

# チーム dentaku FT-LLM 2026 (数学タスク) システム解説

工藤 慧音<sup>1,2</sup> 日高雅俊<sup>3</sup> 塩野 大輝<sup>1</sup> 鴨田 豪<sup>4,5</sup> 葉 夢宇<sup>1</sup> 高橋 良允<sup>1,2</sup>  
青木 洋一<sup>1,2</sup> 矢野 一樹<sup>1</sup> 谷口 雅弥<sup>2,1</sup> 伊藤 拓海<sup>1,3</sup> 坂口 慶祐<sup>1,2</sup> 鈴木 潤<sup>1,2,6</sup>  
<sup>1</sup> 東北大学 <sup>2</sup> 理化学研究所 <sup>3</sup> Machine Learning Solutions 株式会社  
<sup>4</sup> 総合研究大学院大学 <sup>5</sup> 国立国語研究所 <sup>6</sup> 国立情報学研究所 LLMC  
keito.kudo.q4@dc.tohoku.ac.jp

## 概要

本レポートではチーム dentaku の FT-LLM 2026 (数学タスク) [1] における提出システムについて解説する。本システムでは gpt-oss-120b [2] を教師モデルとし、思考過程も含めた知識蒸留を行った llm-jp-v4-8B を基盤としている。モデルは入力として数学問題を受け取り、Python のコードとしてその解法を生成する。異なる設定で学習を行った複数の llm-jp-v4-8B の出力の中から、尤もらしい出力をランキングにより選択することを繰り返し、最終的なシステムの出力とした。コンペティション終了後の事後評価の結果、学習した llm-jp-v4-8B は教師モデルである gpt-oss-120b と同等の性能を達成した。一方で、複数モデルを組み合わせた提出システムの性能は最高性能の単一のモデルに劣る結果となり、複数モデルの出力の組み合わせ方には改善の余地があることが明らかとなった。

## 1 はじめに

FT-LLM 2026 チューニングコンペティション 数学タスク (オープン枠) [1] におけるチーム dentaku の提出システムについて解説する。本タスクは、日本の中学校・高等学校で扱う数学問題に解答するための llm-jp-v4-8B<sup>1)</sup> を基盤とするシステムを構築し、その性能を競うものである。オープン枠では、オープンなリソースのみを用いてシステムを構築する。

我々の提出システムは複数の追加学習を行った llm-jp-v4-8B の出力の中から、尤もらしい出力をランキングにより選択するパイプラインとなっている。推論時は Nvidia H200 x 8 台 (1 ノード) を 500 分間利用可能という時間制限の中で、可能な限りこの

1) llm-jp-v4-8B は LLM-jp [3] により開発中の言語モデルである。本レポート執筆時点では未公開のモデルである。

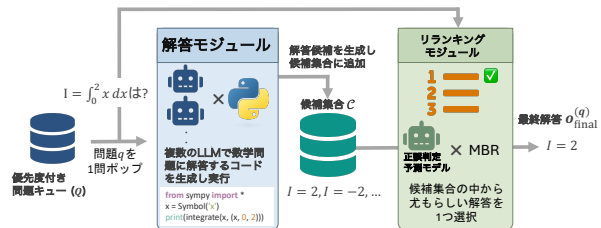


図1 システムの概要図。本システムは解答モジュールとランキングモジュールの2つから構成される。解答モジュールは、問題に対する解答となる Python コードを複数生成する。ランキングモジュールではその中から最終的な解答を選択する。

プロセスを繰り返し、正しい解答を生成することを目指した。我々は問題に対する解法を Python コードとして生成するように llm-jp-v4-8B の追加学習を行い、Python インタプリタの実行結果を各モデルの出力とした [4, 5]。追加学習は gpt-oss-120b [2] を教師モデルとした知識蒸留により、思考過程 (thinking) も含めた出力を再現することを目指した。本論文では、当該システムの詳細を提供する。また、各構成要素の有効性を検証する事後評価結果も報告する。

## 2 システム概要

図1 にシステムの概要を示す。本システムは、解答モジュールとランキングモジュールの2つから構成される。解答モジュールは問題文に対する解答候補を生成し、ランキングモジュールはそれらの解答候補群を再ランキングして最終出力を決定する。本システムでは、解答モジュールによる候補生成とランキングモジュールによる選択パイプラインを時間制限に到達するまで繰り返し、時間制限に到達した時点での解答をシステムの最終出力とする。アルゴリズム1 にパイプラインの疑似コードを

---

**Algorithm 1** 提出システムのパイプライン.

---

**Require:** 問題集合  $\mathbb{Q}$ , 時間制限  $T_{\text{limit}}$ 初期優先度  $P_{\text{init}}$ ,コード生成モデルの数  $K$ **Ensure:** 各問題に対する最終解答の集合  $\{o_{\text{final}}^{(q)}\}_{q \in \mathbb{Q}}$ 

```
1: 全ての問題  $q \in \mathbb{Q}$  を初期優先度  $P_{\text{init}}$  で優先度付  
   きキュー  $Q$  に追加する  
2: 各問題  $q \in \mathbb{Q}$  について, コードと解答のペア集  
   合  $\mathcal{C}_q \leftarrow \emptyset, o_{\text{final}}^{(q)} \leftarrow \text{null}$   
3:  
4: while 経過時間  $< T_{\text{limit}}$  do  
5:    $q \leftarrow Q.\text{Pop}()$   
6:   ▶ 解答モジュール (§ 3)  
7:    $\mathcal{C}_q \leftarrow \text{GENERATOR}(q, \mathcal{C}_q)$   
8:   ▶ リランキングモジュール (§ 4)  
9:    $o_{\text{final}}^{(q)}, p_{\text{score}} \leftarrow \text{RERANKER}(q, \mathcal{C}_q)$   
10:   $Q.\text{Push}(q, \text{priority} = p_{\text{score}})$   
11: end while  
12: return  $\{o_{\text{final}}^{(q)}\}_{q \in \mathbb{Q}}$ 
```

---

13: **function** GENERATOR( $q, \mathcal{C}_q$ )

```
14:   for  $i = 1, \dots, K$  do  
15:      $c \leftarrow \text{LLM}_{\text{code}}^{(i)}(q)$  ▶ 解法コードを生成 (§ 3)  
16:     ▶ コードを実行して標準出力を取得  
17:      $o_c \leftarrow \text{Python}(c)$   
18:     ▶ ルールベースの後処理 (§ 3.6)  
19:      $o'_c \leftarrow \text{PostProcess}(o_c)$   
20:      $\mathcal{C}_q \leftarrow \mathcal{C}_q \cup \{(c, o'_c)\}$   
21:   end for  
22:   return  $\mathcal{C}_q$   
23: end function
```

24: **function** RERANKER( $q, \mathcal{C}_q$ )

```
25:    $\mathcal{J} \leftarrow \emptyset$   
26:   for each  $(c, o'_c) \in \mathcal{C}_q$  do  
27:     ▶ 正誤判定を予測する二値分類 (§ 4.1)  
28:      $\mathcal{J}[c] \leftarrow \text{LLM}_{\text{judge}}(q, c)$   
29:   end for  
30:   ▶ MBR と  $\text{LLM}_{\text{judge}}$  のスコアを統合し最適解  
   と信頼度を算出 (§ 4.2)  
31:    $o_{\text{final}}^{(q)}, p_{\text{score}} \leftarrow \text{JUDGEENHANCEDMBR}(\mathcal{C}_q, \mathcal{J})$   
32:   return  $o_{\text{final}}^{(q)}, p_{\text{score}}$   
33: end function
```

---

示す.<sup>2)</sup>

解答モジュールは追加学習を実施した複数の  $\text{llm-jp-v4-8B}$  である  $\text{LLM}_{\text{code}}$  を基盤としたモジュールである. 問題文  $q$  を入力として受け取り, 解法を表現したコード  $c$  を出力する. その後,  $c$  を Python のインタプリタを用いて実行した標準出力  $o_c$  を得る.  $o_c$  にルールベースの後処理を適用して  $o'_c$  を候補解答とし, 解答候補集合  $\mathcal{C}$  に追加する.

リランキングモジュールは, 問題文  $q$  と生成されたコード  $c$  を受け取り,  $c$  が正しい解法であるか否かの二値分類を行うように追加学習を実施した  $\text{llm-jp-v4-8B}$  である  $\text{LLM}_{\text{judge}}$  と最小ベイズリスク復号 (MBR) [6] に着想を得たリランキング手法の組み合わせにより, 尤もらしい解答を解答候補集合  $\mathcal{C}$  の中から 1 つ選択する.

### 3 解答モジュール

最終的な提出システムに含めたモデル構築までの流れを図 2 に示す. コード生成モデル  $\text{LLM}_{\text{code}}$  の構築のため主に二段階の教師あり微調整 (以下 SFT) と GRPO [7] による強化学習を実施した. また, モデルマージやアンサンブルによるモデルのバリエーションの拡充も実施した. 以下では, 図 2 に示したモデル構築の流れと, モデルの出力に対するルールベースの後処理について説明する.

#### 3.1 データセット構築

データセット構築の流れを図 3 に示す. 異なるデータソースから第一段階 SFT に用いる比較的大規模なデータセットと, 第二段階 SFT に用いる小規模の高品質なデータセットを構築した.

**第一段階 SFT データセット** 比較的大規模なデータセットを用いて, 基盤モデルが問題に対する解法をコードで表現する能力を向上させることを目指した. 最終的に 6,094,013 件のデータセットとなった. 第一段階 SFT データセットは 3 つのデータソースから構築される.

