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。

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 gemma:2b
```

  - 実行：

```
ollama run gemma:2b
```

- **CLI実行での挙動確認**：モデルのロード、チャットへの応答、エラーへの対処（例：メモリ不足）

🚀 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` でOpenWebUIコンテナを構築。
3. `OLLAMA_BASE_URL=http://host.docker.internal:11434` を設定。
4. ブラウザで `http://localhost:3000` にアクセスし、OpenWebUI の画面を確認。

☑ 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）により、信頼性・鮮度の高い回答を生成
- 日本語対応の検索と回答に特化した構成を目指す

☑ 構成概要

- **OpenWebUI**：ユーザーとの対話UI
- **Ollama**：ローカルLLM（日本語モデル：`gemma:2b` など）
- **SearxNG**：Web検索エンジン（オープンソースでGoogle/Bingなどをプロキシ検索可能）
- **RAG構成**：
  - ユーザーのクエリをもとに検索用クエリを生成
  - Web検索で取得したドキュメントを埋め込み生成・検索
  - 関連情報と一緒にLLMに回答生成を依頼

☑ 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: "kun432/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>"
      QUERY_GENERATION_PROMPT_TEMPLATE: |-
        ### Task:
        Analyze the chat history to determine the necessity of generating search
queries, in the given language. By default, **prioritize generating 1-3 broad and
relevant search queries** unless it is absolutely certain that no additional
```

```
information is required. The aim is to retrieve comprehensive, updated, and
valuable information even with minimal uncertainty. If no search is unequivocally
needed, return an empty list.

        ### Guidelines:
        - クエリは必ず **日本語** にしてください
        - Respond **EXCLUSIVELY** with a JSON object. Any form of extra
commentary, explanation, or additional text is strictly prohibited.
        - When generating search queries, respond in the format: { "queries": ["クエリ1", "クエリ2"] }, ensuring each query is distinct, concise, and relevant to the
topic.
        - If and only if it is entirely certain that no useful results can be
retrieved by a search, return: { "queries": [] }.
        - Err on the side of suggesting search queries if there is **any chance**
they might provide useful or updated information.
        - Be concise and focused on composing high-quality search queries,
avoiding unnecessary elaboration, commentary, or assumptions.
        - Today's date is: {{CURRENT_DATE}}.
        - Always prioritize providing actionable and broad queries that maximize
informational coverage.

        ### Output:
        Strictly return in JSON format:
        {
          "queries": ["クエリ1", "クエリ2"]
        }

        ### Chat History:
        <chat_history>
        {{MESSAGES:END:6}}
        </chat_history>
    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
    cap_drop:
      - ALL
    cap_add:
      - CHOWN
      - SETGID
      - SETUID
      - DAC_OVERRIDE
    logging:
```

```
        driver: "json-file"
        options:
          max-size: "1m"
          max-file: "1"

  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 SearXNG の設定ファイルが作成される。 検索結果を JSON 形式で返せるように formats の値として json を追加。

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

変更を保存、コンテナを再起動

```
docker compose up
```

この状態で別ターミナルから以下のコマンドを実行すると、 JSON 形式の検索結果が取得できる

```
 curl "http://localhost:8080/search?lang=ja&q=ノーベル&format=json"
```

☑ 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