

LLMが「賢く，小さく，速く」 なるかColabで実験してみる

川原田 将之

LLMの進化は目覚ましいですが，データ量の巨大さゆえに学習と運用には膨大なリソースが必要という課題があります。本章では，この課題を解決する2つの技術を実装し，その効果を確認します。

実験① QLoRA…モデルの性能を維持しつつ，メモリ使用量を削減する

実験② Key-Valueキャッシング…LLMの応答生成（推論）をキャッシュで高速化する

実験①…QLoRAで，LLMを賢く小さくする

● LLMの実行タスク

大規模言語モデル (LLM) の性能を，特定のタスクに合わせて効率的に引き出す手法 QLoRA を実装します。軽量の LLM Qwen2.5-3B-Instruct を使い，Amazon 日本語レビューの星評価タスクに挑戦します。

● 実行環境の準備

▶ Google Colab での GPU の設定

今回のコード，特にモデルの読み込みやファインチューニングでは，多くの計算が必要となります。そのため，CPU ではなく GPU を利用することが不可欠です。Google Colab では，無料でも高性能な GPU (T4 GPU など) を利用できます。

① Colab ノートブック上部のメニューから「ランタイム」をクリック

② ドロップダウン・メニューから「ランタイムのタイプを変更」を選択

③ 「ハードウェア アクセラレータ」という項目で，「T4 GPU」(または利用可能な他の GPU) を選択

④ 「保存」をクリック

これで，このノートブックは GPU を使ってコードを実行するようになります。セッションがリセットされる場合がありますが，この設定は最初に一度行うだけで OK です。

▶ 必要なライブラリのインストール

必要な Python ライブラリをインストールします。

```
!pip install bitsandbytes trl
```

Colab はデフォルトで必要なライブラリがインストールされています。追加で bitsandbytes と trl をインストールします。Bitsandbytes はモデルを 4 ビットや 8 ビットに量子化するために使用します。今回は QLoRA の中核を担います。trl は主に強化学習を用いたモデルのチューニングに使われますが，今回は SFT (Supervised Fine-Tuning) の便利な機能を利用します。

● ベース・モデルによる Zero-shot 評価

ファインチューニングを一切行わない素の状態のモデルをベース・モデルと呼びます。これがどれくらいの性能を持つのかを確認します。これをベースラインとして，後のチューニング効果を測定します。まずは，コード全体で使用するモジュールやクラスをインポートします (リスト1)。

リスト1 必要なモジュールのインストール

```
import json
import random
import re
from dataclasses import dataclass
from typing import List, Dict, Tuple

import torch
from tqdm.auto import tqdm
from huggingface_hub import hf_hub_download
from datasets import Dataset
from transformers import (
    AutoTokenizer,
    AutoModelForCausalLM,
    BitsAndBytesConfig,
    TrainingArguments,
    Trainer,
)
from peft import LoraConfig, get_peft_model,
    prepare_model_for_kbit_training
```