- **グループ 1:** 英語で記述された数学問題を翻訳して利用
- **グループ 2:** 数学関連文書を日本語で書かれた数学問題に書き換えて利用
- **グループ 3:** 学習項目リストまたは, 既存のデー

---

2) 本システムのモデル, データセット, プロンプトは <https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts> にて公開予定である.

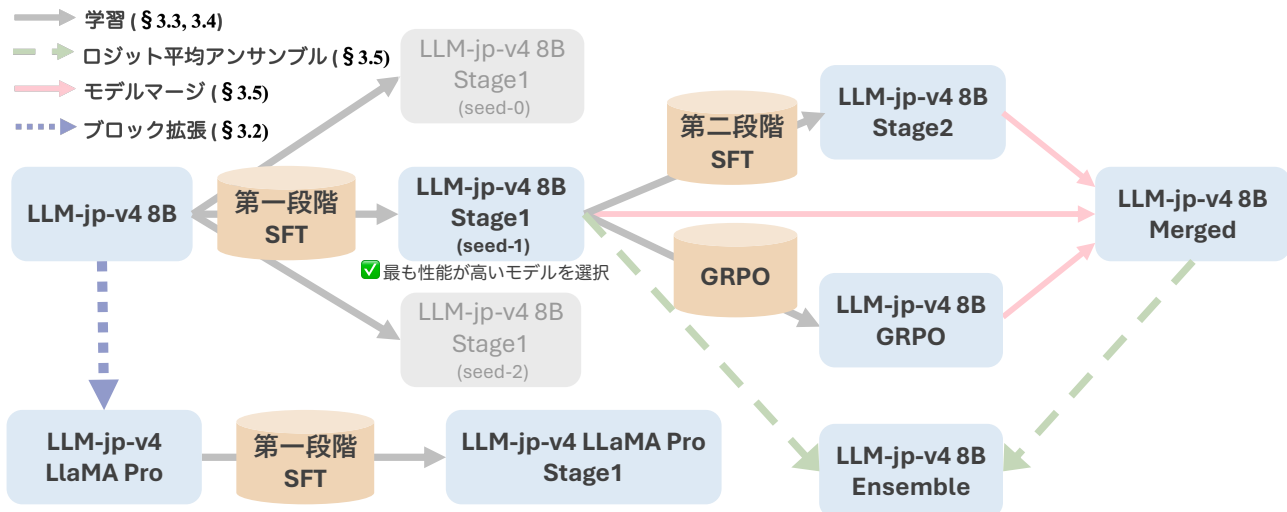


図2 モデル構築の流れ。二段階のSFT、GRPOによる学習と、LLaMA Pro (§ 3.2)、モデルマージ、ロジット平均アンサンブル (§ 3.5) を用いて複数のモデルを構築した。

タセットをシードとして、問題文を言語モデルにより生成した合成データセット

詳細は § A.1 を参照されたい。

**第二段階 SFT データセット** 小規模の高品質なデータソースから構築されたデータセットを用いて、モデルの出力の品質をさらに向上させることを目指した。データソースごとに独自のフィルタリング/クリーニング処理を適用し日本語の数学問題を作成した後に、gpt-oss-120bを用いてコード解法生成とフィルタリングを実施した。最終的に5,741件のデータセットとなった。詳細は § A.2 を参照されたい。

**コード解法生成** gpt-oss-120bを用いて、日本語の数学問題を入力としてそれに解答するためのPythonコードの生成を実施した。生成時に利用したプロンプトはGitHubを参照されたい<sup>3)</sup>。

**データフィルタリング** 表1にデータセットのフィルタリング/クリーニング処理の一覧を示す。これらの処理を適用することで、データセットの品質を向上させることを目指した。それぞれの処理の詳細については § B を参照されたい。

## 3.2 モデルアーキテクチャ

本システムでは、モデルアーキテクチャを拡張するために、LLaMA Pro [12] で提案されたブロック拡張手法を採用した。具体的には、llm-jp-v4-8B が持つ32層のTransformerブロックに対して、インターリーブ方式で8層の恒等ブロック (Identity Block) を追加し、合計40層のモデルを構築した。追加位置は各グループの末尾に1ブロックずつ挿入するインターリーブ配置を採用した。追加された各恒等ブロックは、LLaMA Pro の設計に従い、出力側の線形層の重み行列をゼロで初期化することで、拡張直後のモデル出力が元のモデルと同一となるように設計されている。これにより、拡張前後でモデルの出力を保持したまま学習を開始することができる。本研究では全ブロックを学習対象とし、SFT時に全パラメータを更新対象とする。

## 3.3 SFT

本研究では、SFTの初期モデルとしてllm-jp-v4-8Bを用いる。llm-jp-v4-8Bは事前学習済みモデルであり、指示学習は施されていない。そこで、llm-jp-v4-8Bに思考過程を含む構造化された出力を学習させるため、gpt-ossでも採用されているOpenAIのHarmony response format [13]を導入する。具体的には、llm-jp-v4-8Bのトークナイザに含まれる予約トークンを、Harmony response formatで用いる制御用の特殊トークンへ置換する。さらに、置換によ

3) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/code\\_generation\\_prompt\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/code_generation_prompt_system_prompt.md)  
 ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/code\\_generation\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/code_generation_prompt.md)

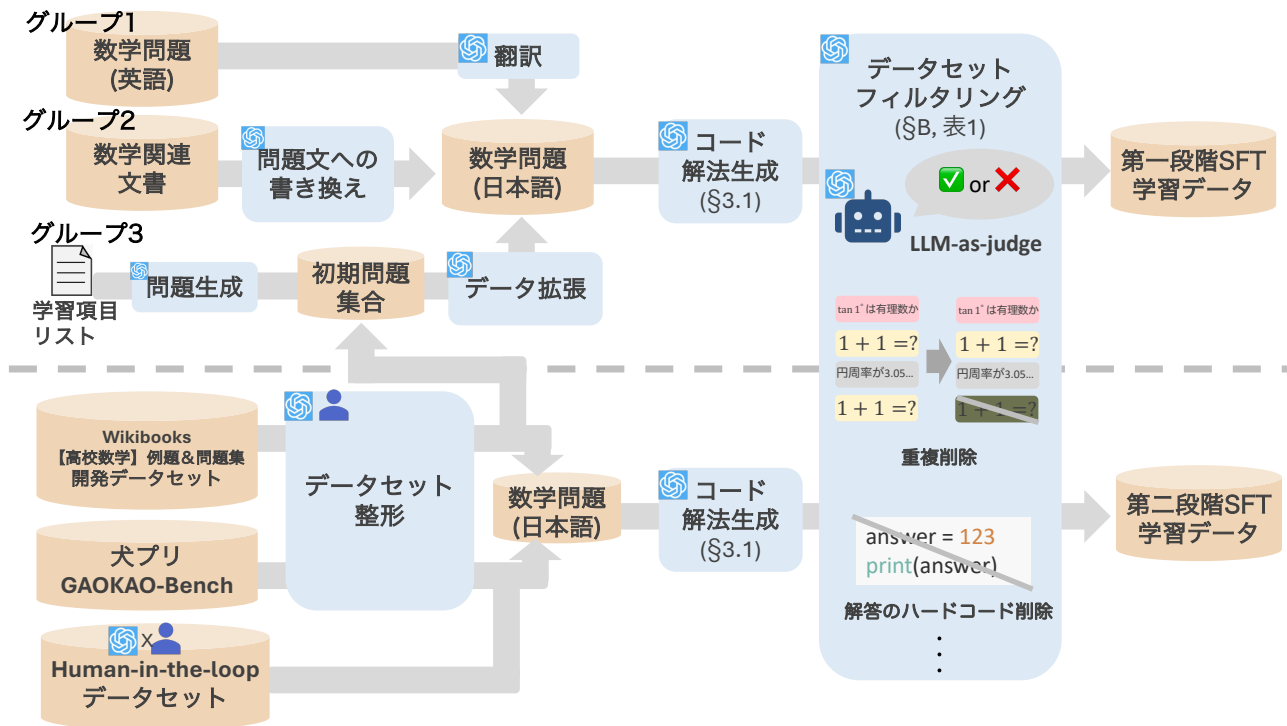


図3 データセット構築の流れ. OpenAI の Blossom ロゴは gpt-oss-120b または gpt-oss-20b の利用箇所を示す. 人の記号は人手でのデータ生成や修正を実施した箇所を表す.

て導入される特殊トークンの埋め込みベクトルは学習の安定化のために, llm-jp-v4-8B の既存の埋め込み行列から算出した平均ベクトルで初期化する. ただし平均ベクトルの算出時には, 予約トークンに対応する埋め込みは除外する. その上で, § 3.1 で構築した学習データセットを用いて llm-jp-v4-8B を SFT する. 学習時の損失は, *thinking* 部分および最終的な応答部分に含まれるトークンのみに対して計算し, それ以外のトークンは損失計算から除外する. 学習は § 3.1 で述べた第一段階/第二段階データセットを用いて二段階に分けて実施した. 第一段階では異なるシード値を指定して学習サンプルの順番をシャッフルし, 複数のモデルを学習した. その中から開発データセット上での性能が最も高いモデルを選択し, 第二段階 SFT 及び GRPO の学習に用いた. SFT における詳細な学習設定は § C に示す.

