

OpenWebUI + Ollama による日本語対応生成 AI チャット構築レクチャー

本資料は、Docker を活用してローカル環境に日本語対応の生成 AI チャットを構築するためのレクチャー資料です。以下のステップで段階的に構築を行います。

☑ 使用モデルの選定について（重要）

🔍 モデル選定の背景

本レクチャーは、**ノートパソコン（メモリ 16GB）環境**でも動作可能な構成を前提としています。

日本語性能と軽量性のバランスから、以下のモデルを推奨します。

☑ 推奨モデル構成

① チャット用 LLM

- **モデル名** : `alfredplpl/gemma-2-2b-jpn-it-gguf`
- **特徴** :
 - 日本語指向のGemma 2bモデル（ITチューニング済）
 - VRAM8GB 以下 / RAM16GB で実行可能（4bit 量子化対応）
- **取得方法** :

```
ollama pull hf.co/alfredplpl/gemma-2-2b-jpn-it-gguf
```

② RAG 用 Embedding モデル

- **モデル名** : `cl-nagoya/ruri-base`
- **用途** :
 - ユーザークエリと検索結果のベクトル化
 - OpenWebUI の RAG で指定
- **取得方法** :

```
ollama pull kun432/cl-nagoya-ruri-base
```

■ 第一段階：前提知識と Ollama 体験

☑ Docker の基礎

- **Docker とは** : アプリケーションをコンテナとして仮想化し、どこでも同じ環境で動作させる技術。
-

☑ Ollama の基礎

- **Ollama とは** : ローカルで LLM を実行できる軽量な推論基盤
- **インストール** : <https://ollama.com>
- **使い方** :

モデルの取得 :

```
ollama pull hf.co/alfredplpl/gemma-2-2b-jpn-it-gguf
```

実行 :

```
ollama run hf.co/alfredplpl/gemma-2-2b-jpn-it-gguf
```

第二段階：OpenWebUI をバックエンド Ollama 接続で起動

☒ 手順概要

1. まず `ollama` をローカルで起動しておく。
2. `docker-compose.yml` を作成する。
3. `OLLAMA_BASE_URL=http://host.docker.internal:11434` を設定。
※ Linux の場合は `--add-host=host.docker.internal:host-gateway` の追加が必要。
4. 以下のコマンドでコンテナを作成・起動：

```
docker compose up -d
```

5. ブラウザで `http://localhost:3000` にアクセスし、OpenWebUI の画面を確認。

☒ よく使う Docker コマンド

操作	コマンド
コンテナ起動	<code>docker compose up -d</code>
コンテナ停止	<code>docker compose down</code>
コンテナ確認	<code>docker ps</code>
ログ確認	<code>docker logs -f openwebui</code>

☒ docker-compose.yml (バックエンド分離型)

```
version: "3.8"
services:
  openwebui:
    image: ghcr.io/open-webui/open-webui:main
    container_name: openwebui
    ports:
      - "3000:8080"
    volumes:
      - openwebui-data:/app/backend/data
    environment:
      - OLLAMA_BASE_URL=http://host.docker.internal:11434
    restart: unless-stopped

volumes:
  openwebui-data:
```

🔄 第三段階：RAG 構成による Web 検索付き日本語チャットボットの構築

☑️ モチベーション

- LLM 単体では情報が古かったり曖昧な回答をすることがある
 - 外部情報（Web 検索結果など）を取り入れた RAG（Retrieval-Augmented Generation）で、信頼性・鮮度の高い回答を生成
-

