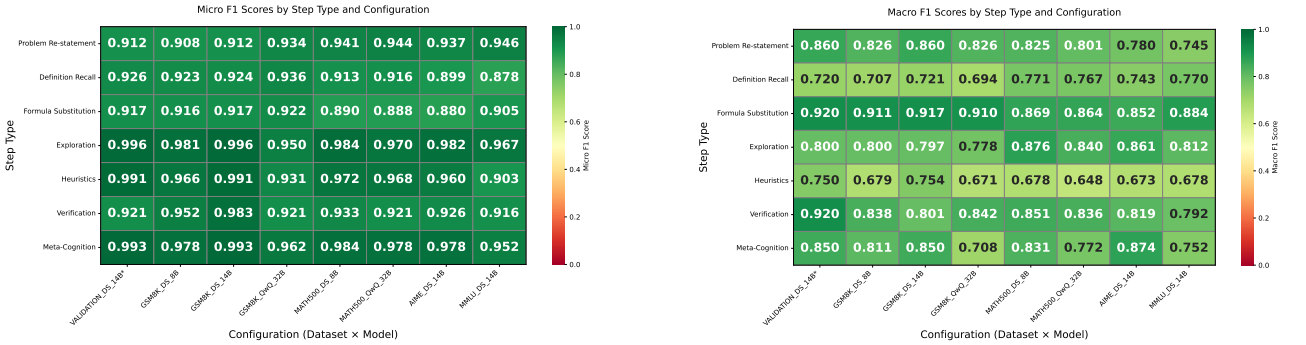
Our first objective is to determine whether tracking the type of reasoning steps is useful to early-stop the generation of LRMs. To do so, we first inferred models on the train split of the dataset, annotated their reasoning steps using `GPT-4o-mini`, and applied ST-ES. Figure 3 presents the number of tokens vs. accuracy of every tag-type with values of threshold ranging from 0 to 20, for each configurations. Each colored curve represent our ST-ES criterion applied to a specific step-type. The doted black curve represent the Token-Count baseline (for $\delta \in [1, 100 - 500]$).

Step-type, a useful signal. First, we observe that the ST-ES curves systematically matches or outperforms the Token-Count baselines for all configurations. For instance, in Figures 3d-3f (MATH500), at least one curve stands above the black doted line. It means that for equivalent token-count, ST-ES achieves higher accuracy. This observation holds across the three models and other datasets.

Step-types of ST-ES. Second, each step-type offer a distinct accuracy/token-count trade-offs. This is particularly visible on GSM8K (Figure 3a), and MATH500 (Figures 3d-3f), where late reasoning step-types such as *Verification* or *Self-Talk* stand above early step-types (e.g. *Problem Re-Statement*). For instance, in Figure 3e, *Verification* with $\tau = 5$ reaches 85% of the accuracy with only 50% of the token-count. Results are less pronounced on AIME and GPQA (Figures 3g and 3h), where curves are noisier - likely due to a lower number of sample and higher token-count relatively to other datasets, making the analysis less stable. Further, we selected three best inference configuration corresponding to expected accuracy drop (namely 85, 90 and 95%), and evaluate them on test datasets.

6. Monitoring LRMs using Step-Tagger

To validate our approach, we first train sentence classifiers for efficient step-type monitoring, and then evaluate our criteria on test datasets.



(a) Micro-F1 *(b)* Macro-F1

Figure 4. Step-Tagger performance - Seed 42 - Figures 4a and 4b present the Micro and Macro F1 score of the binary BERT classifiers across selected step-types and model/dataset configurations. The first column of each heatmaps report the performance on the held-out validation set (20% of the training set), composed of steps generated by DS-14B on MATH500 and GPQA train.

Training data generation. Given that our reasoning step taxonomy was created using `GPT-4o-mini` (OpenAI, 2024) the most direct way to label a reasoning trace would be to use `GPT-4o-mini`. However, this annotation is costly, each step requiring more than a second to be annotated (see Table 3 in Appendix B.2). Consequently, instead, we used `GPT-4o-mini` to label a dataset of reasoning traces with the labels from the taxonomy that we use to train lighter weight reasoning step classifiers. We constructed training datasets by running each LRMs on training samples of each datasets (with a seed of 42). For each step s_i in generated outputs, we prompted GPT-4o-mini to assign a tag τ_i. Appendix B.3 validates the reliability of GPT-4o-mini to annotate the reasoning steps.

Sentence classifiers. We selected the `bert-base-uncased` sentence classifier (Devlin et al., 2019) to construct our Step-Tagging framework, including a single hidden layer. Given the large and fine-grained nature of our taxonomy (13 distinct step types), training a multi-class classifier is challenging due to significant class imbalance. To address this, we trained separate binary classifiers for each step-type. This approach notably improved detection accuracy across low-frequency categories, and fits our definition of early-stopping constraint: one step-type per early-stopping criteria. To reduce the training cost, and evaluate the generalization of our apporach, we train our lightweight modules on MATH500 and GPQA train split obtained on DS-Qwen14B, and evaluate their performance on other configurations. The training details are presented in Appendix E.

Classification metrics. To evaluate the performance of our classifiers, we computed the Macro-F1 and Micro-F1 on the test datasets. While the *Macro-F1* helps to identify the classifier's ability to detect rare classes, the *Micro-F1* offers a more global view on the step detector's performance across all steps.

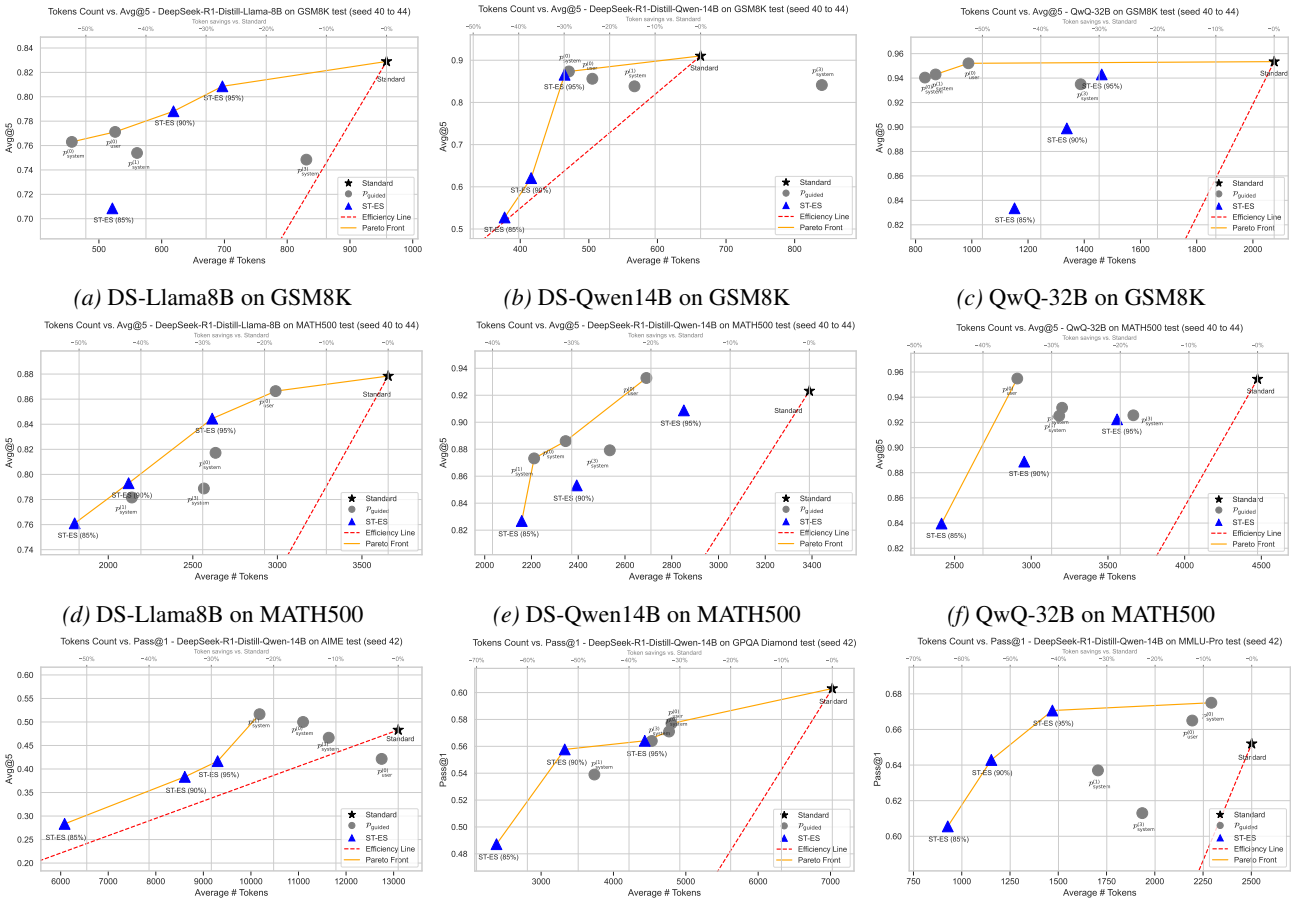
Performance of step monitoring. Next, our objective is to demonstrate that our approach can effectively identify the step-types in reasoning traces. Figure 4 present the performance of the binary step-classifiers on the seven selected step-types for every configurations. We observe that the Micro-F1 is generally high across most steps for all models across all datasets - 0.88 to 0.99, which demonstrates that the classifiers are good at detecting step-tags. Moreover, we also reported the macro-F1 score since the distribution of step-types is highly imbalanced.