### 3.4 GRPO

本コンペティションの評価指標には math-eval [14] が用いられている. math-eval は数学タスクの評価ツールである Math-Verify [9] を基盤として本コンペティションに合わせて改良を加えたツールである. Math-Verify は, 正解とシス

テムの出力した解答が数学的に等価であるかをルールベースで判定することにより, ある程度柔軟な正誤判定を可能にしている. しかしながら, Math-Verify の判定能力には限界があり, 意味的には正解であっても, 不正解と判定される場合が散見された. そこで, 学習したコード生成モデルが Math-Verify で正解と判定されるようなフォーマットを優先して出力するようなコードを生成させることを目指して, Group Relative Policy Optimization (GRPO) [7] による強化学習を実施した.

**データセット** 第一段階 SFT のデータセットに含まれる問題文の一部に対して, 第一段階までの学習を行なった LLM<sub>code</sub> を用いて推論を行なった. その出力結果を gpt-oss-20b を用いてデータフィルタリングと同様の方法 (§ B.1) で LLM-as-a-Judge による正誤判定を実施した. 同時に, Math-Verify による正誤判定も実施した. その結果を元に, モデルの解答自体は (LLM-as-a-Judge による判定で) 正しいが Math-Verify による自動判定では不正解とされるような問題集合を抽出し学習データセットとした. 構築したデータセットをデータセットのクリーニングの際に付与した問題分野のアンोटーション (§ B.2) を元にサブセットに分割し, それぞれ独立に学習を

**表 1** データセットのフィルタリング/クリーニング処理の内容。“種類”の列はフィルタリング(データセットの事例を削除する)か、クリーニング(事例の内容を修正する)かの種別を示す。また、“第一段階”、“第二段階”の列はそれぞれの段階の SFT で適用したフィルタリング/クリーニング処理である箇所に ✓ を記載している。

	種別	第一段階	第二段階	概要
LLM-as-a-Judge	フィルター	✓	✓	gpt-oss-120b によりコードの実行結果と解答を比較し、一致するもののみを残す
分野フィルタリング	フィルター	✓		ルールベースまたは gpt-oss-20b を用いて問題の属する分野 (e.g., 数学, 物理, ...) の分類を実施し、物理、化学分野のサンプルを削除
コードの複雑度フィルタリング	フィルター	✓	✓	生成されたコードのサイクロマティック複雑度 [8] が 30 を超えるサンプルを削除
証明問題のフィルタリング	フィルター	✓		ルールベースで証明問題のサンプルを削除
Math-Verify のパースの可否	フィルター	✓	✓	Math-Verify [9] の parse 関数でパースに成功するサンプルのみを残す
Math-Verify による判定	フィルター	✓		Math-Verify を用いて、元の解答をコードの実行結果の正誤判定を実施し、不正解のサンプルを削除
長さフィルタ	フィルター	✓	✓	標準出力の長さが 100 文字を超えるサンプルを削除
解答のハードコード削除	フィルター	✓		gpt-oss-20b により、解答が print 関数の中で直書きされているサンプルを判別し削除
重複除去	フィルター	✓		MinHash [10] を用いた重複除去
標準出力内容の正規化	クリーニング	✓	✓	標準出力の内容に全角の括弧やスペースが含まれないように正規化
コードの正規化	クリーニング	✓	✓	Python のリンター、コードフォーマッターである Ruff [11] を用いて、import 順などを正規化

行なった。

**目的関数** GRPO の目的関数は  $\mathcal{J}(\theta)$  以下で与えられる：

$$\mathcal{J}(\theta) = \mathbb{E}_{q \sim P(Q), \{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(O|q)} \left[ \frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \left\{ \begin{aligned} & \min \left( r_{i,t}(\theta) \hat{A}_{i,t}, \right. \\ & \left. \text{clip}(r_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t} \right) \\ & \left. - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right\} \right], \quad (1) \end{aligned}$$

ただし、

$$r_{i,t}(\theta) = \frac{\pi_{\theta}(y_{i,t} | q, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} | q, y_{i,<t})}. \quad (2)$$

ここで、 $q$  は入力問題文、 $P(Q)$  は問題分布、 $y_i = (y_{i,1}, \dots, y_{i,|y_i|})$  は方策  $\pi_{\theta_{\text{old}}}$  から入力  $q$  に対して生成された  $i$  番目の出力系列を表す。 $G$  は各入力に対して生成する出力系列数、 $y_{i,t}$  は  $i$  番目の系列の  $t$  番目のトークン、 $y_{i,<t}$  は時刻  $t$  以前の部分系列を表す。また、 $\hat{A}_{i,t}$  はトークン  $y_{i,t}$  に対応するアドバンテージの推定値、 $\epsilon$  はクリッピング係数、 $\beta$  は参照方策  $\pi_{\text{ref}}$  からの乖離を抑制する KL 正則化項である。

**報酬設計** 具体的には、 $\text{LLM}_{\text{code}}$  が生成したコード  $c$  を Python インタプリタで実行し、その最終出

力  $o_c$  に基づいて報酬を設計した。生成コードから得られた最終出力を  $\hat{o}$ 、正解文字列を  $o^*$  とする。報酬は、コードの実行の成功に基づいてゲートした上で、正解文字列との正規化されたレーベンシュタイン距離に基づいて与える：

$$r(c, \hat{o}, o^*) = g(c) s(\hat{o}, o^*). \quad (3)$$

ここで、 $g(c) \in \{0, 1\}$  は実行の成功に基づくゲート関数であり、

$$g(c) = \begin{cases} 1 & \text{実行したコード } c \text{ が正常終了,} \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

である。また、 $s(\hat{o}, o^*)$  は出力文字列と正解文字列の一致度を表すスコアであり、

$$s(\hat{o}, o^*) = \begin{cases} 1 & \hat{o} \equiv o^*, \\ 1 - \frac{d_{\text{lev}}(\hat{o}, o^*)}{\max(|\hat{o}|, |o^*|)} & \text{otherwise,} \end{cases} \quad (5)$$

と定義する。ここで、 $d_{\text{lev}}(\hat{o}, o^*)$  は  $\hat{o}$  と  $o^*$  のレーベンシュタイン距離を表す。また、 $\hat{o} \equiv o^*$  は Math-Verify での正解判定を表す。したがって、実行に失敗した場合の報酬は 0 となり、実行に成功した場合のみ、正解との近さに応じて  $[0, 1]$  の範囲で報酬が与えられる。学習は Low-Rank Adaptation (LoRA) [15] を用いた。各サブセットごとに独立して学習を行い、学

習した LoRA の重みをベースモデルに全てマージすることで LLM-jp-v4 8B (GRPO) (図 2) を構築した。詳細な学習設定は § C を参照されたい。

### 3.5 モデルマージとアンサンブル

**ロジット平均アンサンブル** 複数モデルの出力を統合するために、各モデルの出力ロジットを平均するアンサンブルを採用した。具体的には、 $K$  個のモデル  $\{\text{LLM}^{(k)}\}_{k=1}^K$  に対して、トークン  $v$  の次トークン確率を以下のように計算する。

$$P(v | \mathbf{x}) = \text{softmax}\left(\frac{1}{K} \sum_{k=1}^K \mathbf{z}^{(k)}(\mathbf{x})\right)[v]. \quad (6)$$

ここで、 $\mathbf{x}$  は入力トークン列、 $\mathbf{z}^{(k)}(\mathbf{x}) \in \mathbb{R}^{|V|}$  はモデル  $k$  が出力する語彙全体に対するロジットベクトルである。

**モデルマージ** 二段階の SFT で学習したそれぞれのモデル (図 2 における LLM-jp-v4 8B (Stage1), LLM-jp-v4 8B (Stage2)) と GRPO を適用したモデル (LLM-jp-v4 8B (GRPO)) の重みの算術平均をとるモデルマージ [16] を行い LLM-jp-v4 8B (Merged) を構築した。

### 3.6 後処理

本コンペティションの評価指標である math-eval [14] は、特定の数式表現において意味的に正しい場合であっても、正解と判断されない場合が散見された。そこで、本システムでは、モデルが出力した解答文字列 (LaTeX 数式) に対して、ルールベースの後処理を適用し、表記ゆれの吸収と評価指標との整合を図った。具体的な後処理の内容を以下に列挙する。

- **不等式の整形**:  $\backslashwedge$  (「かつ」) を使用しない表現に変換 (例:  $x < 2 \backslashwedge x > -3 \rightarrow -3 < x < 2$ )。
- **二次方程式の解の整形**: 問題文から二次方程式であることを判定し、解が 2 つのときは可能な限り集合 (中括弧) ではなく、 $\pm$  形式に正規化。
- **座標の整形**: 座標を求める問題であることを判定し、解を丸括弧  $(x, y)$  に統一 (角括弧や行列形式を変換)。
- **小数の正規化**: 問題文に丸め指示があればそれに従い、そうでなければ条件付きで分数へ有理化。
- **対数表記の正規化**: 問題文に  $\backslashln$  が無い場合は  $\backslashln$  を  $\backslashlog$  に置換。
- **係数と  $\backslashfrac{\backslashsqrt}$  の空白除去**:  $2\sqrt{3}$  のように係数と根号の間の空白を除去、ただし  $-\sqrt{3}$  のような

マイナス直後の空白は維持。

- **数式区切りの付与**: 最終結果を  $\$...\$$  で囲んで返す。付与する数式区切り以外の  $\$$  は削除。

## 4 リランキングモジュール

リランキングモジュールは、解答モジュールによって生成されたコードと解答のペア集合を入力として受け取り、それらの中から最も尤もらしい解答を選択するモジュールである。このモジュールは、正誤判定予測モデルと、候補間の合意最大化による選択アルゴリズムを組み合わせ、信頼性の高い解答の選択を目指した。

