Team025 チーム・ビクトリー

本郷 颯人[1]　加地 翔太[2]　小谷 真士[3]　嶋中 雄大[4]
宮川 裕貴[5]　中島 悠樹[1]　安孫子リク[1]　松田 公慶[6]
[1]東京大学　[2]独立研究者　[3]大阪公立大学
[4]芝浦工業大学　[5]中京大学　[6]筑波大学

1. パイプライン

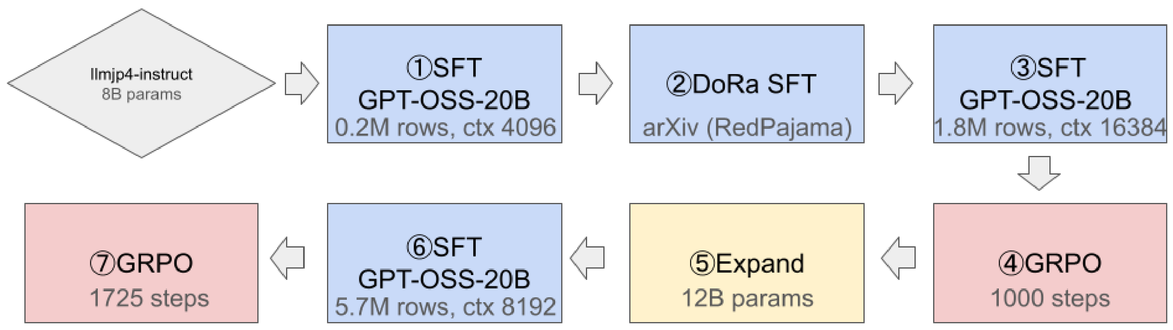

図1. モデルのトレーニングのパイプライン

2. データ

- 本コンペにおける制約事項
  - 近年、強力なプロプライエタリLLMによるデータ生成（蒸留）が主流だが、OpenAI等の利用規約では「競合モデルの開発への出力の利用」が明示的に禁止されている。[1]
  - 本コンペでは著作権的にクリーンなデータを使用することが求められた。
  上記の2つの制約を満たすデータは少なく、1M以上を超えるサイズのものは探した限り見つからなかった。
- 我々のチームが取った解決策
  - GPT-OSS-20B[2]を用いて、問題、解答の両方のデータを生成した。
  - GPT-OSS-20BがApache2.0ライセンスのためコンペ提出用モデルへの蒸留が可能である。
  - 外部データを一切用いることなく、モデルだけから取得するため、著作権的にクリーンである。
- 方法
  - データ生成条件の設定: 数学の分野、難易度（1～10）、ツールの使用有無を指定した。
  - 日本語プロンプトによる生成: ローカル環境の GPT-OSS-20B を使用し、問題・解答・思考過程を出力した。
  - CoT形式では、合計で**約8M**のフィルタ済みのインストラクションデータを生成した。
  - フィルタリングの詳細::
    - ハルシネーション対策として生成トークン長を制限した。
    - 問題と解答の適合性をGPT-OSS-20B 自身に再評価させ、フィルタリングした。（LLM-as-a-Judge [3]）