We observe lower scores, notably for *Heuristics* with 0.69 Macro-F1 on average (*Heuristics* is a rare step type, representing less than 1.5% of the labels, and so we attribute this relatively low score to label imbalance). However, the scores remain relatively high, particularly for *Verification* and *Formula Substitution*. Importantly, we note that results on unseen configurations (other dataset and model configuration, i.e. DS-Llama8B, QwQ-32B, GSM8K, AIME and MMLU) remains satifying. It means that our step-taggers are transferable, and are still able to accurately identifying a given step-types for other models and tasks.

We interpret the strong performance of the classifiers as validating our reasoning step taxonomy in the sense that it indicates that the step types are distinct (i.e., they reflect types with separable properties).

7. Step-Tagging Early-stopping (ST-ES)

Next, we show in this section that Step-Tagging modules can effectively be used as an early-stopping criteria. Figure 5 presents the average token count against the accuracy for the selected configurations. Each plot compares the performance trade-offs between the prompting baselines and the ST-ES criteria. Tables 6 and 7 in the Appendix H report the quantitative metrics of the baselines and our approach on the three models, and five datasets.



(a) DS-Llama8B on GSM8K *(b)* DS-Qwen14B on GSM8K *(c)* QwQ-32B on GSM8K

(d) DS-Llama8B on MATH500 *(e)* DS-Qwen14B on MATH500 *(f)* QwQ-32B on MATH500

(g) DS-Qwen14B on AIME *(h)* DS-Qwen14B on GPQA *(i)* DS-Qwen14B on MMLU-Pro

Figure 5. Number of Tokens vs. Avg@5 and Pass@1 - $\mathcal{P}_{\text{guided}}$ Baselines vs. ST-ES criteria on test datasets - The efficiency lines in red highlight the configurations that improve the efficiency relative to the standard inference, while the Pareto frontiers in yellow show the most efficient approaches. ST-ES achieved up to $20-50\%$ token-count saving with minimal accuracy loss (from +2% to -12% accuracy).

$\mathcal{P}_{\text{guided}}$ baselines. We first notice that simple instruction on the models results in strong token-reduction, achieving 20% to 60% saved tokens across configurations. Specifically, the baselines achieve much better results on QwQ-32B, and the system-prompt variants generally lead to more token-reduction for the Deepseek models.

Performance of ST-ES. Next, we observe that our ST-ES criteria achieved more efficient generation, with all ES-ST settings lying on the left side of the *efficiency line* (in red) compared to the standard inference for all models. Further, ST-ES outperforms most $\mathcal{P}_{\text{guided}}$ baselines for both Deepseek models. Our ST-ES criteria is performing well on the DS-Llama8B model on both GSM8K and MATH500 since almost all ST-ES configurations lies on the Pareto front. For instance, on MATH500 (Figure 5d), ST-ES 85% achieved approximately the same token reduction as $\mathcal{P}^{(0)}_{\text{system}}$ and $\mathcal{P}^{(3)}_{\text{system}}$ (around 30%), while achieving higher accuracy.

Generalization to other tasks. To assess robustness beyond MATH500 and GSM8K dataset, Figures 5g-5i show the performance of ST-ES on AIME, GPQA-Diamond and MMLU-Pro on DS-Qwen14B. Our results indicate that ST-ES scales well on more complex tasks. All ST-ES lies on the Pareto front, offering 30 to 55% token reduction with +2% to -5% accuracy for moderate configurations. Moreover, it shows that ST-ES remains effective across domains.

Robustness of ST-ES. Appendix F shows that the training cost of ST-ES are fully recovered by the efficiency gains obtained. As well, Appendix G shows that our trained tagging modules approximates well the GPT-4o-mini annotation.

8. Conclusion

This work offers a novel view on both monitoring and efficiency of LRMs. We demonstrate that users can effectively track the reasoning flow of LRMs using our *Step-Tagging* framework, paving the way for more work on the monitoring of reasoning steps. Furthermore, we show that tracking the step-type in the generation of LRMs can lead to reliable early-stopping criterion. Our framework can enhance the control of the generation of RLMs enabling a significant token-count saving (20-50%) while preserving performance.

Acknowledgment

This work has been partially supported by the 6G-XCEL project (grant agreement 101139194), funded by the EU Horizon Europe program.

This research was partly supported by the ADAPT Research Centre. The ADAPT Centre for Digital Content Technology is funded under the Research Ireland's Research Centres Programme (Grant 13/RC/2106 II) and is co-funded under the European Regional Development Funds.

References

Aytes, S. A., Baek, J., and Hwang, S. J. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching, 2025. URL https://arxiv.org/abs/2503.05179.

Badshah, S. and Sajjad, H. Reference-guided verdict: Llms-as-judges in automatic evaluation of free-form qa, 2025. URL https://arxiv.org/abs/2408.09235.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Chen, Q., Qin, L., Liu, J., Peng, D., Guan, J., Wang, P., Hu, M., Zhou, Y., Gao, T., and Che, W. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models, 2025a. URL https://arxiv.org/abs/2503.09567.

Chen, X., Xu, J., Liang, T., He, Z., Pang, J., Yu, D., Song, L., Liu, Q., Zhou, M., Zhang, Z., Wang, R., Tu, Z., Mi, H., and Yu, D. Do not think that much for 2+3=? on the overthinking of o1-like llms, 2025b. URL https://arxiv.org/abs/2412.21187.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Galichin, A., Dontsov, A., Druzhinina, P., Razzhigaev, A., Rogov, O. Y., Tutubalina, E., and Oseledets, I. I have covered all the bases here: Interpreting reasoning features in large language models via sparse autoencoders, 2025. URL https://arxiv.org/abs/2503.18878.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Golovneva, O., Chen, M., Poff, S., Corredor, M., Zettlemoyer, L., Fazel-Zarandi, M., and Celikyilmaz, A. Roscoe: A suite of metrics for scoring step-by-step reasoning, 2023. URL https://arxiv.org/abs/2212.07919.

Han, T., Wang, Z., Fang, C., Zhao, S., Ma, S., and Chen, Z. Token-budget-aware llm reasoning, 2025. URL https://arxiv.org/abs/2412.18547.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Huang, Y., Zeng, W., Zeng, X., Zhu, Q., and He, J. From accuracy to robustness: A study of rule- and model-based verifiers in mathematical reasoning, 2025. URL https://arxiv.org/abs/2505.22203.

Jiang, G., Liu, Y., Li, Z., Bi, W., Zhang, F., Song, L., Wei, Y., and Lian, D. What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6490–6514, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.329. URL https://aclanthology.org/2025.emnlp-main.329/.

Kuznetsov, K., Kushnareva, L., Druzhinina, P., Razzhigaev, A., Voznyuk, A., Piontkovskaya, I., Burnaev, E., and

Barannikov, S. Feature-level insights into artificial text detection with sparse autoencoders, 2025. URL https://arxiv.org/abs/2503.03601.

Lee, A., Che, E., and Peng, T. How well do llms compress their own chain-of-thought? a token complexity approach, 2025. URL https://arxiv.org/abs/2503.01141.

Lee, J. and Hockenmaier, J. Evaluating step-by-step reasoning traces: A survey, 2025. URL https://arxiv.org/abs/2502.12289.

Li, Z.-Z., Zhang, D., Zhang, M.-L., Zhang, J., Liu, Z., Yao, Y., Xu, H., Zheng, J., Wang, P.-J., Chen, X., Zhang, Y., Yin, F., Dong, J., Li, Z., Bi, B.-L., Mei, L.-R., Fang, J., Liang, X., Guo, Z., Song, L., and Liu, C.-L. From system 1 to system 2: A survey of reasoning large language models, 2025. URL https://arxiv.org/abs/2502.17419.

Marjanović, S. V., Patel, A., Adlakha, V., Aghajohari, M., BehnamGhader, P., Bhatia, M., Khandelwal, A., Kraft, A., Krojer, B., Lù, X. H., Meade, N., Shin, D., Kazemnejad, A., Kamath, G., Mosbach, M., Stańczak, K., and Reddy, S. Deepseek-r1 thoughtology: Let's think about llm reasoning, 2026. URL https://arxiv.org/abs/2504.07128.

