Implementing STaR on GSM8K with Llama-3.2-3B-Instruct: Self-Taught Reasoner for Mathematical Reasoning

NLP Assignment 1

Adarsh Mohan-1233380241

October 21, 2025

Abstract

We implement the STaR (Self-Taught Reasoner) framework to bootstrap chain-of-thought (CoT) reasoning on GSM8K using `meta-llama/Llama-3.2-3B-Instruct`. Following the assignment constraints, we compare three methods: (i) Zero-Shot CoT prompting, (ii) Vanilla SFT (fine-tuning on GSM8K rationales from the train split only), and (iii) STaR (iterative rationale generation with rationalization hints and SFT). Using consistent decoding with vLLM, we obtain exact-match (EM) accuracies on GSM8K test of **46.78%** (Zero-Shot CoT), **65.81%** (Vanilla SFT), and **78.17%** (STaR). We include full prompts, workflow, commands, results, and analysis.

1 Introduction

Generating explicit rationales improves LLM reasoning. STaR iteratively augments a dataset with model-generated rationales: solve with CoT; when the answer is wrong, *rationalize* by conditioning on the gold answer; fine-tune on rationales that yield the correct final answer; repeat. We implement this loop with Llama-3.2-3B-Instruct on GSM8K.

2 Setup

Model. `meta-llama/Llama-3.2-3B-Instruct`.
Dataset. GSM8K (`train` for SFT/STaR; `test` for evaluation).
Decoding. vLLM for fast deterministic batched generation.
Metric. Exact Match (EM) on the numeric final line (‘‘`#### <number>`’’).
Precision. bfloat16 (GPU) or float32 (CPU).

3 Prompts Used

Zero-Shot CoT (no hint)

Listing 1: Prompt used in all runs without hints.

```
"You are a careful math tutor. Solve step by step. End with: #### <number>"
<s>[INST] <<SYS>>
You are a careful math tutor. Solve step by step. End with: #### <number>
<</SYS>>
Q: {question}
A: [/INST]
```

Rationalization (with hint)

Listing 2: Prompt used for rationalization with the correct final answer as a hidden target.

```
"You already know the correct final answer, but DO NOT mention that fact.
 Produce a clear step-by-step rationale that independently leads to it.
 End exactly with: #### {gold}."
<s>[INST] <<SYS>>
You already know the correct final answer, but DO NOT mention that fact.
Produce a clear step-by-step rationale that independently leads to it.
End exactly with: #### {gold}.
<</SYS>>
Q: {question}
A: [/INST]
```

4 Workflow and How to Run

Code Workflow

1. **Zero-Shot CoT:** Evaluate the base model on GSM8K test using reasoning prompts.
2. **Vanilla SFT:** Fine-tune on GSM8K train set with ground-truth rationales.
3. **STaR:** Iteratively bootstrap rationales using the model itself:
   - Generate initial rationales; accept those producing correct final answers.
   - Retry incorrect ones with the correct final answer as hidden hint.
   - Fine-tune on all accepted rationales.
   - Repeat for multiple rounds, each time improving the model.
4. **Evaluation:** All models are evaluated with `evaluate_model.py` for EM accuracy.

5 Reproducibility Commands

All experiments can be reproduced exactly with the following shell commands.

Create Environment (run once)

```
python3.11 -m venv ~/envs/star-sol
source ~/envs/star-sol/bin/activate
python -m pip install --upgrade pip setuptools wheel
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121
pip install "transformers>=4.44.0" "datasets>=2.19.0" accelerate peft trl bitsandbytes \
            evaluate sentencepiece tiktoken tqdm safetensors huggingface_hub vllm
```

