

ローカル環境で RAGを実行してみる

佐藤 聖

本章では、RAG (Retrieval-Augmented Generation：検索拡張生成) システムを自分のPC (ローカル環境) で構築する実際の手順を解説します。

RAGは、大規模言語モデル (LLM) の知識の限界を克服するための手法です。RAGはまず外部の知識ベースから質問の関連情報をベクトル類似度検索によって取得します (Retrieval)。その後、検索結果をコンテキストとしてLLMに渡し、より正確で最新の情報を基に回答を生成します (Generation)。

このRAGの知識ベースとして機能するのがベクトル・データベースです。本章では、オープンソースで軽量なChromaを採用しました。

Chromaはデータをベクトル (埋め込み) として効率的に格納し、クエリの意味に近い情報を高速に検索できるよう設計されたデータベースです。特にローカル環境での動作が軽快で、小規模な実験やプロトタイプ開発に適しています。また、LangChainやLlamaIndexなどのフレームワークとも容易に連携できます。

次の3段階に分けて説明します。

- ①RAGのローカル環境の整備
- ②ベクトル・データベースの整備
- ③RAGの整備

ステップ①…RAGのローカル環境を整備する

準備

● PC環境

実験で利用したPC環境を表1に示します。RAGを実行する際には、LLM (Large Language Models) の実行にGPU、ベクトル・データベースであるChromaの実行にCPU性能、RAM容量が重要な要素になります。

RAM容量が不足すると処理が非常に遅くなったり、一度に大量データを永続化しようとするメモリ不足で異常終了したりします。Chromaはデフォルトで、全データをメモリ上に読み込んで近傍探索を行うので、十分なRAM容量を確保するか、Chromaのメモリ制限設定を利用するか、分割してデータを永続化するなどの工夫が必要になります。RAM容量が少ない環境で実験するときには、データを減らして永続化するとよいでしょう。データを永続化しておくことで、PCの電源を切ってもデータの消失を防げます。

● GPUドライバのインストール

RAGのタスクを高速に処理するためには、GPU

ボードが必要です。CPUだけで動かないことはないのですが、処理に多くの時間を要します。RAGの中で使用するLLMによっても異なりますが、1回の回答の生成に20～30分かかることもあります。

実験ではGeForce RTX 3060 (NVIDIA) を利用し、CUDAコアで並列演算するのにCUDAドライバv12.9 (執筆時点、最新版) をインストールしました。ドライバは、CUDA Toolkit ページ (<https://developer.nvidia.com/cuda-toolkit>) からダウンロードできます。CUDA Toolkit をインストール後、リスト1のコマンドでCUDAコアが利用できることを確認します。

表1
筆者の
実行環境

項目	内容
OS	Windows 11 (x64) Home Edition
CPU	Core i5 (インテル)
RAM	48G バイト
GPU	GeForce RTX 3060 12G バイト (NVIDIA)
SSD	2T バイト