Minegishi, G., Furuta, H., Kojima, T., Iwasawa, Y., and Matsuo, Y. Topology of reasoning: Understanding large reasoning models through reasoning graph properties, 2025. URL https://arxiv.org/abs/2506.05744.

Moons, F. and Vandervieren, E. Measuring agreement among several raters classifying subjects into one or more (hierarchical) categories: A generalization of fleiss' kappa. *Behavior Research Methods*, 57(10), 2025. ISSN 1554-3528. doi: 10.3758/s13428-025-02746-8. URL http://dx.doi.org/10.3758/s13428-025-02746-8.

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Oh, G., Goel, R., Hidey, C., Paul, S., Gupta, A., Shah, P., and Shah, R. Improving top-k decoding for non-autoregressive semantic parsing via intent conditioning, 2022. URL https://arxiv.org/abs/2204.06748.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. Routellm: Learning to route llms with preference data, 2025. URL https://arxiv.org/abs/2406.18665.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Project-Numina. Ai-mo/aimo-validation-aime, 2025. URL https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

Pu, X., Saxon, M., Hua, W., and Wang, W. Y. Thought-terminator: Benchmarking, calibrating, and mitigating overthinking in reasoning models, 2025. URL https://arxiv.org/abs/2504.13367.

Qu, X., Li, Y., Su, Z.-C., Sun, W., Yan, J., Liu, D., Cui, G., Liu, D., Liang, S., He, J., Li, P., Wei, W., Shao, J., Lu, C., Zhang, Y., Hua, X.-S., Zhou, B., and Cheng, Y. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond, 2025. URL https://arxiv.org/abs/2503.21614.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof qa benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Sui, Y., Chuang, Y.-N., Wang, G., Zhang, J., Zhang, T., Yuan, J., Liu, H., Wen, A., Zhong, S., Zou, N., Chen, H., and Hu, X. Stop overthinking: A survey on efficient reasoning for large language models, 2025. URL https://arxiv.org/abs/2503.16419.

Team, Q. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Venhoff, C., Arcuschin, I., Torr, P., Conmy, A., and Nanda, N. Base models know how to reason, thinking models learn when, 2025. URL https://arxiv.org/abs/2510.07364.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Wang, X., McInerney, J., Wang, L., and Kallus, N. Entropy after ¡/think¿ for reasoning model early exiting, 2026. URL https://arxiv.org/abs/2509.26522.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., and Chen, W. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Xu, F., Hao, Q., Zong, Z., Wang, J., Zhang, Y., Wang, J., Lan, X., Gong, J., Ouyang, T., Meng, F., Shao, C., Yan, Y., Yang, Q., Song, Y., Ren, S., Hu, X., Li, Y., Feng, J., Gao, C., and Li, Y. Towards large reasoning models: A survey of reinforced reasoning with large language models, 2025a. URL https://arxiv.org/abs/2501.09686.

Xu, S., Xie, W., Zhao, L., and He, P. Chain of draft: Thinking faster by writing less, 2025b. URL https://arxiv.org/abs/2502.18600.

Yang, C., Si, Q., Duan, Y., Zhu, Z., Zhu, C., Li, Q., Chen, M., Lin, Z., and Wang, W. Dynamic early exit in reasoning models, 2025. URL https://arxiv.org/abs/2504.15895.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W.-Y., Zhang, Y.-Q., Yan, L., Qiao, M., Wu, Y., and Wang, M. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yu, S., Xiong, Y., Wu, J., Li, X., Yu, T., Chen, X., Sinha, R., Shang, J., and McAuley, J. Explainable chain-of-thought reasoning: An empirical analysis on state-aware reasoning dynamics, 2026. URL https://arxiv.org/abs/2509.00190.

Zhang, A., Chen, Y., Pan, J., Zhao, C., Panda, A., Li, J., and He, H. Reasoning models know when they're right: Probing hidden states for self-verification, 2025. URL https://arxiv.org/abs/2504.05419.

LLM Usage

We acknowledge the use of Large Language Models for the purpose of our experimentation in our paper. Specifically, as stated in Section 3, we relied on `GPT-4o-mini` to set our *ReasonType* taxonomy. This approach is borrowed from work on behavior analysis of LLMs, such as Galichin et al. (2025); Kuznetsov et al. (2025).

Appendix

A. Future work

While our ST-ES framework offers significant efficiency gains while maintaining the model's performance, a few limitations should be acknowledge. First, we introduced the *ReasonType* taxonomy in Section 6, and demonstrate though experiments and ablations study that this taxonomy enable meaningful dynamic step-level monitoring of LRM generation. However, future work should focus on applying ST-ES on other taxonomies (Galichin et al., 2025; Marjanović et al., 2026; Minegishi et al., 2025; Venhoff et al., 2025).

Further, ST-ES requires calibration to find the most adapted inference configurations (step-types and value of δ). We found across datasets and models that Verification and self-reflection steps with values of $\delta < 10$ are offering satisfying efficiency trade-offs. And we found that ST-ES is more practical than token-count baselines. Yet, the models are sensible to the value of δ. Future work should focus on experimenting dynamic values of δ to make the criteria even more agnostic.

B. Construction and Validation of the ReasonType taxonomy

This appendix presents the methodology that we adopted to create the ReasonType taxonomy. Importantly, this section demonstrates the robustness of our annotation approach using our taxonomy, further validating our overall methodology.

B.1. Construction of the taxonomy

In Section 3, we provided a high-level description of our methodology to generate a taxonomy of reasoning step. In this section, we will describe our methodology in more details.

Methodology. Our goal in constructing the *ReasonType* taxonomy is to observe the behavior of LRMs and look at aggregating similar types together to follow the generation of LRMs closely. Because no fine-grained taxonomy of reasoning steps exists, we used an open-ended labeling procedure: we prompted a model to generate free-form step-type labels, and manually merged common labels together. We took inspiration from previous work, who relied on the summarization and behavior detection capabilities of strong models such as `GPT-4o-mini`, as this method has proven to be effective (Galichin et al., 2025; Kuznetsov et al., 2025).

Dataset	Model	# Steps	# Unique Tags
MATH500	`DeepSeek-R1-Distill-Llama-8B`	3,840	162
	`QwQ-32B`	3,057	179

Table 1. Step for taxonomy generation

We first generated 100 reasoning traces from the MATH500 train dataset, using `DeepSeek-R1-Distill-Llama-8B` and `QwQ-32B`, to obtain a pool of reasoning steps (20 samples from each complexity level). We obtained a pool of $6,897$ reasoning steps (see Table 1). Each step is then passed to `GPT-4o-mini` using our Taxonomy prompt, presented in Figure 6 below. We obtained an open-ended label for each step-type.

Taxonomy Prompt

Below is a reasoning trace of a reasoning language model, split by steps. In these examples, can you please identify the different type of steps? Suggest some reasoning-type labels for each of them.
- Step 1: {step_1}
- **[...]**
- Step t: {step_t}

Figure 6. Prompt used to generate the Taxonomy

Figure 7 presents the most frequent labels generated. We observe that, even though our prompting was open-ended (without particular instruction for the model to generate certain type of labels), some labels were obtained frequently, such as *Problem Identification/Re-Statement*, *Substitution*, or *Verification*. We manually inspected the labels obtained, and merged labels that have a common signification. Table 2 illustrates several examples.

Takeaways. While the ReasonType taxonomy might not be optimal, we showed that it reflects the different steps of the model, and offers enough consistency and granularity to analyze the reasoning process of LRMs and to train accurate classifiers for our study. It also echoes previous work, and the categories overlaps with existing taxonomies (Galichin et al., 2025; Marjanović et al., 2026; Minegishi et al., 2025; Venhoff et al., 2025). Furthermore, the size of our taxonomy (13-14 step-types) is supported by the finding of Venhoff et al. (2025): *"we find [...] that reasoning mechanisms are reasonably well represented using 10 to 20 categories."*.

Limitations. To over-come the domain dependency, we could rely on the OpenThoughts-114k dataset. Indeed, it seems to provide a diverse set of possible behavior for the models. However, it would come at a high processing cost.