Full Pipeline to Reproduce Results

```
source ~/envs/star-sol/bin/activate

# 1) Zero-Shot CoT on GSM8K test
python Scripts/run_zero_shot.py \
  --engine vllm \
  --batch_size 16 \
  --max_new 256 \
```

```
  --save_dir Data/eval_zeroshot

# 2) Build Vanilla SFT training JSONL
python Scripts/build_vanilla_sft.py

# 3) Train Vanilla SFT
python Scripts/train_sft_vanilla.py \
  --train_jsonl Data/vanilla_train.jsonl \
  --out_dir Models/vanilla_sft

# 4) Evaluate Vanilla SFT
python Scripts/evaluate_model.py \
  --model_dir Models/vanilla_sft \
  --save_dir Data/eval_vanilla \
  --engine vllm \
  --batch 8 \
  --max_new 640

# 5) Run STaR (iterative bootstrapping)
python Scripts/star_bootstrap_iteratively.py


# 6) Train base-model from produced STaR JSONL (SFT)
python Scripts/train_sft_star.py \
  --train_jsonl Data/star_train_from_iterative.jsonl \
  --out_dir Models/star_sft \
  --per_device_batch 1 \
  --grad_accum 48 \
  --epochs 1 \
  --max_len 1536 \
  --lr 2e-5 \
  --warmup_ratio 0.05 \
  --weight_decay 0.05

# 7) Evaluate STaR SFT
python Scripts/evaluate_model.py \
  --model_dir Models/star_sft \
  --save_dir Data/eval_star \
  --engine vllm \
  --batch 8 \
  --max_new 640
```

6 Results

Exact-Match (GSM8K Test, n=1319)

Method	Correct	Accuracy (EM)	Notes
Zero-Shot CoT (base)	617	0.4678	vLLM, max_new=256
Vanilla SFT	868	0.6581	vLLM, SFT on human rationales
STaR (iterative)	1031	**0.7817**	vLLM, bootstrapped + hinted

Table 1: Exact-match accuracy across methods on GSM8K test.

Raw Metrics.


```
{"method":"zero_shot_cot","n":1319,"correct":617,"accuracy_exact_match":0.4678}
{"method":"eval_vanilla","n":1319,"correct":868,"accuracy_exact_match":0.6581}
{"method":"eval_star","n":1319,"correct":1031,"accuracy_exact_match":0.7817}
```


7 Analysis: Why STaR Helps

1. Error-Directed Data Growth. STaR expands the training set with *only* rationales that lead to the correct answer. This selectively reinforces effective reasoning chains.
2. Rationalization as Guided Search. Conditioning on the correct final answer yields more faithful rationales, improving alignment between reasoning and solution.
3. Iterative Curriculum. Each round builds a more capable model, able to solve increasingly difficult problems, forming a self-improving loop.
4. Quantitative Gains. STaR improves EM by +31.4 points over Zero-Shot CoT and +18.4 points over Vanilla SFT.

8 Screenshots (Evidence)

Figures below correspond to VS Code terminals and prompts from your runs.

9 Limitations

STaR assumes a reasonably competent base model to begin bootstrapping. Formatting errors in numeric answers may slightly reduce EM, even when reasoning is correct.

10 Conclusion

With identical architecture and data, STaR outperforms both Zero-Shot CoT and Vanilla SFT, demonstrating the benefits of rationalization and self-improvement through reasoning-based fine-tuning.