### 4.1 正誤判定予測モデル

正誤判定予測モデルは、問題文とコードを入力として受け取り、そのコードが正しい解法であるか否かの二値分類を行うように追加学習を実施した llm-jp-v4-8B である。

**データセット構築** § 3.1 の第一段階 SFT で作成したデータセットと同様の手順で構築した開発段階のデータセットで学習したモデルを用い、その学習データの全てのサンプルに対して推論を行った。推論時には 4 つの異なる temperature (0.0, 0.2, 0.4, 0.8) を指定し 1 問あたり 4 つのコードを生成した。その推論結果に対して gpt-oss-120b を用いた LLM-as-a-judge による正誤判定 (§ B.1 と同様) と Math-Verify による正誤判定を実施し、いずれの手法でも正解と判定されたコードを正答、いずれの手法でも不正解と判定されたコードを誤答とした。最終的にモデルが正答したコード、誤答したコードそれぞれ 1,080,610 件からなる合計 2,161,220 件のデータセットを構築した。このデータセットを用いて、llm-jp-v4-8B を追加学習し正誤判定予測モデル  $\text{LLM}_{\text{judge}}$  を構築した。詳細は § D を参照されたい。

**学習設定** 構築したデータセットを用いて、llm-jp-v4-8B を初期値とした SFT を実施した。学習時のハイパーパラメータは § C に記載する。

### 4.2 候補間の合意最大化による選択

**他候補との一致度の抽出** まず、各解答候補間の一致度を表す行列  $M \in \mathbb{R}^{n \times n}$  を定義し、各要素は

$$M_{i,j} = \begin{cases} \text{sim}(o_i, o_j) & (i \neq j) \\ 0 & (i = j) \end{cases} \quad (7)$$

で与える. ここで  $\text{sim}(o, o')$  は2つの解答間の一致度を表す関数であり, `math-eval` を用いて計算する. 具体的には, `strict`・`complex`・`soft` の3つの検証モードを順に適用し, `strict` モードで一致と判定された場合は 1.0, `complex` モードで一致と判定された場合は 0.5, `soft` モードで一致と判定された場合は 0.2 を返す. いずれのモードでも一致と判定されない場合は 0.0 とする.

**解答間合意スコア** 候補解答  $o_i$  が他の候補とどの程度一致しているかを定量化するため, 以下の合意スコア  $s_i^{\text{agree}}$  を定義する:

$$s_i^{\text{agree}} = \sum_j M_{i,j}. \quad (8)$$

この定義は, 複数のサンプルから得られた解答間の一致度を期待効用として評価する `sample-based MBR` [17, 6] の考え方と整合的である.

### 4.3 リランキングスコア

最終的なリランキングのスコアは, 上記の**正誤判定予測モデル**と**他候補との一致度**を組み合わせたスコアを計算する. 具体的には, 以下の3要素から構成される.

- 解答合意スコア (§ 4.2)
- 正誤判定予測 (§ 4.1)
- 正誤判定予測で補正した解答合意スコア

**正誤判定予測で補正した解答合意** 信頼性の高い解答による支持を重視する形で合意度を補正することを考える. 各候補解答  $o_i$  には, § 4.1 で作成した正誤判定器の予測  $c_i$  (正答 1, 誤答 0) が与えられる. これを用いて以下のスコアを定義する:

$$s_i^{\text{support}} = \sum_j M_{i,j} c_j. \quad (9)$$

**最終スコア** 以上を組み合わせ, 各候補解答の最終スコア  $S_i$  を

$$S_i = s_i^{\text{agree}} + \gamma(n-1)c_i + \lambda s_i^{\text{support}} \quad (10)$$

と定義する.  $\lambda$ ,  $\gamma$  はハイパーパラメータである. 本システムでは,  $\lambda = 4.0$ ,  $\gamma = 0.8$  を用いた. この値は, 開発データセット上でのハイパーパラメータ探索の結果に基づいて選択した.

**キュー優先度の計算** アルゴリズム 1 において, 各問題をパイプラインで処理する際の優先度を決

定するために, 優先度付きキューを用いる. この優先度は, リランキング時のあるサンプルに対する理論上の最大値  $S_i^{\text{max}}$  に対する実際のスコア  $S_i$  の比  $p_{\text{score}} = \frac{S_i}{S_i^{\text{max}}}$  で与えられる. この値が低いほど優先的に, 次の推論及びリランキング処理が行われる.

## 5 提出システムの構成

最終的に提出したシステムでは図 2 における, `llm-jp-v4-8B (Stage1)` を 4 モデル, `LLM-jp-v4 LLaMA Pro (Stage1)` を 1 モデル, `LLM-jp-v4 8B (Merged)` を 1 モデル, `LLM-jp-v4 8B (Ensemble)` を 1 モデル, そして正誤判定予測モデルを 1 モデルの各 GPU に独立して配置し推論を行なった. 500 分の制限時間内で可能な限り候補の生成とリランキングを繰り返し(アルゴリズム 1), 制限時間を迎えた時点での結果を最終的なシステムの出力とした. モデルの推論には `vLLM` [18] を用いた.

## 6 事後評価

本コンペティション終了後, 本システムに含まれる各要素の効果を検証するため比較実験を実施した.

### 6.1 実験設定

評価対象のモデルとして図 2 に示すモデルのうち, 提出システムに含まれているモデルである `LLM-jp-v4 8B (Stage1)`, `LLM-jp-v4 LLaMA Pro (Stage1)`, `LLM-jp-v4 8B (Merged)`, `LLM-jp-v4 8B (Ensemble)` の性能を独立に評価した. また, 最終的な提出システム (Submission system) の性能も評価した. 加えて, 対照実験として以下のモデル/設定についても評価を行なった.

- **LLM-jp-v4 8B (Merged) + Rerank**: `LLM-jp-v4 8B (Merged)` に 8 つの候補を生成させ, それらを `Judge-Enhanced MBR` でリランキングする.
- **gpt-oss-120b**: 解答コードの生成に利用した教師モデル.
- **LLM-jp-v4 8B (Stage1, w/o thinking)**: 思考過程を学習データに含めずに学習を行なったモデル.

評価時には, 基本的に各テスト事例に対して 1 回の解答生成を行なった. なお, 最終システムについてのみアルゴリズム 1 に従い, 提出時と同様に可能な限り多くの候補生成とリランキングを繰り返した場

**表 2** 事後評価結果. Submission system (1 cycle) は各問題に対して一度だけ、コード生成とリランキングを行った場合の結果である. Submission system (5 hours) は制限時間を 5 時間として可能な限り、コード生成とリランキングを繰り返した場合の結果である (アルゴリズム 1)

システム/モデル	math-eval↑
a. LLM-jp-v4 8B (Stage1)	86.75
b. LLM-jp-v4 8B (Merged)	92.57
c. LLM-jp-v4 8B (Merged) + Rerank	<b>93.98</b>
d. LLM-jp-v4 LLaMA Pro (Stage1)	89.96
e. LLM-jp-v4 8B (Ensemble)	90.96
f. Submission system (1 cycle)	89.16
g. Submission system (5 hours)	91.16
h. LLM-jp-v4 8B (Stage1, w/o thinking)	78.31
i. gpt-oss-120b	92.57

合の評価を行なった<sup>4)</sup>.

評価データセットには、本コンペティションの最終評価に利用された 498 件の日本語で記述された数学問題とその解答からなるデータセットを用いた. 評価指標には math-eval [14] を用いた<sup>5)</sup>.

## 6.2 結果

評価結果を表 2 に示す.

**各構成要素の有用性** 第二段階の SFT/GRPO を導入することで性能の向上が確認され (a. vs. b.), 教師モデルである gpt-oss-120b と同等の性能を達成した. さらに, リランキングモジュールの導入により, 1.23 ポイントの性能向上が得られた (b. vs. c.) ことから, その有効性が示された. 加えて, LLaMA Pro によるブロック拡張手法 (§ 3.2) を適用した場合, 3.21 ポイントの性能向上が確認された (a. vs. d.). 一方で, ロジット平均によるアンサンブル (e.) の性能は, アンサンブル対象である二つのモデル (a., b.) の中間にとどまった. また, 提出システム全体の性能は, 単一の最高性能モデルに対してリランキングを適用した設定 (d.) を下回った (c. vs. f.). このことから, モデルの組み合わせ方には改善の余地があると考えられる.

**繰り返し推論の有用性** アルゴリズム 1 に示したように, 提出システムでは制限時間内で可能な限り候補生成とリランキングを繰り返す推論戦略を採用した. この戦略の有用性を評価するため, 提出シス

4) 利用可能な計算リソースの制約から, 制限時間を 300 分として Nvidia H100 80GB GPU を 8 枚用いて推論を行なった. そのため, 最終的なコンペティションのスコアとは乖離がある可能性があることに注意されたい.

5) math-eval の実装は提出締め切り後に変更が加えられており, 提出したシステム中で利用した math-eval は厳密には異なるものであることに注意されたい.

テムについて, 候補生成とリランキングを 1 回のみ行う場合と, 繰り返し推論を行う場合とで性能を比較した (f. vs. g.). その結果, 時間制限内で繰り返し推論を行った場合には 2.00 ポイントの性能向上が確認された. 以上の結果は, 繰り返し推論を行うことの有用性を示している.