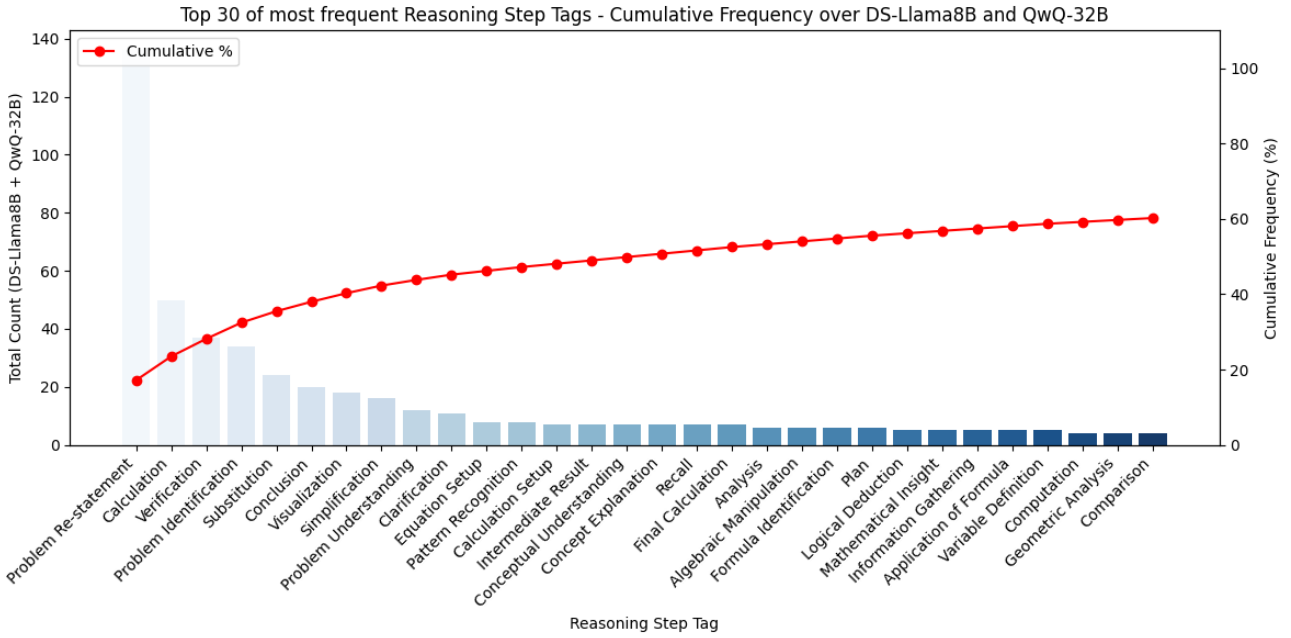


Figure 7. Most frequent labels obtained from open-end label generation

Tags from ReasonType	`DeepSeek-R1-Distill-Llama-8B`	**QwQ-32B**
Problem Re-statement / Setup	Problem Re-statement, Problem Identification, Clarification	Problem Re-statement, Coordinate Setup, Step-by-Step Breakdown
Definition Recall	Recall of Relevant Concepts, Definition Recall	Recall, Rule Recall, Definition Explanation
Formula Substitution	Value Substitution, Application of Formula, Calculation	Equation Setup, Equation Manipulation, Application of Formula
Exploration	Approach Exploration, Exploration of Alternatives	Exploration
Self-Talk	Procedure Explanation, Plan of Action, Comparison of Options	Confirmation, Reflection, Logical Breakdown
Verification	Confirmation, Verification	Backward Calculation, Example Verification, Assumption Checking
Final Answer	Final Evaluation, Final Calculation	Final Evaluation, Final Calculation

Table 2. Example of categories from the ReasonType Taxonmy obtained by merging labels from the annotation - each set of labels were generated using our prompt and sampled reasoning steps, and we manually merged common labels to create the ReasonType taxonomy.

B.2. Labeling the reasoning traces

Table 3 presents statistics on the number of steps and `GPT-4o-mini` annotation for each models on the selected datasets.

Model	Dataset	# Tok. / Steps	# Steps / Sample	# Steps	Runtime / Steps	Runtime / Sample	Total Runtime
`DS-Llama-8B`	GSM8K	34.58	23.64	70,911	0.87	20.71	62,143
	MATH500	34.11	96.27	96,270	0.88	84.86	84,861
`DS-Qwen-14B`	GSM8K	36.64	14.01	42,017	0.86	12.09	36,293
	MATH500	34.33	94.84	94,838	0.85	80.26	80,263
	AIME 22-24	27.74	469.29	42,236	0.85	398.68	35,881
	GPQA-Diamond	38.96	186.51	36,928	0.84	156.05	30,898
	MMLU-Pro	35.86	64.59	90,428	0.87	56.42	78,985
`QwQ-32B`	GSM8K	45.59	40.43	121,288	0.87	35.33	105,977
	MATH500	38.97	117.28	117,283	0.83	97.49	97,494

Table 3. Avg. # of steps and annotation runtime per sample - MATH500 and GSM8K are the trained dataset ($1,000$ and $3,000$ samples, respectively). AIME, GPQA and MMLU are the full dataset (90, 198 and $1,400$ samples, respectively). Runtime in seconds.

B.3. Reliability of `GPT-4o-mini` as an annotator

Claim. Our framework rely on the annotation capability of the `GPT-4o-mini` model. Since the model is large and achieved great performance on a range of tasks, we assumed that the model is able to provide us with labels of good quality. In this section, we will observe and analyse the reliability of the annotation of the steps by the `GPT-4o-mini` model.

Methodology. To verify our claim, we sampled $1,000$ reasoning steps of DS-Qwen14B model from its inference on the MATH500 dataset. We then annotated each steps 5 times using `GPT-4o-mini`, and observe the agreement of each annotation. In addition, we also compared the annotation agreement between the `GPT-4o-mini` model, and 3 additional models: `GPT-4o` (a larger, closed-source model), `llama-3-3-70b-instruct`, and `Mixtral-8x22B-Instruct-v0.1` (both open-source and smaller relatively to the two model selected). This setup allows us to assess both the internal consistency of `GPT-4o-mini` and its alignment with other model annotators.

Experimental Design. We refer to self-model agreement as the *Inner-model agreement* (agreement between different runs of the same model on the same reasoning steps). The inner-model agreement is measured using the Fleiss' kappa metric. Indeed, the Fleiss' kappa measure the agreement between more than 2 annotators, making it suitable to compare many annotation trials (Moons & Vandervieren, 2025). Furthermore, we call agreement between two different model the *Inter-model agreement*. To compute this, we first selected the most consistent label generated across the different runs for each model, and we measured the Cohen's Kappa (Badshah & Sajjad, 2025). For this experiment, we used the OpenAI default decoding parameters for the GPT models. Same parameter was set for the other models selected.

Models	GPT-4o-mini	GPT-4o	Llama-3-3-70b	Mixtral-8x22B
GPT-4o-mini	**0.780**	0.601	0.457	0.392
GPT-4o		**0.799**	0.445	0.384
Llama-3-3-70b			**0.722**	0.398
Mixtral-8x22B				**0.587**

Table 4. Agregation metrics of the annotation process - $1,000$ samples reasoning steps from the `DeepSeek-R1-Distill-Qwen-14B` model on the MATH500 dataset - **Fleiss' kappa** in bold (diagonal - inner-model aggregation) - **Cohen's kappa** otherwise (inter-model aggregation)

Internal consistency of `GPT-4o-mini`. The Fleiss' Kappa score of 0.780 (see Table 4 in the diagonal) indicates high agreement among multiple independent runs of `GPT-4o-mini` on the same reasoning steps. This demonstrates that the model produces stable and consistent annotations when prompted repeatedly. We observe that the Fleiss' Kappa score seems to decrease as the model size becomes smaller.

`GPT-4o-mini` against other models. When comparing `GPT-4o-mini` with other models, we observe significant agreement with `GPT-4o`, with a Cohen's Kappa of 0.601, and moderate agreement with smaller models (0.457 with Llama70b and 0.392 with Mixtral). The higher agreement with `GPT-4o` suggests that larger models tend to produce more stable and higher-quality annotation, while smaller models exhibits more variability.

Takeaway. Overall, these results supports that `GPT-4o-mini` is a reliable annotator for reasoning steps. Indeed, for our annotation task, the model appears to be consistent accross multiple runs, and shows meaningful agreement with other strong models. The results obtained on smaller models confirms our decision of selecting `GPT-4o-mini` as a annotator to label the reasoning steps of LRMs.

B.4. Reason-Type, a robust taxonomy for identifying reasoning behaviors

Objective. This ablation study is looking at further validating our ReasonType taxonomy. In other words, we are investigating whether our proposed taxonomy captures meaningful distinctions in reasoning steps. We are looking to demonstrate that:

1. The ReasonType taxonomy enable semantic distinction of the type of reasoning.
2. Our annotation method with the `GPT-4o-mini` model, coupled with the ReasonType taxonomy, is a robust method to access to the ground-truth labels of the reasoning steps.

