LLM-JP Tuning Competition 2026 における数学タスク向け LLM チューニングパイプラインの構築

田中元樹[1] 髙倉真直[1] 齊藤翼[2]
[1]日鉄ソリューションズ中部株式会社 [2]中央コンピューター株式会社
tanaka.motoki.86p@jp.nssol.nipponsteel.com takakura.masanao.9r5@jp.nssol.nipponsteel.com
saito-tsubasa@chuo-computer.co.jp

概要

LLM-JP Tuning Competition 2026 における数学タスクへの取り組みとして大規模言語モデル（LLM）のチューニング手法およびモデル構築手順を整理しその再現可能なパイプラインを報告する．既存の LLM をベースモデルとし，数学問題データセットを用いた追加学習を行った．データ前処理，モデルチューニングまでの一連の実験手順を体系化し，コンペ参加過程で得られたモデル構築および実験設定に関する技術的知見をまとめる．

1 はじめに

近年，大規模言語モデル（Large Language Models: LLM）は自然言語処理のみならず，数学的推論を含む多様なタスクにおいて高い性能を示している．しかし数学問題の解答生成のような推論能力を必要とするタスクでは,モデルの追加学習やデータ整形の方法が性能に大きく影響することが知られている．

本資料では，LLM-JP Tuning Competition 2026 における数学タスクへの参加を通して，大規模言語モデルの学習手順を整理し，再現可能な学習パイプラインを構築することを目的とする．具体的には，公開されている数学問題データセットを整形し，既存の LLM に対して追加学習を行うことで，数学タスクに適したモデルの構築を試みた．

本稿では，データ前処理からモデル学習までの一連の構築手順を整理し，実験環境や学習設定を含めて報告する．これにより，同様の環境において学習手順を再現可能とするための基礎的な情報を提供することを目的とする．

2 データセット

2.1 使用したデータセット

学習データとして，Hugging Face 上で公開されている OpenMathInstruct-2-ja-CoT[1] に加えて，GPT-OSS-20B で作成した学習データを使用した．本データセットは，何れも日本語の数学問題とその解答過程を含むデータで構成されており，Chain-of-Thought（CoT）形式の推論過程を含む特徴を持つ．

LLM に数学的推論能力を学習させるため，データセットに含まれる問題文および解答を学習用データとして利用した．学習に際しては，モデルへの入力形式に合わせてデータの整形を行い，問題文と解答の対応関係を維持した形で学習データを構築した．

2.2 データセットの作成

GPT-OSS-20B を利用した学習データ作成では，中学校学習指導要領(平成 29 年)および，高等学校学習指導要領(平成 30 年)に記載されている指導観点から各単元別に，中学範囲で 44 項目，高校範囲で 670 項目の問題分類を用意した．これらの問題分類それぞれに対して、中学範囲では 30 問題例，高校範囲では 50 問題例をそれぞれ生成させ，学習指導範囲を網羅した 34820 の問題例を用意した．

3 データ前処理

学習データの順序による学習バイアスを軽減するため，データセットのシャッフル処理を行った．具体的には，JSONL 形式で保存された各サンプルを読み込み，JSON としての整合性を確認した後，サンプル単位でランダムシャッフルを実施した．

シャッフル処理には Python の乱数生成器を用い，再現性を確保するため乱数シードを固定した．また，

シャッフル後のデータは元の JSONL 形式を維持したまま保存した．これにより各サンプルが 1 行 1JSON として保持される形式を保ちながら学習時のデータ順序をランダム化した．

本研究では数学データセットとして，公開データセットである OpenMathInstruct-2-ja-CoT を使用し，日本語の問題文・推論過程・最終解答を含むデータを学習用データとして利用した．

4 学習方法

大規模言語モデルに対して数学問題の推論能力を付与することを目的として，教師ありファインチューニング（Supervised Fine-Tuning: SFT）を実施した．