**思考過程の有用性** 対照実験として, gpt-oss-120b から得られた思考過程 (*thinking*) を学習データに含めずに学習したモデルを構築し, その性能を評価した (h.). その結果, 思考過程を学習データに含めた場合と比較して (a. vs. h.), 8.43 ポイントの性能向上が確認された. この結果は, 思考過程を学習データに含めることが, 本システムにおいて有効であることを示している.

## 6.3 システムの限界

本システムでは, 最終的な解答を, 生成したコードを Python インタプリタで実行することで得る. また, 生成したコードの多くは, 記号計算ライブラリである SymPy [19] を利用している. そのため, 本システムが出力する解答の表現形式は, SymPy に大きく依存する. この影響により, コードの処理内容自体は妥当であっても, 最終的に得られる数式表現が, 高校数学で通常用いられる形としては不自然になる場合が定性的に見られた. 具体例を リスト 1 に示す. 問題例 1 では, コードの処理自体は妥当であるが, 実行結果は  $\sqrt[6]{-1}(1+2i)$  のように 6 乗根を含む形で表される. この表現は SymPy の複素数の主値に基づくものとしては整合的である一方, 高校数学で通常用いられる表現としては不自然である<sup>6)</sup>. 問題例 2 においても, 不定積分を求める処理自体は妥当であるが, 実行結果は  $C + \log(x)$  となり,  $\log(x)$  の  $x$  に絶対値が付されていない. このため, 高校数学の文脈では正答とみなされない可能性がある.

## 7 おわりに

本レポートでは, チーム dentaku の FT-LLM 2026 (数学タスク) における提出システムについて解説した. 我々は基本戦略として Python コードによる解法生成を採用した. 入力として数学問題を受け取り, Python のコードとしてその解法を生成するように llm-jp-v4-8B を追加学習した. 学習データセットの構築には主に gpt-oss-120b を用い, 思考過程を含

6) この現象は SymPy の `expand_complex` 関数を利用することで回避できる可能性がある.

---

**Listing 1** 実行結果が自然な表現とならない例.

---

**問題例 1:** 複素数平面上の点  $z = 1 + 2i$  を原点を中心  
に反時計回りに  $\frac{\pi}{6}$  回転させた点を求めなさい。

**Python コード:**

```
1 # Variable declaration
2 from sympy import I, exp, latex, pi, simplify
3
4 # Original complex number
5 z = 1 + 2 * I
6
7 # Rotation angle (counter^^e2^^80^^91clockwise)
8 theta = pi / 6
9
10 # Rotate the point about the origin
11 z_rot = simplify(z * exp(I * theta))
12
13 # Output the answer with the LaTeX representation
14 latex_answer = latex(z_rot)
15 print(f"Answer: {latex_answer}")
```

実行結果:  $\sqrt{-1}(1 + 2i)$

---

**問題例 2:**  $\frac{1}{x}$  の不定積分を求めよ。ただし、積分定  
数を  $C$  とする。

**Python コード:**

```
1 # Variable declaration
2 from sympy import integrate, latex, symbols
3
4 x = symbols("x")
5 C = symbols("C")
6
7 # Indefinite integral of 1/x
8 F = integrate(1 / x, x) + C
9
10 # Output the answer with the latex representation
11 latex_answer = latex(F)
12 print(f"Answer: {latex_answer}")
```

実行結果:  $C + \log(x)$

---

た。また、定性分析の結果、コードの実行結果が人間にとって不自然と考えられる事例も確認された。そのため、本タスクにおける今後の改善の方向性としては、言語モデル自体の性能向上に加え、記号計算ライブラリの改善も重要であると考えられる。

む出力全体を模倣する形で知識蒸留を行った。また、追加学習を施した複数の llm-jp-v4-8B の出力から、より尤もらしいものを選択するリランキングモジュールを構築し、制限時間内で推論とリランキングを繰り返すパイプラインを設計した。コンペティション終了後に実施した事後評価の結果、学習後の llm-jp-v4-8B は、教師モデルである gpt-oss-120b と同等の性能を達成した。一方で、複数モデルを組み合わせた提出システムの性能は、最高性能の単一モデルを下回る結果となり、複数モデルの出力の統合方法には改善の余地があることが明らかになっ

## 謝辞

本研究は JSPS 科研費 JP25KJ0615, JP25K03175, 24K16077; JST SPRING JPMJSP2114; JST BOOST JPMJBS2412, JPMJBS2421; 中島記念国際交流財団, 文部科学省の補助事業「生成 AI モデルの透明性・信頼性の確保に向けた研究開発拠点形成」の助成を受けたものです。本研究の一部は九州大学情報基盤研究開発センター研究用計算機システムの「一般利用」を利用した。終わりに, 本研究を進めるにあたり多くの協力を賜りました Tohoku NLP グループの皆様にご感謝申し上げます。また, 本コンペティションを運営し貴重な機会を提供して下さった主催者の皆様にも深く感謝申し上げます。

## 貢献

- ・ 工藤慧音: データセット構築, 学習補助, 各コンポーネントの統合
- ・ 日高雅俊: パイプラインの実装
- ・ 塩野大輝: 学習 (SFT)
- ・ 鴨田豪: リランキング, 後処理
- ・ 葉夢宇: 学習 (SFT) 評価・強化学習 (GRPO)
- ・ 高橋良允: データセット構築
- ・ 青木洋一: データセット構築
- ・ 矢野一樹: LLaMA Pro, モデルアンサンブルの実装
- ・ 谷口雅弥: アドバイザー, インフラ整備
- ・ 伊藤拓海: アドバイザー, 後処理
- ・ 坂口慶祐: アドバイザー
- ・ 鈴木潤: アドバイザー

## 参考文献

- [1] FT-LLM. 大規模言語モデルのファインチューニング技術と評価: Fine-tuning and evaluation of large language models (ft-llm). <https://llm-jp.github.io/tuning-competition/>, 2025. Accessed: 2026-03-07.
- [2] OpenAI. gpt-oss-120b & gpt-oss-20b model card. **CoRR**, Vol. abs/2508.10925, , 2025.
- [3] Akiko Aizawa, Eiji Aramaki, Bowen Chen, Fei Cheng, Hiroyuki Deguchi, Rintaro Enomoto, Kazuki Fujii, Kensuke Fukumoto, Takuya Fukushima, et al. Llm-jp: A cross-organizational project for the research and development of fully open japanese llms. **arXiv preprint arXiv:2407.03963**, 2024.
- [4] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: program-aided language models. In **Proceedings of the 40th International Conference on Machine Learning, ICML'23**. JMLR.org, 2023.
- [5] Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. **Transactions on**

**Machine Learning Research**, 2023.

- [6] Patrick Fernandes, António Farinhas, Ricardo Rei, José G. C. de Souza, Perez Ogayo, Graham Neubig, and Andre Martins. Quality-aware decoding for neural machine translation. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, **Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**, pp. 1396–1412, Seattle, United States, July 2022. Association for Computational Linguistics.
- [7] Qihao Zhu Runxin Xu Junxiao Song Mingchuan Zhang Y.K. Li Y. Wu Daya Guo Zhihong Shao, Peiyi Wang. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.
- [8] Thomas J. McCabe. A complexity measure. **IEEE Transactions on Software Engineering**, Vol. SE-2, No. 4, pp. 308–320, December 1976.
- [9] Hynek Kydlíček. Math-verify: Math verification library.
- [10] A.Z. Broder. On the resemblance and containment of documents. In **Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)**, pp. 21–29, 1997.
- [11] Astral Software Inc. Ruff. Python linter and code formatter.
- [12] Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ying Shan, and Ping Luo. LLaMA pro: Progressive LLaMA with block expansion. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 6518–6537, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [13] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, et al. gpt-oss-120b & gpt-oss-20b Model Card. **arXiv preprint arXiv:2508.10925**, 2025.
- [14] llm-jp. math-eval, 2026. Evaluation tool for the math task in FT-LLM-2026. Accessed: 2026-03-08.
- [15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In **International Conference on Learning Representations**, 2022.
- [16] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. **ACM Comput. Surv.**, Vol. 58, No. 8, February 2026.
- [17] Bryan Eikema and Wilker Aziz. Is MAP Decoding All You Need? The Inadequacy of the Mode in Neural Machine Translation. In Dornia Scott, Nuria Bel, and Chengqing Zong, editors, **Proceedings of the 28th International Conference on Computational Linguistics**, pp. 4506–4520, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.
- [18] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In **Proceedings of the 29th Symposium on Operating Systems Principles, SOSP '23**, p. 611–626, New York, NY, USA, 2023. Association for Computing Machinery.
- [19] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K. Moore, et al. Sympy: symbolic computing in python. **PeerJ Computer Science**, Vol. 3, p. e103, January 2017.
- [20] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, et al. Training verifiers to solve math word problems. **CoRR**, Vol. abs/2110.14168, , 2021.
- [21] Wei Du, Shubham Toshniwal, Branislav Kisacanian, Sadegh Mah-