Methodology. To address our objective, we compare the performance of BERT classifiers across Original labels (OG - from `GPT-4o-mini` annotation using the ReasonType taxonomy), and shuffled labels for three step-types, namely: *Verification*, *Exploration* and *Self-Talk*. For the shuffled labels version, we took the exact same proportion of positive labels as in Original datasets, and used random shuffle with a seed of 42. Each experiment is run on the same training and held-out validation dataset, using the MATH500 training dataset on the DS-Qwen14B model. We trained binary step-taggers following the same training configuration (see Section 6). We set the label of selected class to 1, and put other labels to 0. To compare performances, we report both training loss, and classification metrics (precision and recall on both classes, along with macro and micro average.)

Evaluation. Figure 8 shows the training loss of the Original and Shuffled versions, for the three labels. Models trained on the Original labels presents significant lower losses, and are smoothly decreasing. It demonstrate that the Original datasets contains meaningful patterns between reasoning steps and their labels. In comparison, the models trained on shuffled labels present almost constant loss, relatively higher than the one from the Original labels.

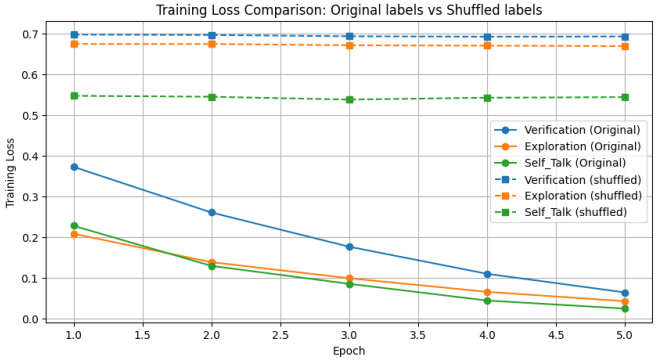
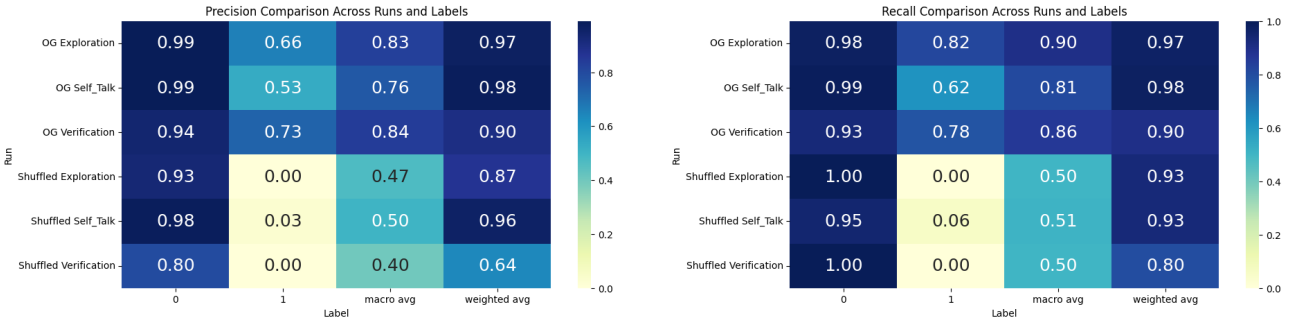


Figure 8. Training losses - ReasonType vs. Shuffled labels

Furthermore, Figures 9a and 9b show the Precision and Recall classification metrics on the testing dataset, respectively. For Original runs, both classes (0 and 1) achieve good performance despite dataset imbalancity, with Macro average Precision and Recall lying between 0.76 and 0.90 across labels. In comparison, shuffled runs presents poor results, with models failing in predicting positive classes - Precision and Recall of class 1 between 0.00 and 0.06. Along with the training loss, theses metrics highlight that the models trained on shuffles labels cannot learn meaningful relations between steps and labels. In comparison, Original labels (from the ReasonType taxonomy) resulted in satisfying model performance, and smooth training.



(a) Precision *(b)* Recall

Figure 9. Precision and Recall - ReasonType vs. Shuffled labels

Takeaway. Overall, these results comforts our finding that the ReasonType taxonomy labels enable annotation methods to results in reasoning steps carrying semantic meaning.

B.5. Generalizability of the ReasonType taxonomy

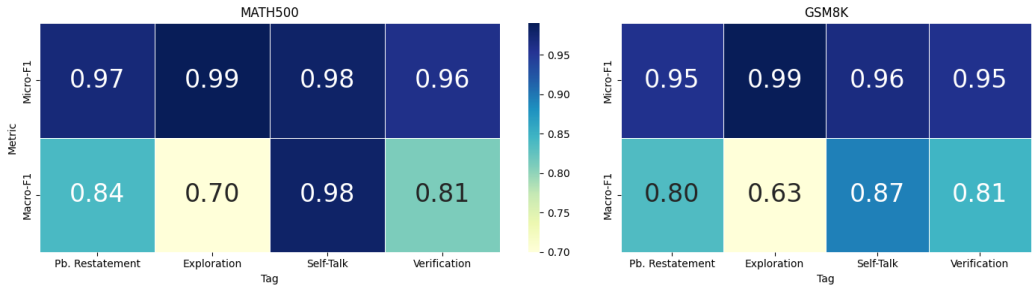
We proved in Appendix B.3 that `GPT-4o-mini` is a reliable annotator model for our taxonomy and the task of tagging the steps. However, the taxonomy is generated from specific samples of specific models. We already proved to some extend that the Taxonomy is generalizable. Indeed, we trained our step-tagging classifier on reasoning steps obtained using DS-Qwen14B, a model that was not used to derive the taxonomy (see Appendix B.1). Moreover, we obtained satisfying performance of our step-tagging classifier on models and datasets that were not used to for training, namely DS-Llama8B and QwQ-32B on MATH500 and GSM8K, and DS-Qwen14B on GSM8K, AIME and MMLU-Pro (see Section 6).

Claim. To further validate our taxonomy, we will test its applicability to 2 additional reasoning models. Our claim is that if the annotation process from `GPT-4o-mini` results in accurate training of step-tagging classifiers, it means that the ReasonType taxonomy is applicable to other models, therefore generalizable. To evaluate this, we trained binary BERT classifiers – similarly as Appendix B.4 – on four step-types, namely: *Problem Re-statement*, *Exploration*, *Self-Talk* and *Verification*.

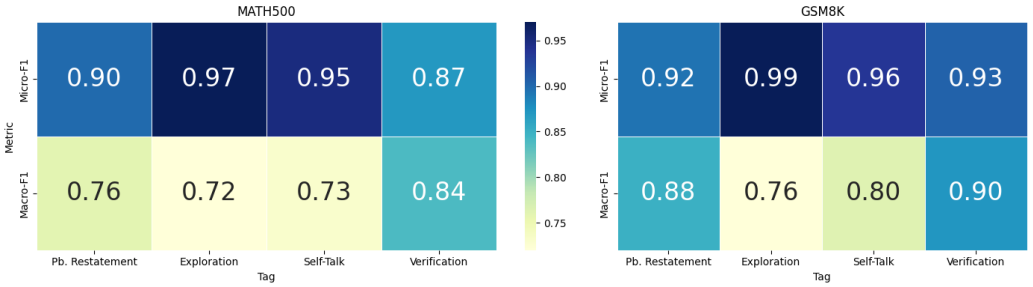
Methodology. To verify our claim, we inferred 2 additional models on both MATH500 and GSM8K, specifically `microsoft/Phi-4-reasoning` and `Qwen/Qwen3-30B-A3B-Thinking-2507` since they are both reasoning models, and comes from additional providers or training methods. We then trained 4 BERT classifiers for each model and dataset. We used 500 and $3,000$ samples from the MATH500 and GSM8K train datasets, respectively.

Performances of the Step-Taggers. We split the resulting annotated datasets following random 80:20 train/test split. Figure 10a and 10b show the micro-F1 and macro-F1 for the Phi-4 and Qwen3-30B-A3B models on MATH500 and GSM8K, respectively. We observe satifying performances of Step-Taggers on both models. Specifically, on the `Phi-4` model, we obtained between 0.7 to 0.98 and 0.63 to 0.87 macro-F1 on MATH500 and GSM8K datasets, respectively. The lower macro-F1 on Exploration can be explained by its low representation in the datasets (around 1% of labels in both datasets).

Similarly, `Qwen3-30B-A3B` obtained satifying perfromance on both MATH500 and GSM8K, with 0.72 to 0.84 and 0.76 to 0.90 macro-F1, respectively. Performances of both models are comparable to results obtained on the DS-Qwen14B using the original dataset (see Appendix B.4).



(a) Phi-4



(b) Qwen3-30B-A3B

Figure 10. Performance of Step-Taggers - seed 42

Takeaways. We interpret the satisfying performances of the Step-Taggers trained on other models as further validating the applicability of our taxonomy to other models. Indeed, these models were not used to create our *ReasonType* taxonomy, but still resulted in step-type identification performance of BERT classifiers trained on their reasoning traces using our methodology.