入力データは，問題文と推論過程を含む形式として構築した．具体的には，問題文を ### problem セクションとして与え，続く ### reasoning セクションにおいて推論および最終解答を生成する形式とした．

また，モデルが最終解答部分を明確に識別できるよう，解答部分を <output> トークンで囲む形式を採用した．学習時にはこの <output> トークン以降の部分のみを損失計算の対象とすることで，問題文や推論プロンプト部分に対する不要な学習を抑制した．

モデルの学習には Hugging Face Transformers ライブラリを用い，トークナイザおよびモデルの読み込みを行った．学習時には勾配チェックポイントを有効化し，メモリ使用量を削減しながら学習を実施した．また，追加トークンとして <output> をトークナイザに登録し，モデル語彙に反映させた．モデルの学習においては, 最大入力長 2048 トークンに設定し，問題文および推論過程を含む入力系列をトークナイズして学習データを構築した．学習時のバッチサイズは 2 とし，勾配蓄積（gradient accumulation）を 8 ステップ設定することで，実効バッチサイズを拡張した．学習エポック数は 4 とし，最適化のための学習率は 3×10^{-5}に設定した．学習率スケジューラには線形スケジューリングを採用し，ウォームアップ比率は 0.05 とした．

また，学習時のメモリ効率を向上させるため，勾配チェックポイント（gradient checkpointing）を有効化した．モデルの数値精度には bfloat16 を用い，GPU メモリ使用量の削減と学習効率の向上を図った．

学習の進行状況は一定ステップごとにログ出力を行い，モデルチェックポイントは 300 ステップごとに保存した．保存されるチェックポイント数は最大 3 とし, 古いモデルは自動的に削除される設定とした．

5 実験環境

モデル学習は, 計算資源として ABCI (AI Bridging Cloud Infrastructure) を利用して実施した．ジョブの実行には, 同システムで提供されている GPU ノードを利用し，ジョブクラスとして rt_HG を指定して学習処理を実行した．

ジョブの投入および実行方法については，公式ドキュメントに従い設定を行った．具体的なジョブ実行方法および環境設定の詳細については，ABCI のジョブ実行手順に関する公開資料を参照 [2].

6 結果

学習後のモデルに対して簡易的な動作確認を行ったところ，期待される出力形式が安定して生成されないことが確認された．具体的には，本研究では最終解答を <output> タグで囲む形式を想定していたが，モデルが <output> タグを出力しない場合が多く見られた．

そのため，生成された回答から最終解答部分のみを自動的に抽出することが困難となり，正解率の定量的な評価を実施することができなかった．結果として，本研究の設定では評価スクリプトによる正答判定が行えず，正解率は実質的に 0 に近い結果となった．

この挙動は，学習データの形式や学習方法，あるいは出力フォーマットの設計がモデルの生成挙動に十分に反映されていない可能性を示唆している．

7 まとめ

本資料では LLM-JP Tuning Competition 2026 の数学タスクへの参加を通して，大規模言語モデルの学習パイプラインの構築手順を整理した．公開データセットを用いたデータ整形，学習データの構築，および教師ありファインチューニングの実装を行い，モデル学習までの一連の手順を整理した．

また，実験を通じて，出力フォーマットの設計がモデルの生成挙動に大きく影響することが確認された．本研究では最終解答を <output> タグによって抽出する形式を採用したが，モデルが安定して当該タグを生成しないという問題が観察された．

今後は, プロンプト設計や学習データ形式の改善, および評価方法の再検討を行うことで，数学タスクにおける推論性能の向上を目指す必要がある.

謝辞

本研究の実施にあたり，計算資源として ABCI (AI Bridging Cloud Infrastructure) を提供していただいた関係者の皆様に深く感謝する.

参考文献

[1] Kendamarron.*OpenMathInstruct-2-ja-CoT*. Hugging Face Datasets. https://huggingface.co/datasets/Kendamarron/OpenMathInstruct-2-ja-CoT

[2] ABCI Job Execution Guide https://docs.abci.ai/v3/ja/