- davi, Ivan Moshkov, George Armstrong, Stephen Ge, Edgar Minasyan, Feng Chen, and Igor Gitman. Nemotron-math: Efficient long-context distillation of mathematical reasoning from multi-mode supervision. **arXiv preprint arXiv:2512.15489**, 2025.
- [22] Yuichi Tateno. Fineweb2 edu japanese, February 2025.
- [23] Yifan Zhang. StackMathQA: A Curated Collection of 2 Million Mathematical Questions and Answers Sourced from Stack Exchange. Technical report, Math-AI, 2024.
- [24] LLM-jp. llm-jp-corpus-v4. <https://gitlab.llm-jp.nii.ac.jp/datasets/llm-jp-corpus-v4/-/blob/main>, June 2025.
- [25] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, et al. Qwen2.5 Technical Report. **arXiv [cs.CL]**, 19 December 2024.
- [26] Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. Fineweb2: One pipeline to scale them all – adapting pre-training data processing to every language, 2025.
- [27] DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025.
- [28] GLM. GLM-4.5: agentic, reasoning, and coding (ARC) foundation models. **CoRR**, Vol. abs/2508.06471, , 2025.
- [29] Z.ai. Glm-4.6: Advanced agentic, reasoning and coding capabilities with an open 355b moe model. <https://z.ai/blog/glm-4.6>, 2025. Accessed: 2026-03-09.
- [30] Qwen Team. Qwen3 technical report. **CoRR**, Vol. abs/2505.09388, , 2025.
- [31] OpenAI. Introducing gpt-oss-safeguard. <https://openai.com/index/introducing-gpt-oss-safeguard/>, October 2025. Accessed: 2026-03-09.
- [32] Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr 2: Visual causal flow. **CoRR**, Vol. abs/2601.20552, , 2026.
- [33] Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. Evaluating the performance of large language models on gaokao benchmark. 2024.
- [34] マスあくありうむ. 【高校数学】例題 & 問題集. <https://web.math-aquarium.jp/>, n.d. Accessed: 2026-02-27.
- [35] Naoaki Okazaki. Doubri: C++ implementation of deduplication. <https://github.com/swallow-llm/doubri>, 2024.
- [36] Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In **First Conference on Language Modeling**, 2024.
- [37] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In **International Conference on Learning Representations**, 2019.

## A データセットの詳細

### A.1 第一段階 SFT

表 3 に第一段階 SFT で利用したデータセットの一覧を示す。図 3 に示した通り、グループ 1 はデータセット内の英語の問題文を日本語に翻訳して利用した。グループ 2 は、データセットに含まれる数学関連文書を gpt-oss-120b を用いて、日本語の数学の問題へ書き換えを行った。以下ではそれぞれのデータセットについてその内容と利用方法について説明する。

**GSM8K (グループ 1)** GSM8K [20] は人手で作成された約 8500 件の小学校教育レベルの英語で記述された算数問題とその解答からなるデータセットである。本システムの構築にあたっては、英語で記述された問題文を gpt-oss-120b を用いて日本語に翻訳して利用した。英日翻訳に利用したプロンプト<sup>7)</sup>は GitHub を参照されたい。

**nvidia/Nemotron-Math-v2 (グループ 1)** nvidia/Nemotron-Math-v2 は gpt-oss-120b 及び Qwen2.5-32B-Instruct [25] を用いて構築された数学問題のデータセットである。本システムの構築にあたっては、このデータセットの StackExchange-Math サブセットのみを利用した。このサブセットは、StackExchange データダンプのポリシーが変更される 2024 年 7 月以前にリリースされたデータを元に構築されており、CC BY-SA-4.0 ライセンスのもとで利用可能なデータのみから構成されている<sup>8)</sup>。GSM8K と同様に、gpt-oss-120b を用いて英語で記述された問題文を日本語に翻訳して利用した。

**math-ai/StackMathQA (グループ 2)** fineweb-2-edu-japanese と同様に gpt-oss-120b を用いて日本語の数学の問題に書き換えることで、数学の問題と解答のペアからなるデータセットを構築した。利用したプロンプトは hotchpotch/fineweb-2-edu-japanese 整形時と同一である。

**hotchpotch/fineweb-2-edu-japanese (グループ 2)** FineWeb2 (ライセンス: ODC-By) [26] を元データ

セットとして、DeepSeek を知識蒸留して作成した分類器によるデータセットフィルタリングが適用されたデータセットである<sup>9)</sup>。我々は gpt-oss-20b を用いて、このデータセットから数学に関連する文書を抽出し、さらにそれらの文書を gpt-oss-120b を用いて日本語の数学の問題に書き換えることで、日本語の数学の問題データセットを構築した。数学関連文書の抽出に利用したプロンプト<sup>10)</sup>、数学関連文書の問題文への書き換えに利用したプロンプト<sup>11)</sup>は GitHub を参照されたい。

**llm-jp-corpus-v4 (グループ 2)** fineweb-2-edu-japanese と同様に数学に関連する文書を gpt-oss-20b を用いて抽出し、さらにそれらの文書を gpt-oss-120b を用いて日本語の数学の問題に書き換えることで、数学の問題と解答のペアからなるデータセットを構築した。本システムの構築にあたっては、ja\_wiki, ja\_kaken サブセットのみを利用した。利用したプロンプトは hotchpotch/fineweb-2-edu-japanese 整形時と同一である。

**合成データセット (グループ 3)** はじめに DeepSeek-R1-0528 [27] を用いて 562 件の学習項目リストを構築した。学習項目の例を表 4 に示す。この学習項目リストをもとに、複数の言語モデルを用いて日本語の数学の問題と解答のペアからなる初期問題集合を構築した。利用した言語モデルは、gpt-oss-120b に加えて GLM-4.5-Air [28], GLM-4.6 [29], Qwen3-235B-A22B-Thinking-2507 [30], gpt-oss-safeguard-120b [31] である。また、学習項目リストをもとに生成した初期問題集合に加えて、第二段階 SFT データセットで利用する事例の一部(図 3)を初期問題集合として、gpt-oss-120b を用いて問題文と解答のペアをさらに生成した。データ拡張のアル

9) HuggingFaceFW/fineweb-edu は Llama3 を用いているため、Llama Community License を継承していると考えられるため、本コンペティションでの使用は禁止されている。一方で、hotchpotch/fineweb-2-edu-japanese は FineWeb2 をもとに Llama3 を用いることなく構築されているため、Llama Community License を継承していない。

10) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math\\_related\\_text\\_classification\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math_related_text_classification_system_prompt.md)

ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math\\_related\\_text\\_classification.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math_related_text_classification.md)

11) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/rewrite\\_math\\_problem\\_gen\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/rewrite_math_problem_gen_system_prompt.md)

ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/rewrite\\_math\\_problem\\_gen\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/rewrite_math_problem_gen_prompt.md)

7) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/translate.en-ja.system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/translate.en-ja.system_prompt.md)

ユーザプロンプト: <https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/translate.en-ja.md>

8) <https://nvidia-nemo.GitHub.io/Skills/releases/nemotron-math-v2/>

表3 第一段階 SFT で利用したデータセット一覧.

	ライセンス	使用方法
GSM8K [20]	MIT	グループ 1
nvidia/Nemotron-Math-v2 (StackExchange サブセット) [21]	CC BY-SA 4.0	グループ 1
hotchpotch/fineweb-2-edu-japanese [22]	ODC-By, CommonCrawl Terms of Use	グループ 2
math-ai/StackMathQA [23]	CC BY 4.0	グループ 2
llm-jp-corpus-v4/ja/ja_wiki [24]	CC BY-SA 3.0	グループ 2
llm-jp-corpus-v4/ja/ja_kaken [24]	CC BY 4.0	グループ 2

表4 学習項目リストの一部.

項目タイトル	内容	カテゴリ
基本概念・定義 — 約分・整数部分・小数表示	分子・分母に共通因数があれば約分し, $[x]$ で整数部分, $\{x\} = x - [x]$ で小数部分を表す。	数学 I: 数と式
積分範囲の変換 (逆順・対称性)	奇関数は $\int_{-a}^a f(x) dx = 0$ , 偶関数は $\int_{-a}^a f(x) dx = 2 \int_0^a f(x) dx$ 。これを利用して計算を簡略化。	数学 II: 積分
数列の和・積の変形テクニック (部分和の置換)	例えば $\sum_{k=1}^n k \cdot 2^k$ は, $2 \sum_{k=1}^n k 2^{k-1}$ と置換し, 帰納的に求める。	数学 B: 数列
放物線の標準形と焦点・準線	頂点 $(h, k)$ , 軸の向きに応じて $(y-k) = \frac{1}{4p}(x-h)^2$ (水平) や $(x-h) = \frac{1}{4p}(y-k)^2$ (垂直) で表し, 焦点は $(h+p, k)$ , 準線は $x = h-p$ (水平の場合)。	数学 C: 平面上の曲線
行列の転置と逆行列の関係 (対称行列の性質)	対称行列は $A^T = A$ で, 可逆なら $(A^{-1})^T = (A^T)^{-1} = A^{-1}$ が成り立つ。	数学 C: 行列

ゴリズムの概要をアルゴリズム [アルゴリズム 2](#) に示す。gpt-oss-120b に提示した問題文を手がかりに, 内容の異なる問題文を作るよう指示し, 問題文と解答のペアを生成する工程を繰り返した。gpt-oss-120b に与えるプロンプトは [GitHub](#) を参照されたい<sup>12)</sup>。

## A.2 第二段階 SFT

**WikiBook** Wikibooks 日本語版<sup>13)</sup> に公開されている中学・高校数学の教科書的記述から数学問題を抽出して構築したデータセットである。ライセンスは CC BY-SA 4.0 である。SFT データセットの構築には, gpt-oss-120b を用いて問題文・解法・解答を構造化された JSONL 形式で抽出した。必要に応じて抽出された問題に対しては人手による修正を実施し, 最終的に 938 件 (中学数学 610 件, 高校数学 328 件)