C. Step-Tagging Early-Stopping algorithm

Algorithm 1 lists the Step-Tagging Early-Stopping criteria. The user needs to define a constraint $\{\tau*, \delta\}$, and input a Binary Step-Tagger $\phi_{\tau*}$, which returns 1 if the step tag is τ^* and 0 otherwise. If the constraint breaks, the algorithm stops the generation, and prompts the model with $\mathcal{P}_{\text{exit}}$ to give the current best answer.

Algorithm 1 Step-Tagger Early-Stopping

Require: Prompt x; reasoning delimiter $\alpha \in V$; minimal step size $k \in \mathbb{N}$; max steps $T_{\max}$; Reasoning Language Model $\mathcal{M}$; tokenizer $\mathcal{T}$; EOS token γ; Constraint $\{\tau*, \delta\}$; Binary Step-Tagger ϕ_{τ^*}; Early-Exit Prompt $\mathcal{P}_{\text{exit}}$

$y \leftarrow \mathcal{T}(x)$ {Tokenize the input}
$S_{running} \leftarrow [\,]$; {Initialize output}
$t \leftarrow 0$
$f_{\tau^*} \leftarrow 0$ {Initialize frequency track of τ^*}
while $c_{\tau*}(S_{running}, \delta)$ **do** {Generate until constraint breaks}
    Generate step s_i using $\mathcal{M}$, α, where $|s_i| > k$
    $y \leftarrow s_i$
    if $\phi_{\tau^*}(s_i)$ **then** $f_{\tau^*} \leftarrow f_{\tau^*} + 1$ {Increase the counter}
    else
        Continue the generation
    end if
    $t \leftarrow t + 1$
end while
$y \leftarrow \mathcal{M}(y + \mathcal{P}_{\text{exit}})$ {Infer $\mathcal{M}$ with the early exit prompt}
return y
=0

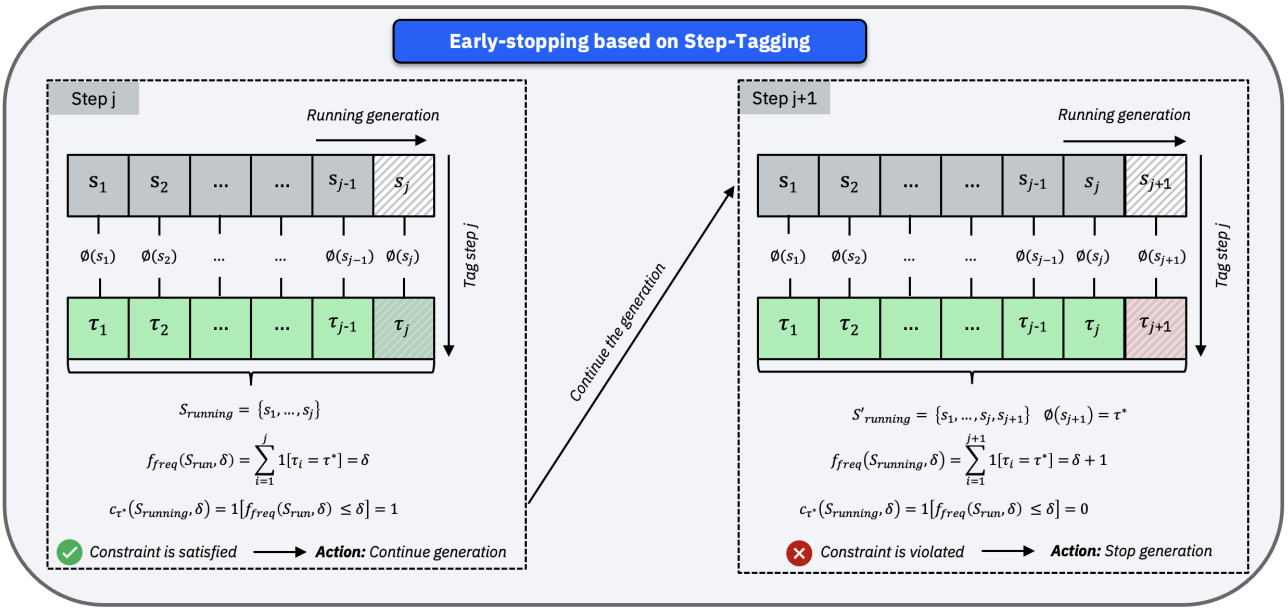


Figure 11. Illustration of early-stopping based on Step-Tagging

D. Experimental Design

Datasets. Table 5 summarizes the datasets and metrics we used to evaluate our selected reasoning models.

Domain	Ref.	Name	Train Size	Test Size	Type of evaluation	
					Math-Verify	MCQ Prompting
Maths	(Cobbe et al., 2021)	GSM8K	3,000	1,329	✓	
	(Hendrycks et al., 2021)	MATH500	1,000	500	✓	
	(Project-Numina, 2025)	AIME 22-24	30	60	✓	
In-domain	(Rein et al., 2023)	GPQA-Diamond	40	158		✓
	(Wang et al., 2024)	MMLU-Pro	240	1,200		✓

Table 5. Overview of selected reasoning datasets and metrics

Evaluating LRMs. To assess the model's performance on challenging reasoning tasks, the Avg@k, Pass@k, and Cons@k are common metrics (Chen et al., 2021; 2025a; Yu et al., 2025). The Pass@k measures the proportion of the samples where at least one of k attempts leads to the correct answer, while the Cons@k consider a sample correct if all k attempts are correct. Since we are interested about both performance and robustness of our approach, we selected the Avg@5, the Pass@5 and the Cons@5 as the quantitative metrics. Assessing the performance of LRMs on mathematical questions is challenging. This is due to the open nature of the question. For our experiments, we selected the *Math-Verify*[1] library which is a common metric to assess mathematical problems. It uses text extraction and formal verification. This metric also reported strong correctness compared to other evaluation methods such as Harness (Gao et al., 2024) or Qwen-Math Verifier (Huang et al., 2025).

Inference setting. To monitor the steps and intervene in the generation process, we assume that each model generates one token at a time, and we split the steps dynamically. However, for the purposes of our experiments instead of re-designing the generation process, we performed standard inference and applied our *Step-Tagging* and *Early-Stopping* algorithms *offline*.

Prompt-guided budget compression. Figure 12 presents the Prompt-guided baselines that we used to evaluate our ST-ES framework.

[1] https://github.com/huggingface/Math-Verify

Prompt Baselines $\mathcal{P}_{\text{guided}}$

User Prompt - $\mathcal{P}_{\text{user}}^{(0)}$

User Prompt: Please do not reason extensively, be succinct, and put your final answer within boxed{}. {question}

System Prompt $\mathcal{P}_{\text{system}}^{(0)}$

System Prompt: Respond concisely and confidently. Skip validations and over-verification steps.
User Prompt: {question}

System Prompt - $\mathcal{P}_{\text{system}}^{(1)}$

System Prompt: Respond concisely and confidently. Skip validations and over-verification steps. Here is an examples: Example 1: {FS_1}
User Prompt: {question}

System Prompt - $\mathcal{P}_{\text{system}}^{(3)}$

System Prompt: Respond concisely and confidently. Skip validations and over-verification steps. Here are some examples: Example 1: {FS_1} Example 2: {FS_2} Example 3: {FS_3}
User Prompt: {question}

Example 1 - Verification step

Wait, let me double-check. If I plug in $x = -3$ into the denominator, $(-3)^2 + (-3) - 6 = 9 - 3 - 6 = 0$. Yep, that works. For $x = 2 : 2^2 + 2 - 6 = 4 + 2 - 6 = 0$. Correct. So both roots are valid.

Example 2 - Verification step

Therefore, the graph of $y = 2(x^2 + x - 6)$ has vertical asymptotes at $x = -3$ and $x = 2$, so that's two vertical asymptotes. I don't think there's any chance that I made a mistake here, but maybe I should check by graphing the function or plugging in values close to -3 and 2 to see if the function does go to infinity.