☑️ docker-compose 構成（OpenWebUI + SearxNG）

```
version: "3.8"
services:
  openwebui:
    container_name: openwebui_host
    image: ghcr.io/open-webui/open-webui:main
    environment:
      GLOBAL_LOG_LEVEL: "debug"
      ENABLE_RAG_LOCAL_WEB_FETCH: True
      ENABLE_RAG_WEB_SEARCH: True
      RAG_EMBEDDING_ENGINE: "ollama"
      RAG_EMBEDDING_MODEL: "cl-nagoya/ruri-base:latest"
      RAG_EMBEDDING_BATCH_SIZE: 1
      RAG_OLLAMA_BASE_URL: "http://host.docker.internal:11434"
      CHUNK_SIZE: 500
      CHUNK_OVERLAP: 50
      RAG_WEB_SEARCH_ENGINE: "searxng"
      RAG_WEB_SEARCH_RESULT_COUNT: 3
      RAG_WEB_SEARCH_CONCURRENT_REQUESTS: 10
      SEARXNG_QUERY_URL: "http://searxng:8080/search?lang=ja&q=<query>"
    ports:
      - "3000:8080"
    volumes:
      - open-webui:/app/backend/data
  searxng:
    container_name: searxng_host
    image: searxng/searxng:latest
    ports:
      - "8080:8080"
    volumes:
      - ./searxng:/etc/searxng:rw
    env_file:
      - .env
    restart: unless-stopped

volumes:
  open-webui:
```

☑ OpenWebUI の設定手順 (UI)

1. ブラウザで `http://localhost:3000` にアクセス
 2. 「設定」→「ウェブ検索」タブへ
 3. `searxng` を選択
 4. `http://searxng:8080/search?lang=ja&q=<query>` を設定
 5. 保存
-

☑ SearXNG の設定変更

`searxng/settings.yml` に以下を追加：

```
formats:
  - html
  - json # ←ココ追加
```

☑ 動作確認例

- 質問例：「2024 年のノーベル平和賞受賞者は誰？」
- → RAG 構成により検索 → 要約 → 回答が行われる

☑ まとめ

本レクチャーでは、以下の構成で日本語対応のローカル生成 AI チャットを構築しました。

- **チャットモデル** : alfredplpl/gemma-2-2b-jpn-it-gguf (軽量 + 日本語対応)
 - **RAG Embedding モデル** : cl-nagoya/ruri-base
 - **Web 検索エンジン** : SearxNG
 - **UI** : OpenWebUI
-

☑ 追加 : 今後の改善と応用について

🔗 さらなる精度向上のための改善ポイント

本構成は基本的にローカルで日本語対応チャットが実現できる構成ですが、まだまだ改善を行うことでさらに高精度な運用が可能になります。

① ウェブ検索APIの工夫

- SearxNGの設定を見直し、**日本語ニュースや時事情報を重視するエンジン設定**を推奨します。
- 不要な中国語圏・英語圏エンジンは無効化し、**日本語・時事ニュース特化**に最適化することで回答精度が向上します。

② モデルの選定と比較

- **gemma-2-2b-jpn-it-gguf**は軽量で動作しますが、モデルごとの得意分野を理解し、用途別に切り替えることも重要です。
-

🚀 応用 : OpenWebUIをAPIサーバーとして活用する

OpenWebUIは単なるGUIではなく、**APIとしても利用可能**です。

- **独自のアプリやソフトウェアに、無料でLLMを組み込むことができる**
- HTTPリクエストでOpenWebUIに問い合わせることで、チャットボットや自動化タスクに組み込める
- APIエンドポイントを利用することで、ノーコードツールや外部システムとの連携も容易

☑ 補足

- **host.docker.internal**が使えない場合は**127.0.0.1**とネットワーク設定を確認
- 初回起動時は OpenWebUI のユーザー登録が必要 (管理者アカウント作成)

参考文献・参考資料

- RyoWakabayashi 氏 Qiita 記事
[OpenWebUI + Ollama によるローカル LLM 実行環境構築](#)
- Ollama 公式
<https://ollama.com>
- OpenWebUI GitHub
<https://github.com/open-webui/open-webui>
- SearXNG GitHub
<https://github.com/searxng/searxng>
- モデル配布
 - [gemma-2-2b-jpn-it-gguf](#) (Ollama対応)
 - [cl-nagoya/ruri-base](#) (Hugging Face)