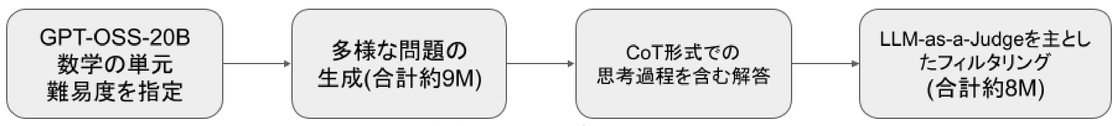

図2. データ生成のパイプライン

3. 事後学習（SFT+モデルサイズ拡張）

継続事前学習

- Arxivデータセットを用いる
- コンテキスト長を**16,384**まで拡張
- **DoRa[4]** を用いることで、壊滅的忘却を防ぐ

モデルサイズの拡張

- **OpT-DeUS[5]** を用いて層数を32層から48層にする
- パラメータ数が1.5倍に上昇

SFT

- **Open-R1[6]**をベースとしたコードで、1ノードで学習する

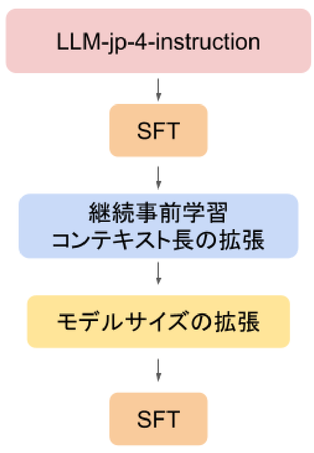

図3. コンテキスト長とモデルサイズの拡張を伴う学習パイプライン

4. 事後学習（open-instructによる非同期RL）

図4. 非同期RLにおける有効バッチ内平均生成長のグラフ。
縦軸がトークン長さ、横軸がエピソード数を示す。

最適化における帰納バイアス設計ポイント:
Token-level loss集約による長さ依存の正負学習信号強化
サンプル平均の総和ではなくトークン平均の総和を採用

RLトレースサンプル:1000 step v.s. 1725 step

- 1.32×長さ↑ 正答率 +12.5 ↑: P16 分数方程式
  verify, double-checkの増加
- 0.41×長さ↓ 正答率 +34.6 ↑:P5 指示追従問題
  maybe, alternatively減少
- 1.19×長さ↑ 正答率 -9 ↓:P14 極限
  検証追加かつ計算ミス増加
- 0.84×長さ↓ 正答率 -47.5 ↓:P9 1次関数交点
  答えが解なしに対して数値を出力

補足資料QRコード:
RLサンプル詳細とEDA

トークン平均の損失集約の詳細:

コードベースは **Open-Instruct[7]** に基づく。これはかつて Olmo 3[8] に対する事後学習として使用された。目的関数における報酬信号の集約の仕方は、従来のGRPO[9]のサンプル単位の平均ではなく、バッチ内の総トークン数の平均をとるtoken-level loss集約を用いた。そのため、vLLMが長い系列長を生成すればするほど、その分、正解ならば正のadvantage（学習信号）による更新が相対的に大きくなる。なぜなら、アドバンテージはトークン毎の項に掛かる重みとして働く一方で、サンプル平均では、サンプル長による正規化を経るため、系列長の差が均され、長文サンプルの寄与が相対的に弱まるからである。同様に、長文かつ不正解ならば負のadvantage（学習信号）も相対的に強まる。

5. 推論システム（Self-Consistency）

Self-Consistency[10]の概要

- 1つの問題を複数回モデルに入力し、複数個の解答を得る
- 得られた解答の内、最も出現頻度の高い解答を最終的な解答とする

今回のコンペティションにおける実装

1. 1つの問題に対して複数回の出力を取得
2. 各出力の数式部分を抽出（math-verifyのparse関数を使用）
3. 抽出結果を比較し、最も出現頻度の高い数値・数式を最終解答とする

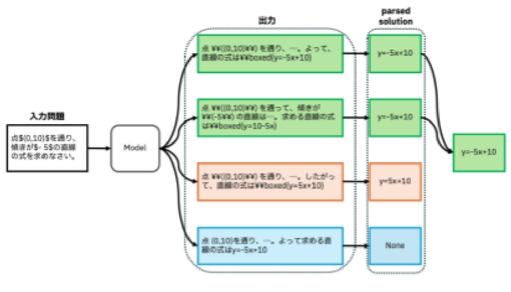

図5. 推論処理のイメージ図（[10] Fig 1 を元に作成）

6. 評価結果

	cons@1	cons@40	cons@160
LIm-jp-4-instruct (baseline)	0.354	0.552	0.594
GRPO後(図1⑦)	0.818	0.878	**0.880**

表1. コンペ運営から提供された評価用データ500問に対する正解率. @kは推論回数を表す.