の問題・解答ペアを得た。収録されている単元は多岐にわたり, 中学数学では文字と式, 因数分解, 連立方程式など, 高校数学では三角関数, 極限, 対数, 微分などが含まれる。これにより, 中学・高校数学の広範な分野をカバーする高品質なデータセットを構築した。問題の抽出に利用したプロンプトの詳細は [GitHub](#) を参照されたい<sup>14)</sup>。

**犬プリント** 本システムでは, 基礎的な高校数学の内容を網羅的に含むデータセットを構築するため, 個人運営の教育サイト「ようこそ! 犬プリの世界へ」<sup>15)</sup> に掲載されている高校数学の問題および解説資料を収集し, データセットに追加した。具体的には, 当該 Web ページからアクセス可能な数学関連文書 (PDF 形式) を対象としてクロールを行い, DeepSeek-OCR [32] を用いてテキストを抽出した。その後, gpt-oss-120b を用いて問題文となるように

12) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math\\_problem\\_multi\\_gen\\_prompt\\_with\\_similar\\_problem\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math_problem_multi_gen_prompt_with_similar_problem_system_prompt.md)

ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math\\_problem\\_multi\\_gen\\_prompt\\_with\\_similar\\_problem.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math_problem_multi_gen_prompt_with_similar_problem.md)

13) <https://ja.wikibooks.org/wiki/%E6%95%B0%E5%AD%A6>

14) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/wikibook\\_math\\_qa\\_extraction\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/wikibook_math_qa_extraction_system_prompt.md)

ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/wikibook\\_math\\_qa\\_extraction.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/wikibook_math_qa_extraction.md)

15) <https://inupri.web.fc2.com/index.html>

---

**Algorithm 2** データ拡張処理の疑似コード.

---

**Require:** 初期問題集合  $D_{\text{init}}$

gpt-oss-120b: GPToss

**Ensure:** 生成されたデータセット  $D_{\text{gen}}$

```
1: キュー  $Q$  を作成し,  $D_{\text{init}}$  に含まれる全ての問題を  $Q$  に追加する
2:  $D_{\text{gen}} \leftarrow \emptyset$ 
3:
4: while True do
5:    $P_{\text{ref}} \leftarrow Q.\text{Pop}()$ 
6:   ▶  $P_{\text{ref}}$  を参考に問題を生成するプロンプトを作成
7:    $x \leftarrow \text{MakePrompt}(P_{\text{ref}})$ 
8:   ▶ 新しい問題と解答のペアの集合を生成
9:    $S_{\text{new}} \leftarrow \text{GPToss}(x)$ 
10:  for  $(P_{\text{new}}, A_{\text{new}}) \in S_{\text{new}}$  do
11:     $D_{\text{gen}} \leftarrow D_{\text{gen}} \cup \{(P_{\text{new}}, A_{\text{new}})\}$ 
12:     $Q.\text{Push}(P_{\text{new}})$ 
13:  end for
14: end while
15: return  $D_{\text{gen}}$ 
```

---

整形を行った。利用したプロンプトは GitHub を参照されたい<sup>16)</sup>。なお、同サイト内には大学入試の過去問題も掲載されているが、これらは各大学が著作権を保有しているため、本システムのデータセットには含めていない。また、本研究において当該 Web ページ上の PDF ファイルを収集するにあたり、サイト管理者より当該コンペティションでの利用/公開について明示的な許諾を得たうえで収集を行った。

**GAOKAO-Bench** GAOKAO-Bench [33] は中国の大学入学試験（高考/Gaokao）の 2010 年から 2022 年までの過去問題を収録した Apache License 2.0 ライセンスで公開されているベンチマークデータセットである。選択問題や穴埋め問題などの客観式問題 1,781 問と、記述式などの主観式問題 1,030 問の合計 2,811 問が含まれており、数学を含む複数の科目をカバーしている。本システムの構築にあたっては、GAOKAO-Bench から数学に関連する問題 844 問に対し、以下の 2 段階の処理を実施した。まず、中

16) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract\\_multiple\\_math\\_qa\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract_multiple_math_qa_system_prompt.md)  
ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract\\_multiple\\_math\\_qa.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract_multiple_math_qa.md)

国語で記述された問題文を gpt-oss-120b を用いて日本語に翻訳した。翻訳時には数式 (LaTeX 記法) を保持しつつ、gpt-oss-120b の思考モードを活用することで文脈を考慮した高品質な翻訳を実現した。次に、翻訳された問題に対して gpt-oss-120b を用いて各問題を独立した小問題に分解する処理を実施した。具体的には、選択問題については選択肢形式を記述式に変換し、穴埋め問題および記述式問題については複数のステップを含む問題を各ステップごとの独立問題に分解することで、最終的に 2,180 件の独立した問題を含むデータセットを構築した。問題の抽出に利用したプロンプトの詳細は GitHub を参照されたい<sup>17)</sup>。

**【高校数学】例題 & 問題集** 【高校数学】例題 & 問題集 [34] は高校数学の PDF 教材を公開しているサイトである。本ページには「著作権は放棄しないが、自由な利用を許諾する。」と明記されている。我々は前処理として DeepSeek-OCR [32] を用いて PDF ファイルからテキストを抽出した。その後 gpt-oss-120b により抽出したテキストから数学問題とその解答を抽出した。数学問題とその解答の抽出時には vLLM [18] の構造化出力の機能を利用して、json 形式で問題抽出を実施した。利用したプロンプトは GitHub を参照されたい<sup>18)</sup>。

**開発データセット** 本コンペティション運営より配布された 100 件の日本語の数学問題とその解答からなるデータセットである。このデータセットは学習データセットとして利用することも明示的に許可されている。

**Human in the loop** 第一段階のモデルの出力を人手で確認し、その間違いの傾向を定性的に分析した。その分析結果に基づいて、gpt-oss-120b に与える問題作成プロンプトを人手で作成し、問題と解答のペアからなる合成データセットを構築した。具体例としては、

- 群数列の問題

17) 翻訳プロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/translate\\_zh-ja.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/translate_zh-ja.md)

問題抽出プロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract\\_gaokao\\_problems.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract_gaokao_problems.md)

18) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract\\_math\\_qa\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract_math_qa_system_prompt.md)

ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract\\_math\\_qa.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract_math_qa.md)

- 直線の交点を求める問題
- 対数計算を含む不等式の問題

などの問題を生成するようにプロンプトを設計した。

### A.3 その他のデータセット

**センター試験データセット** 本システムの開発にあたり、「ロボットは東大に入れるか」プロジェクト<sup>19)</sup>により公開されている大学入試センター試験の数学問題データセットを開発用データセットとして利用した。このデータセットは1997年から2015年までのセンター試験（数学IA、数学IIB）の本試験および追試験の問題と正答がXML形式で構造化されており、問題文、数式（MathML形式）、画像、正答が含まれている。本システムの構築にあたっては、gpt-oss-120bを用いてXML形式のデータをMarkdown形式に変換し、MathML形式の数式をLaTeX記法に変換した上で解答を問題文中に穴埋めした。その後、変換されたMarkdownファイルから大問に含まれる小問を抽出して各問題が単独で解答可能な形式に整形し、各問題に対する詳細な解法の生成と正答のLaTeX形式への統一を実施した。最終的に1,245件の独立した問題を含むデータセットを構築した。なお、このデータセットは商用利用不可のライセンスであるため、コンペティションのルールに基づき開発データセットとしてのみ利用しており、学習には利用していない。問題抽出に利用したプロンプトの詳細はGitHubを参照されたい<sup>20)</sup>。

## B データセットフィルタリングの詳細

以下では、表1に示したデータセットのフィルタリング/クリーニング処理の一部について補足説明を行う。

### B.1 LLM-as-a-Judge

gpt-oss-120bを用いて、生成されたコードの実行結果が元の解答と一致しているか否かを判定した。これにより誤答として判定された事例をデータセットから削除した。これにより、問題の生成段階またはコードの生成段階のいずれかで誤りがある可能性

19) <https://21robot.org/>

20) Markdown変換プロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/convert\\_xml\\_to\\_markdown.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/convert_xml_to_markdown.md)

問題抽出プロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract\\_center\\_exam\\_problems.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/extract_center_exam_problems.md)

がある事例がデータセットから除去されることを期待している。gpt-oss-120bに与えたプロンプトの詳細はGitHubを参照されたい<sup>21)</sup>。

### B.2 分野フィルタリング

ルールベースまたはgpt-oss-20bを用いて問題の属する分野（e.g., 数学, 物理, ...）の分類を実施した。初めに、問題文に特定のキーワードが含まれているか否かに基づいて、問題がどの分野に属するかの分類を行った。その後、ルールベースでの判定が困難な事例に対しては、gpt-oss-20bを用いて問題の分野を判定した。gpt-oss-20bを用いた判定では、入力として問題文を与えその問題がどの分野に属するかを分類するようにプロンプトを設計した。プロンプトの詳細はGitHubを参照されたい<sup>22)</sup>。これらの分類の結果に基づいて、物理、化学分野の事例をデータセットから削除した。

### B.3 証明問題のフィルタリング

問題文が特定のキーワード（e.g., 証明せよ, 示せ）で終わる問題を証明問題とみなし、これらの事例をデータセットから削除した。

### B.4 解答のハードコードの削除

ソースコード内に解答が直書きされている事例をデータセットから削除した。解答が直書きされているコードの例をリスト2に示す。このコードでは、解答である円の中心座標と半径はコードを実行した結果導出されたものではなく、マジックナンバーとしてコードに直書きされている。このようなコードの場合、計算部分も言語モデルが担うため、その正確性に懸念がある。そのため、解答が直書きされていると判定されたサンプルはデータセットから除外した。直書きされたコードか否かの判定はgpt-oss-20bを用いた二値分類タスクとして実施した。利用したプロンプトの詳細はGitHubを参照さ

21) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/llm\\_as\\_a\\_judge\\_cot\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/llm_as_a_judge_cot_system_prompt.md)

ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/llm\\_as\\_a\\_judge\\_cot.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/llm_as_a_judge_cot.md)

22) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math\\_sub\\_field\\_classification\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math_sub_field_classification_system_prompt.md)

ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math\\_sub\\_field\\_classification.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/math_sub_field_classification.md)

**Listing 2** 解答が直書きされたコード例.

問題：次の円の方程式が与えられている.

$$x^2 + y^2 - 4x - 6y + 9 = 0.$$

この円の中心と半径を求めよ.