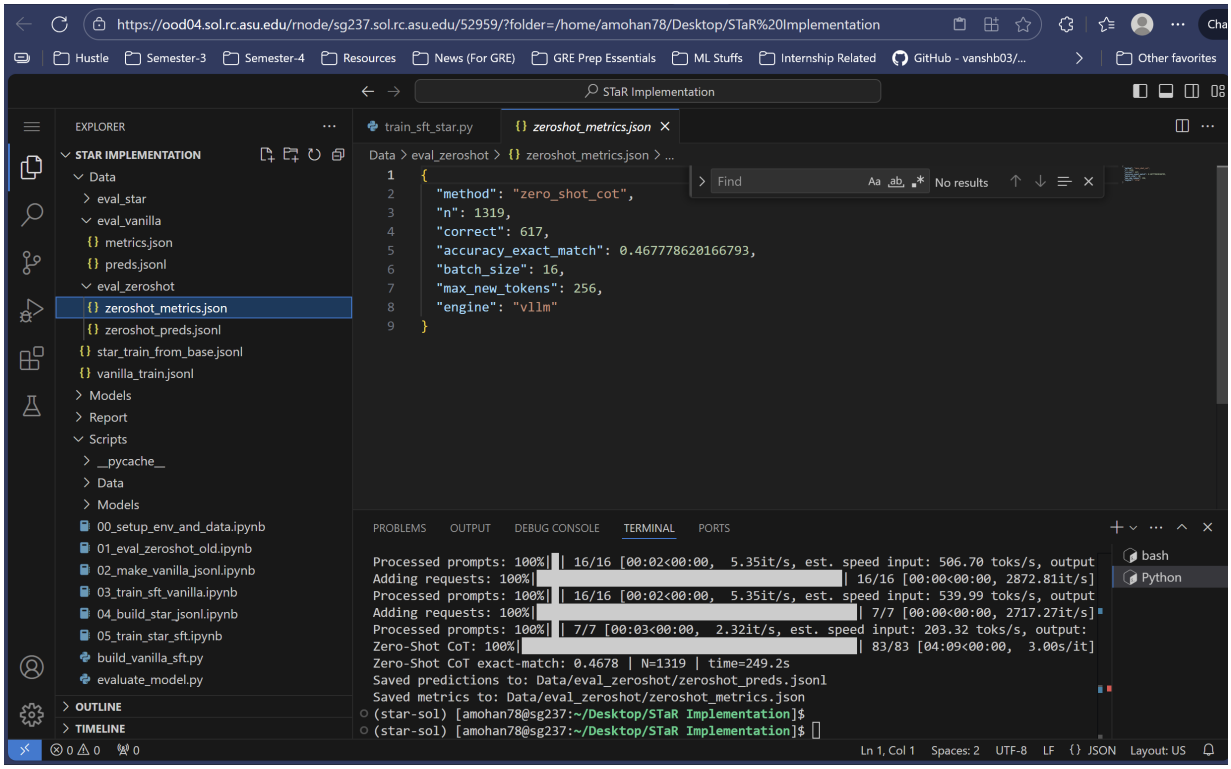
References

References

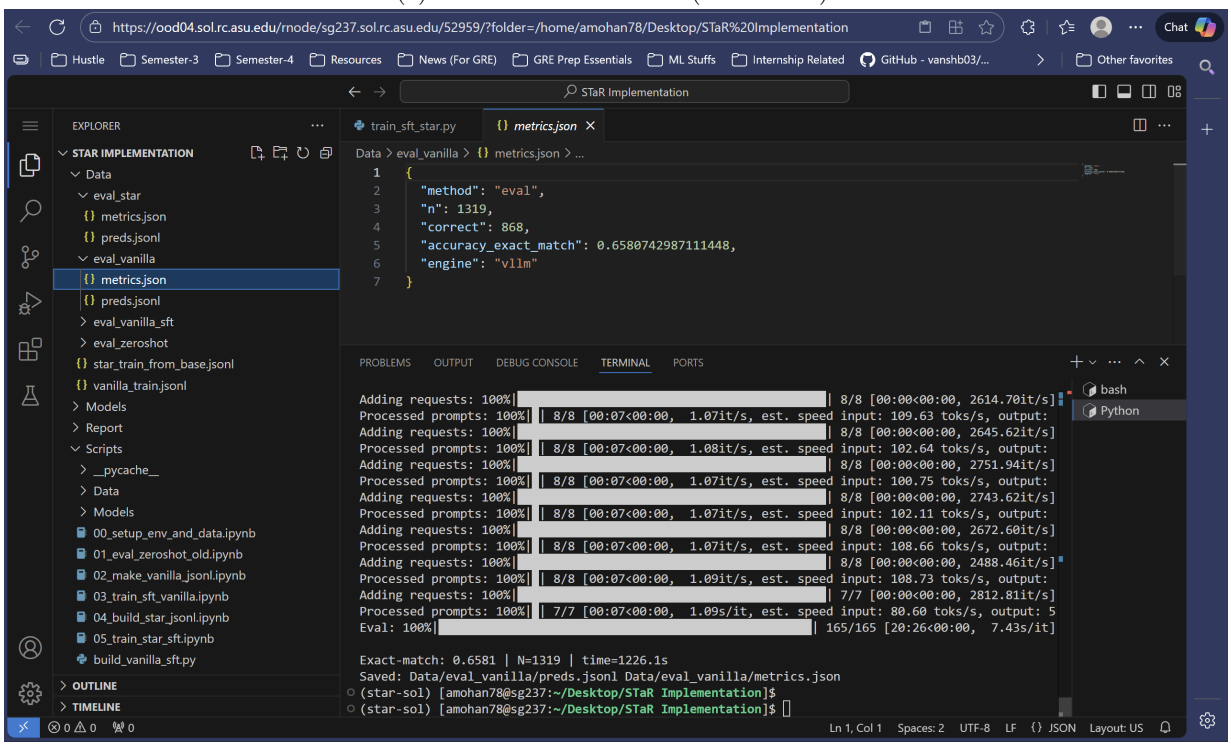
[1] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Self-Taught Reasoner—Bootstrapping Reasoning With Reasoning. *arXiv:2203.14465*, 2022.