- 推論回数を増やすことにより正解率が増加した
- 事後学習により同じ推論回数でも精度が大幅に増加しており，事後学習の重要性が示された
- 事後学習後のモデルで比較した場合cons@40とcons@160では大きな違いが見られなかった

7. 試行錯誤

①RSA[11]

手法
・多数決や集約を繰り返し行うことで、ハルシネーションを抑制
・反復を通じて正解率を段階的に向上させる
・学習データにない難問に対し、試行錯誤のプロセスを 擬似的に実行させる

採用しなかった理由
・精度が向上しなかった。
・LLM自身による誤答のフィルタリングが不十分であり、質の低い回答を排除しきれなかった。
・膨大なトークン数を必要とすることもあった

② s1[12]

手法
・出力に"wait"と挿入したものを新しい入力として再度モデルに入力し、出力を獲得

不採用の理由
・Self-Consistencyのみと、同じ回数のSelf-Consistencyとs1の両方を比較した際に、後者の精度の向上が見られなかったため

③コードSFT

目的
・LLMにPythonコードを生成・実行させる推論
・複雑な計算ミスを削減し, 数学タスクの正答率の向上

手法
GPT-OSS-20Bから蒸留した。手法はCoT形式と同様。

不採用の理由
教師モデルであるGPT-OSS-20Bについて評価を行ったところ、
CoT形式の方がコード実行形式よりも有意に精度が高かったため。

④RAG

目的
ベクトルベースで類似問題を検索し、Few-shotの例題として入力プロンプトに含める

手法
multilingual-e5-base[13]を用いてembeddingし、FAISSを用いてデータセットの類似問題を検索する。top5として出力した問題と解答のセットを参考問題として追記する。

不採用の理由
精度が向上しなかった。
top5の簡易な実装で検証したため高品質な参照問題を用意できなかった

8. 参考文献

[1] Wang, X., et al. (2024). *A survey on knowledge distillation of large language models*. arXiv. https://arxiv.org/abs/2402.13116
[2] OpenAI. (2025). *Introducing gpt-oss*. OpenAI. https://openai.com/index/introducing-gpt-oss/
[3] Zheng, L., et al. (2023). *Judging LLM-as-a-judge with MT-bench and Chatbot Arena*. Advances in Neural Information Processing Systems, 36, 46595–46623.
[4] Liu, S. Y., et al. (2024). *DoRA: Weight-decomposed low-rank adaptation*. In Proceedings of the 41st International Conference on Machine Learning (ICML).
[5] Cao, M., et al. (2025). *Progressive depth up-scaling via optimal transport*. arXiv. https://arxiv.org/abs/2508.08011
[6] Hugging Face. (2025). *Open-R1: Fully open reproduction of DeepSeek-R1*. GitHub. https://github.com/huggingface/open-r1
[7] Allen Institute for AI. (2023). *Open-Instruct: Training open instruction-following language models*. GitHub. https://github.com/allenai/open-instruct
[8] Team OLMo. (2025). *OLMo 3*. arXiv. https://arxiv.org/abs/2512.13961
[9] DeepSeek-AI. (2025). *DeepSeek-R1: Incentivizing reasoning capability in large language models via reinforcement learning*. arXiv. https://arxiv.org/abs/2501.12948
[10] Wang, X., et al. (2022). *Self-consistency improves chain-of-thought reasoning in language models*. arXiv. https://arxiv.org/abs/2203.11171 (arXiv)
[11] Venkatraman, S., et al. (2025). *Recursive self-aggregation unlocks deep thinking in large language models*. arXiv. https://doi.org/10.48550/arXiv.2509.26626
[12] Muennighoff, N., et al. (2025). *s1: Simple test-time scaling*. arXiv. https://doi.org/10.48550/arXiv.2501.19393
[13] Wang, L., et al. (2024). *Multilingual E5 text embeddings: A technical report*. arXiv. https://arxiv.org/abs/2402.05672