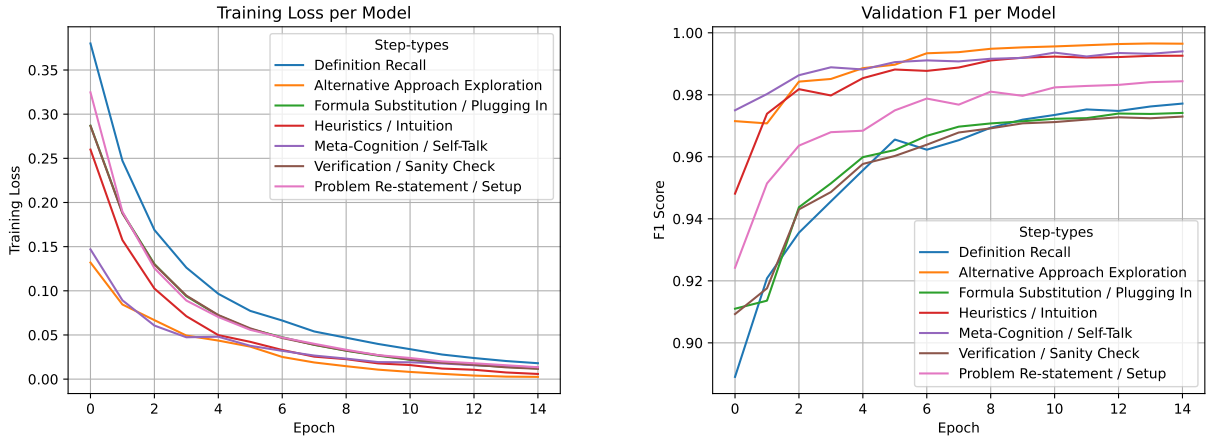
Example 3 - Verification step

Another test with $n = 3$. Let's compute manually. All non-empty subsets: Single elements: $\{1\}$, $\{2\}$, $\{3\}$ with sums 1,2,3. Pairs: $\{1,2\} \to 2 - 1 = 1$ $\{1,3\} \to 3 - 1 = 2$ $\{2,3\} \to 3 - 2 = 1$. Triple: $\{1,2,3\} \to 3 - 2 + 1 = 2$. Total sum: $1 + 2 + 3 + 1 + 2 + 1 + 2 = 12$. Using the formula: contributions from each k : $k = 3$: $3 * 2^2 * 1 = 3 * 4 = 12$. $k = 1$ and $k = 2$ contribute 0. So total sum 12, which matches.

Figure 12. Prompt baselines

E. Training of step-taggers

Figure 13 details the training metrics of our BERT step-taggers. Each step-tagger is trained to identify a specific step-type. We trained and validated our BERT models on reasoning traces obtained using DS-Qwen14B inferred on MATH500 and GPQA train datasets. We used a *balanced cross-entropy* to enhance the performance of the models on low-represented classes. We trained classifiers for 15 epochs, with a batch size of 16 and we used an AdamW optimizer with a learning rate of 2.10^{-5}.



(a) Training loss *(b)* Validation Micro-F1

Figure 13. Training metrics of the BERT Step-Taggers

F. Training-Inference cost trade-off of ST-ES

This section presents an analysis of the cost of ST-ES. In this paper, we trained a seven BERT models (on seven different step-types). However, the data generation and labeling process is a one-time exercise, using the DS-Qwen14B on MATH500 and GPQA train split. Then, each step-tagger is used for all models and datasets configurations.

Figure 14 presents the training-inference cost trade-off of ST-ES on the MATH500 test dataset, over the 3 selected models, across the five seeds for each models (to observe how ST-ES scales). We compare the estimated saved runtime of ST-ES against the training runtime.

ST-ES testing inference time includes three components: (1) the runtime to generate the early-stopped trace (assuming linear relation between the token-count and the runtime (Oh et al., 2022)), (2) the runtime to infer the BERT Step-Tagging module, and (3) the runtime to forcing the module to answer.

The training runtime also includes three components: (1) the inference of DS-Qwen14B on training samples (MATH500 and GPQA train), (2) the annotation of reasoning steps obtained from the training inference, and (3) the training runtime of the BERT Step-Taggers (seven in total, one for each selected step-type).

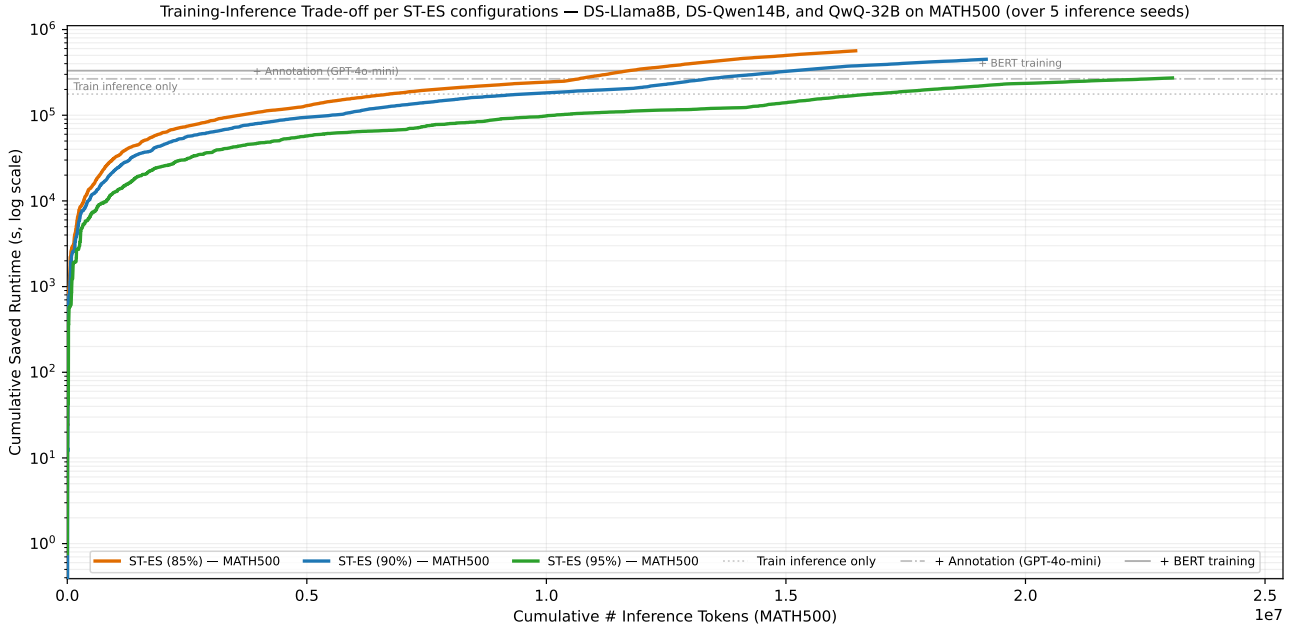


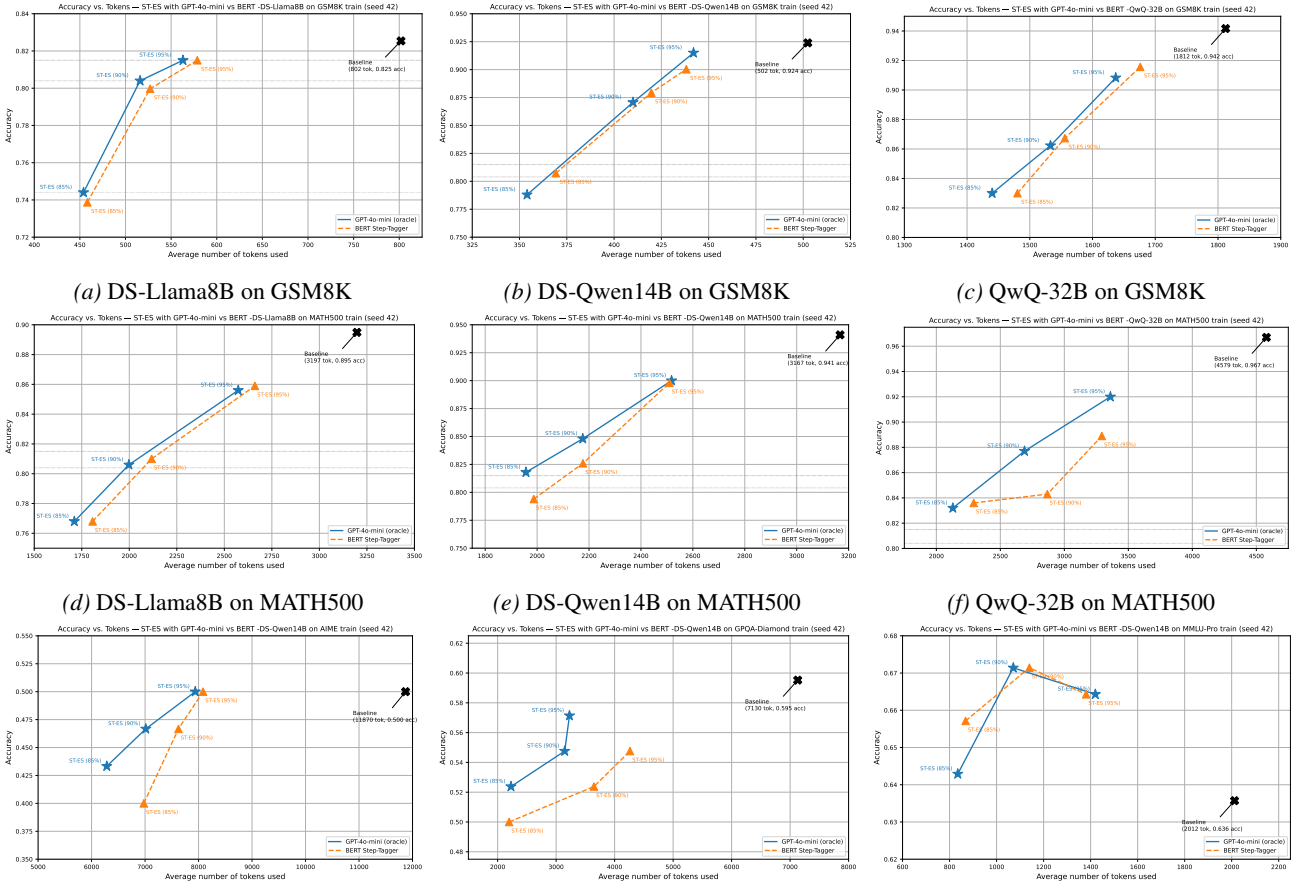
Figure 14. Cumulative saved runtime of ST-ES vs. Cumulative number of tokens generated on MATH500 test dataset for DS-Llama8B, DS-Qwen14B, and QwQ-32B. The gray lines represents the training costs of the seven BERT Step-tagging modules. We observe that configurations ST-ES 85% and 90% fully recovers the training costs of the BERT Step-Tagging modules.

