

Supplementary Material

Overview and Directory Structure

We release the core dataset and source code for CoTP in this supplementary material.

```
core_code_and_dataset/  
├── core_dataset/  
│   ├── core_set_example.jsonl  
│   ├── mid-train_data_example.jsonl  
│   ├── sft_data_example.jsonl  
│   └── rl_data_example.jsonl  
└── source_code/  
    ├── annotation/  
    │   ├── annotate_cot_patterns.py  
    │   └── compute_cot_entropy.py  
    ├── data_selection/  
    │   ├── compute_pattern_tfidf_importance.py  
    │   └── data_selection.py  
    ├── pretrain/  
    │   ├── pretrain_gpt.py  
    │   └── pretrain_gpt.sh  
    ├── rl/  
    │   ├── general_runner.py  
    │   ├── grpo_trainer.py  
    │   ├── gspo_config.py  
    │   ├── megatron_gspo_loss.py  
    │   └── run_gspo.sh  
    └── sft/  
        └── settings.json
```

Dataset Description

core_dataset/

We release part of the core dataset, mid-train data, SFT training data, and RL training data of CoTP, with the review and curation of the full dataset currently in progress.

Core Set Example ([core_set_example.jsonl](#)): 100 mathematical reasoning examples, each with 2 related CoT responses

Mid-Train Data Example ([mid-train_data_example.jsonl](#)): 500 mathematical reasoning examples

Data Format: JSONL with fields:

- **identity**: Unique identifier
- **question**: Mathematical problem statement
- **cot_answer**: Chain-of-thought reasoning in `<think>` tags followed by the answer
- **pattern**: Detailed pattern analysis (in Chinese)

- **problem_type**: Problem category
- **output_token_entropies**: Token-level entropy data

SFT Data Example (**sft_data_example.jsonl**): 100 examples from SFT training data with format containing **input**, **target**, and conversation messages for supervised fine-tuning

RL Data Example (**rl_data_example.jsonl**): 100 examples from RL training data with format containing **prompt**, **reference**, and **data_type** for reinforcement learning training

Source Code Modules

annotation/

annotate_cot_patterns.py

- Analyzes Chain-of-Thought reasoning processes and extracts structured reasoning patterns using vLLM inference engine
- Input: JSONL with **cot** field; Output: Adds **pattern** field with structured analysis

compute_cot_entropy.py

- Computes token-level entropy values for Chain-of-Thought outputs using Hugging Face causal language models
- Input: JSONL with **question** and **cot** fields; Output: Adds **output_token_entropies** field

data_selection/

compute_pattern_tfidf_importance.py

- Computes TF-IDF importance scores for reasoning patterns across multiple CoT responses
- Clusters similar patterns using character n-gram cosine similarity and calculates importance weights per question

data_selection.py

- Performs Hungarian algorithm-based optimal matching between Core Set and Source Data
- Uses weighted DTW distances combining pattern chain similarity and entropy sequence similarity
- Features flexible lambda weighting to balance pattern vs entropy importance in matching decisions
- Outputs selected source records ordered according to core set with 1-to-1 matching guarantee

pretrain/

pretrain_gpt.py

Main GPT model pretraining implementation using Megatron framework.

pretrain_gpt.sh

Training launch script that configures environment variables and distributed training setup.

rl/

general_runner.py

Main entry point for distributed reinforcement learning training with role management and multiple trainer

support.

grpo_trainer.py

GRPO trainer implementation with online sampling, experience generation, and policy updates.

gsपो_config.py

Configuration file for GSPO training setup including model architecture and training parameters.

megatron_gspo_loss.py

Loss function implementations for PPO and GSPO training with clipping and regularization.

run_gspo.sh

GSPO training execution script with environment setup and hardware configuration.

sft/

settings.json

JSON configuration file for supervised fine-tuning using Megatron framework with parallelism and optimization settings.