**Python コード：**

```
1 # Variable declaration
2 from sympy import symbols, latex
3
4 x, y = symbols('x y')
5
6 # Original equation: x^2 + y^2 - 4x - 6y + 9 = 0
7 expr = x**2 + y**2 - 4*x - 6*y + 9
8
9 # Complete the squares manually
10 # (x - 2)^2 = x^2 - 4x + 4
11 # (y - 3)^2 = y^2 - 6y + 9
12 expr_sq = (x - 2)**2 + (y - 3)**2 - 4 # equivalent
13     ↪ to expr
14
15 # From (x - 2)^2 + (y - 3)^2 = 4 we read the center
16     ↪ and radius
17 center = (2, 3) # (h, k)
18 radius_squared = 4
19 radius = 2 # sqrt(4)
20
21 # Output the answer with the LaTeX representation
22 latex_center = latex(center) # "(2, 3)"
23 latex_radius = latex(radius) # "2"
24 latex_answer = f"{latex_center}, {latex_radius}"
25 print(f"Answer: {latex_answer}")
```

りたい<sup>23)</sup>.

## B.5 重複除去

フィルタリング後のデータセットに対して、MinHash [10] を用いた重複除去を実施した。問題文とコードを連結した文字列を1つの文書とみなし、それらの文書に対して MinHash を計算して重複除去を行った。実装には `doubri` [35] を使い、特徴量抽出における文字  $n$ -gram のサイズを5、1バケットあたりのハッシュ数を10、バケットの総数を22に設定して重複除去を実行した。

## C ハイパーパラメータ

23) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/detect\\_hard\\_coded\\_calculation\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/detect_hard_coded_calculation_system_prompt.md)  
ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/detect\\_hard\\_coded\\_calculation.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/detect_hard_coded_calculation.md)

**表 5** SFT 時の学習設定の詳細.

第一段階 SFT / 正誤判定予測モデル	
バッチサイズ	128
エポック	1
最大学習率	$5.0 \times 10^{-5}$
最小学習率	$1.0 \times 10^{-6}$
学習率スケジューラ	Warmup Stable Decay [36]
学習率ウォームアップ ステップ率	10% (線形に増加)
学習率ディケイステッ プ率	10% (線形に減少)
最適化器	AdamW [37]
Optimizer Config	$\beta_1 = 0.9, \beta_2 = 0.95$
Weight Decay	0.1
Gradient Clipping	1.0
系列長	8,192
第二段階 SFT	
バッチサイズ	16
エポック	1
最大学習率	$1.0 \times 10^{-6}$
最小学習率	0
学習率スケジューラ	Linear Decay
学習率ウォームアップ ステップ率	10% (線形に増加)
最適化器	AdamW [37]
Optimizer Config	$\beta_1 = 0.9, \beta_2 = 0.95$
Weight Decay	0.1
Gradient Clipping	1.0
系列長	8,192

**SFT** SFT 時の詳細な学習設定を表 5 に示す。また学習の効率化のために、複数のサンプルを連結し、最大系列長を超えない範囲で入力系列を作成する Packing 戦略を採用した。

**GRPO** GRPO における詳細な学習設定を表 6 に示す。

## D 正誤判定予測モデルの詳細

`gpt-oss-120b` を用いた正誤判定時の出力テキストを利用した知識蒸留により `LLMjudge` を構築した。`gpt-oss-120b` による正誤判定時に利用したプロンプトについては `GitHub` を参照されたい<sup>24)</sup>。 `LLMjudge` の入出力例を `リスト 3, 4` に示す。

24) システムプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/llm\\_as\\_a\\_judge\\_cot\\_system\\_prompt.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/llm_as_a_judge_cot_system_prompt.md)  
ユーザプロンプト: [https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/llm\\_as\\_a\\_judge\\_cot.md](https://GitHub.com/team-dentaku/tcmath2026-team-dentaku-artifacts/blob/main/prompts/llm_as_a_judge_cot.md)

表 6 GRPO 時の学習設定.

GRPO の設定	
1 入力あたりの生成 ( $G$ )	8
温度	1.1
Top- $p$	1.0
Top- $k$	80
最大生成系列長	8,192
系列長	8,192
ステップ数	500-5500 ステップ
バッチサイズ	8
Gradient Accumulation Steps	2
最適化器	AdamW [37]
学習率	$1.0 \times 10^{-4}$
ウォームアップステップ数率	5%
LoRA ハイパーパラメータ	
Rank ( $r$ )	8
Alpha ( $\alpha$ )	32
Dropout	0.05
バイアス項	なし

### Listing 3 正誤判定予測モデルの入力例

#### # Math Problem Solution Script Evaluation Task

You will be given a **Japanese** math problem and a  
 ↳ piece of Python code intended to solve it.  
 Your goal is to evaluate whether the given code is  
 ↳ capable of correctly solving the problem.

## Input

### Question

```tex

正の数  $(a, b, c)$  が等比数列をなすとする.

\\[

$a, b, c$ ;  $(b=ar, c=ar^2)$

\\]

このとき、積  $(abc=216)$  かつ和  $(a+b+c=21)$  が成り  
 ↳ 立つ.  $a$  の値を求めなさい (ただし  $a < b < c$ )  
 ↳ とする).

```

### Python Code for Solution

```python

# Variable declaration

from sympy import Eq, latex, simplify, solve,  
 ↳ symbols

# Define the ratio  $r$  (positive real)

$r = symbols("r", positive=True, real=True)$

# Equation derived from the conditions:

$21 * r^2 * (1 + r + r^2) = 216$

$eq = Eq(21 * r^2 * (1 + r + r^2), 216)$

# Solve the cubic equation for  $r$

$solutions_r = solve(eq, r)$

# Choose the real positive solution that satisfies

↳  $a < b < c$  (i.e.,  $r > 1$ )

$r\_val = [sol for sol in solutions_r if sol.is\_real$

↳ and  $sol > 1][0]$

# Compute  $a = 6 / (1 + r + r^2)$

$a = simplify(6 / (1 + r\_val + r\_val^2))$

# Output the answer with the LaTeX representation

$latex\_answer = latex(a)$

$print(f"Answer: {latex\_answer}")$

```

<|end|>

---

**Listing 4** 正誤判定予測モデルの出力例

---

```
<|start|>assistant<|channel|>analysis<|message|>We  
↪ need to evaluate if the code correctly solves  
↪ the problem. Let's analyze problem:
```

... [途中省略] ...

Also they compute  $a = 6/(1+r+r^2)$  which is not  
↪ correct;  $a$  should be  $21/(1+r+r^2)$  or  $6/r$ . So  
↪ code is wrong.

Thus decision: Invalid.

We need to produce output with rationale and  
↪ decision.<|end|><|start|>assistant  
↪ <|channel|>final<|message|>

### ## Step by Step Rationale

#### 1. **Understanding the Original Problem**

- The numbers  $(a, b, c)$  form a geometric  
↪ progression:  $(b = ar)$ ,  $(c = ar^2)$  with  
↪  $(r > 0)$ .
- Conditions:
  - \* Product:  $(abc = a \cdot ar \cdot ar^2 =$   
↪  $a^3 r^3 = 216)$ .
  - \* Sum:  $(a + b + c = a(1+r+r^2) = 21)$ .
- From the product we obtain  $(ar = 6)$  (since  
↪  $(ar)^3 = 216$ ).
- Substituting  $(a = 6/r)$  into the sum gives the  
↪ correct equation  
$$\left[ \frac{6}{r}(1+r+r^2) = 21 \right]$$
$$\rightarrow \rightarrow; \rightarrow; 2r^2 - 5r + 2 = 0,$$
$$\left[ \right]$$
whose positive solution larger than 1 is  $(r =$   
↪  $2)$ .
- Consequently  $(a = 6/r = 3)$ .

... [途中省略] ...

#### 4. **Logical Consistency and Case Selection**

- The code attempts to select the real positive  
↪ solution with  $(r > 1)$ , which aligns with the  
↪ condition  $(a < b < c)$ . However, because the  
↪ underlying equation is incorrect, the  
↪ selected  $(r)$  does not satisfy the original  
↪ problem's constraints.

#### 5. **Overall Assessment**

- The code `mis^e2^80^91` derives the governing  
↪ equation, uses an incorrect formula for  
↪  $(a)$ , and therefore cannot produce the  
↪ correct answer  $(a = 3)$ .
- Consequently, the solution method is logically  
↪ flawed and does not solve the given problem.

### ## Decision

Invalid  
<|return|>

---