We observe that 85% and 90% fully recovers the training costs solely with their inferences on the MATH500 dataset, and 95% almost recovers all training costs. Furthermore, 85% saves almost two time the training runtime, while needing lower amount of tokens. But this is at the cost of the accuracy.

We note that the gains expected by ST-ES would be even larger when including saved runtime on other datasets used in this paper (namely GSM8K, MMLU, and AIME). Importantly, the ST-ES early-stopping criteria is designed to be used at scale, and only requires a one-time training exercice.

G. Impact of BERT Step-Tagging performance on ST-ES

Figure 15 compares ST-ES criteria applied using the `GPT-4o-mini` (in blue) and BERT Step-Tagging (in orange) tags. We observe that the approximation achieved best results on GSM8K and MATH500 datasets. On AIME, GPQA, and MMLU, the difference in performance is higher, but still acceptable (less than 2% accuracy drop, and 5 to 10% higher token-count). We suspect that this is due to two reasons. First, the higher value of δ on harder tasks means that if the BERT taggers are making an error, this error is likely to be propagated across longer reasoning chains. Second, the datasets contains much less samples, future work should look at expanding the datasets to observe how the results scales.



(a) DS-Llama8B on GSM8K *(b)* DS-Qwen14B on GSM8K *(c)* QwQ-32B on GSM8K

(d) DS-Llama8B on MATH500 *(e)* DS-Qwen14B on MATH500 *(f)* QwQ-32B on MATH500

(g) DS-Qwen14B on AIME *(h)* DS-Qwen14B on GPQA *(i)* DS-Qwen14B on MMLU-Pro

Figure 15. Number of Tokens vs. Pass@1 - ST-ES with `GPT-4o-mini` vs. BERT Step-tagging on train datasets - seed 42

H. Evaluation of Step-Tagging Early-Stopping

Table 6 reports all the token-usage, the proportion of saved number of tokens, the Avg@5, the Pass@5 and the Cons@5 for all configurations. Results are averaged over the 5 seeds we used.

Model	Config.	MATH500					GSM8K				
		# Tokens	Saved (%)	Avg@5	Pass@5	Cons@5	# Tokens	Saved (%)	Avg@5	Pass@5	Cons@5
DS-8B	Standard	3655.0	–	0.878	0.970	0.726	958.3	–	0.829	0.943	0.651
	Basel. $\mathcal{P}_{\text{user}}^{(0)}$	2989.6	18.21	0.866	0.952	0.722	525.8	45.13	0.771	0.917	0.579
	Basel. $\mathcal{P}_{\text{system}}^{(0)}$	2634.4	27.92	0.817	0.960	0.592	456.9	52.32	0.763	0.895	0.574
	Basel. $\mathcal{P}_{\text{system}}^{(1)}$	2139.5	41.46	0.782	0.942	0.526	560.8	41.48	0.754	0.914	0.537
	Basel. $\mathcal{P}_{\text{system}}^{(3)}$	2565.3	29.81	0.789	0.952	0.540	830.5	13.34	0.748	0.904	0.541
	ST-ES (85%)	1801.4	50.71	0.761	0.908	0.568	521.5	45.58	0.708	0.842	0.575
	ST-ES (90%)	2120.3	41.99	0.793	0.920	0.616	618.8	35.43	0.788	0.884	0.692
	ST-ES (95%)	2614.5	28.47	0.845	0.936	0.704	696.7	27.29	0.809	0.890	0.727
DS-14B	Standard	3388.8	–	0.923	0.980	0.836	662.9	–	0.910	0.952	0.843
	Basel. $\mathcal{P}_{\text{user}}^{(0)}$	2691.5	20.58	0.933	0.982	0.834	505.1	23.80	0.856	0.956	0.662
	Basel. $\mathcal{P}_{\text{system}}^{(0)}$	2346.2	30.77	0.886	0.966	0.754	470.9	28.96	0.873	0.949	0.710
	Basel. $\mathcal{P}_{\text{system}}^{(1)}$	2211.4	34.74	0.873	0.974	0.708	566.5	14.54	0.838	0.952	0.629
	Basel. $\mathcal{P}_{\text{system}}^{(3)}$	2535.0	25.19	0.879	0.968	0.748	839.6	-26.65	0.841	0.952	0.631
	ST-ES (85%)	2158.4	36.30	0.827	0.950	0.646	377.2	43.09	0.527	0.802	0.262
	ST-ES (90%)	2393.6	29.36	0.853	0.960	0.688	415.9	37.26	0.621	0.843	0.368
	ST-ES (95%)	2851.9	15.84	0.909	0.972	0.812	464.6	29.91	0.865	0.946	0.723
QwQ-32B	Standard	4475.3	–	0.954	0.984	0.898	2075.7	–	0.953	0.965	0.934
	Basel. $\mathcal{P}_{\text{user}}^{(0)}$	2908.8	35.00	0.955	0.986	0.916	988.0	52.40	0.952	0.968	0.937
	Basel. $\mathcal{P}_{\text{system}}^{(0)}$	3201.1	28.47	0.932	0.976	0.852	833.3	59.85	0.940	0.974	0.869
	Basel. $\mathcal{P}_{\text{system}}^{(1)}$	3182.4	28.89	0.925	0.974	0.856	871.2	58.02	0.943	0.975	0.876
	Basel. $\mathcal{P}_{\text{system}}^{(3)}$	3665.5	18.09	0.926	0.974	0.858	1387.3	33.16	0.935	0.974	0.855
	ST-ES (85%)	2415.1	46.03	0.839	0.950	0.666	1152.1	44.49	0.833	0.945	0.646
	ST-ES (90%)	2953.9	33.99	0.889	0.960	0.758	1338.1	35.53	0.899	0.958	0.794
	ST-ES (95%)	3559.5	20.46	0.922	0.966	0.846	1462.2	29.55	0.943	0.964	0.903

Table 6. Performance of Step-Tagging Early stopping - 5 seeds (40, 41, 42, 43, 44)

Table 7 shows the token-usage, the proportion of saved number of tokens, and the Pass@1 for DS-Qwen14B on AIME, GPQA, and MMLU test datasets. Results are obtained with a seed of 42.

Config.	AIME 23-24			GPQA-Diamond			MMLU-Pro		
	# Tokens	Saved (%)	Pass@1	# Tokens	Saved (%)	Pass@1	# Tokens	Saved (%)	Pass@1
Standard	13096.3	-	0.483	7024.7	-	0.603	2501.9	-	0.652
Basel. $\mathcal{P}_{\text{user}}^{(0)}$	12748.4	2.66	0.422	4801.0	31.66	0.577	2194.1	12.30	0.665
Basel. $\mathcal{P}_{\text{system}}^{(0)}$	11095.9	15.27	0.500	4768.3	32.12	0.571	2292.9	8.35	0.675
Basel. $\mathcal{P}_{\text{system}}^{(1)}$	10181.2	22.26	0.517	3736.2	46.81	0.539	1705.6	31.83	0.637
Basel. $\mathcal{P}_{\text{system}}^{(3)}$	11635.7	11.15	0.467	4530.1	35.51	0.564	1935.2	22.65	0.613
ST-ES (85%)	6082.9	53.55	0.283	2383.2	66.07	0.487	927.6	62.92	0.606
ST-ES (90%)	8609.6	34.26	0.383	3326.6	52.64	0.557	1153.1	53.91	0.643
ST-ES (95%)	9299.8	28.99	0.417	4432.9	36.89	0.564	1469.6	41.26	0.670

Table 7. Performance of ST-ES - DS-Qwen14B - seed 42