Appendix A: File Inventory

- `shared.py` (prompts, model/vLLM helpers)
- `run_zero_shot.py` (Zero-Shot CoT evaluation)
- `build_vanilla_sft.py` (creates SFT dataset)
- `train_sft_vanilla.py` (fine-tuning baseline)
- `star_bootstrap_iteratively.py` (full STaR loop)
- `train_sft_star.py` (SFT from STaR data)
- `evaluate_model.py` (evaluation harness)

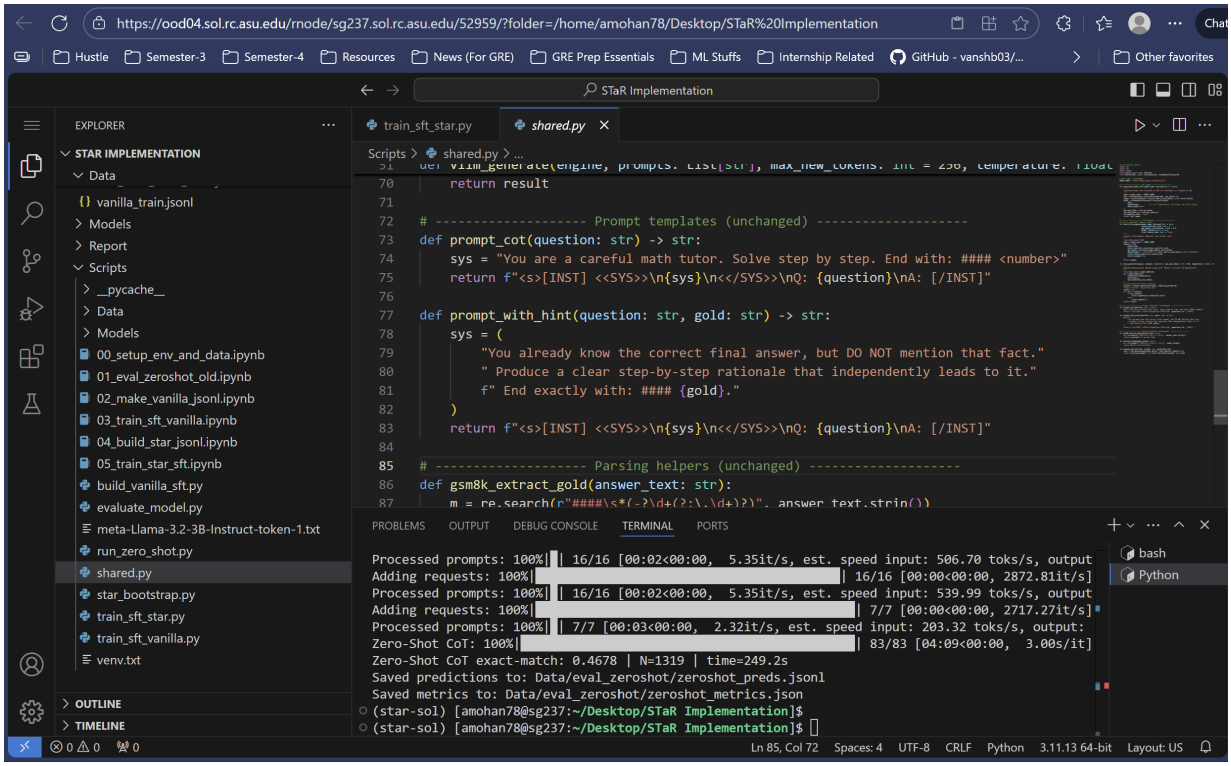


(a) Zero-Shot CoT metrics (EM ≈ 0.47).

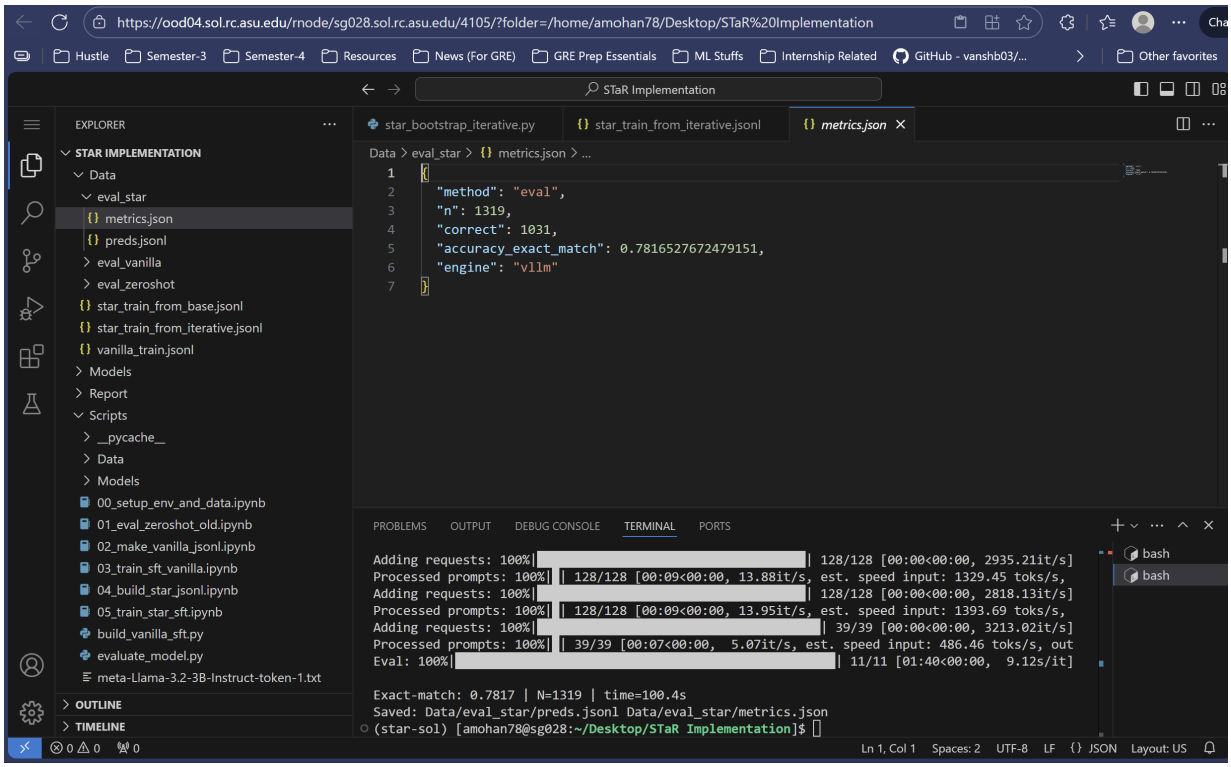


(b) Vanilla SFT metrics (EM ≈ 0.66).

Figure 1: Baseline runs (Zero-Shot and Vanilla SFT).



(a) Prompt templates in `shared.py`.



(b) STaR eval metrics (EM ≈ 0.78).

Figure 2: Prompts and STaR evaluation evidence.