

無料ツールをつかって
検索支援AIチャットボット
をつくりたかったけど
失敗した話

～キャラクターと一緒に検索する体験を目指して～

例えば、図書館のマスコットとお喋りしながら検索ができれば素敵だと思いませんか？ これはそんな夢の、オリジナルキャラクターをつかった、**AI検索支援チャットボット**を叶えようとして、**パソコンのスペック**と**予算**と**時間**、そして**知識不足**で、挫折するまでの格闘の記録

～成る司書のアトリエ～

コンセプト（制作方針）

1. なるべく無料のツールを使うこと
2. ローカルLLMを使うこと
3. PythonのコードはAI（Gemini 3.5 Flash-Lite）に任せる

使用ソフトとツール

1. **AITuberKit**
2. **LMstudio**
3. **Python**

今回の制作では、2つの理由からローカルLLMを使用することにしました。

1. 費用を抑えるため。
 2. AIに検索履歴を学習させないため
- AIを使ったシステムではAIへの問い合わせが増えると費用がどんどん増えてしまうと問題があります。また、一般的なオンラインの大規模言語モデルでは、会話の内容がAIの学習に利用されてしまいます。有料プランに登録することで、学習を止めることができるものも存在はしますが、どの程度信用していいか疑問が残ります。
- ローカルLLMはパソコン上で動くため、上記の2点の問題を克服することできるはずでした。

仕様ソフトと大まかなフロー

AITuberKit

AITuberKitは「ニケちゃん」氏が**Pixiv**が開発した**ChatVRM**をフォークして制作したAIキャラクターとチャットができるオープンソースのツールキットです。様々なAIサービスに広く対応しています。また、**VRM**、**Live2D**、**PNGTuber**に対応しているため、オリジナルキャラクターを使ったチャットボットの開発にも向いています。

今回は**AITuberKit**をフロントエンドとして、バックエンドとして、**Python**と**Lmstudio**を使う方針です。

Python

今回の開発ではプログラム言語として**Python**を使っています。**Python**のプログラムは**AITuberKit**と**Lmstudio**の間の橋渡しとして機能します。

LMStudio

LMstudioはアメリカの**Element Labs**が開発・提供を行っている、大規模言語をパソコン上にダウンロードして、オフラインでも**AI**を使うことできるソフトウェアです。



AITuberKit の外部連携機能で接続



Pythonの公式 openai ライブラリをつかって接続

- 当初はローカルLLMにMCPを登録して検索から要約までを行う方針でした。
- しかし、実際に試してみるとかなり問題が多く、実用的ではありませんでした。

【直面した課題】

1. ローカルLLM*にMCPの遅延問題

- ローカルLLMで複雑な処理をすると、動作が重く、応答に数分もかかり、実用的でない。
- 「検索語の決定」「検索の実行」「検索の要約」という必要な処理をすべてLLMで行うと時間は、その分伸びる。

2. ハルシネーション（幻覚）の制御の難しさ。

- 結果に独自解釈を混ぜたり、余計な情報を加える事がある。
- MCPから返ってきた結果自体をLLMが要約する。
- 検索語の入力先も明確化しておかないと、検索語にない、出版社などを勝手に加えて失敗することがある。

【解決策】

1. 処理の明確化

- 事前にユーザーの検索目的を分岐させ、目的に合わせたAPIから情報を取得するように変更した。

2. 入力内容の厳格化

- 検索を依頼する際の【書き方】も指定する。

3. ローカルLLMの役割の限定

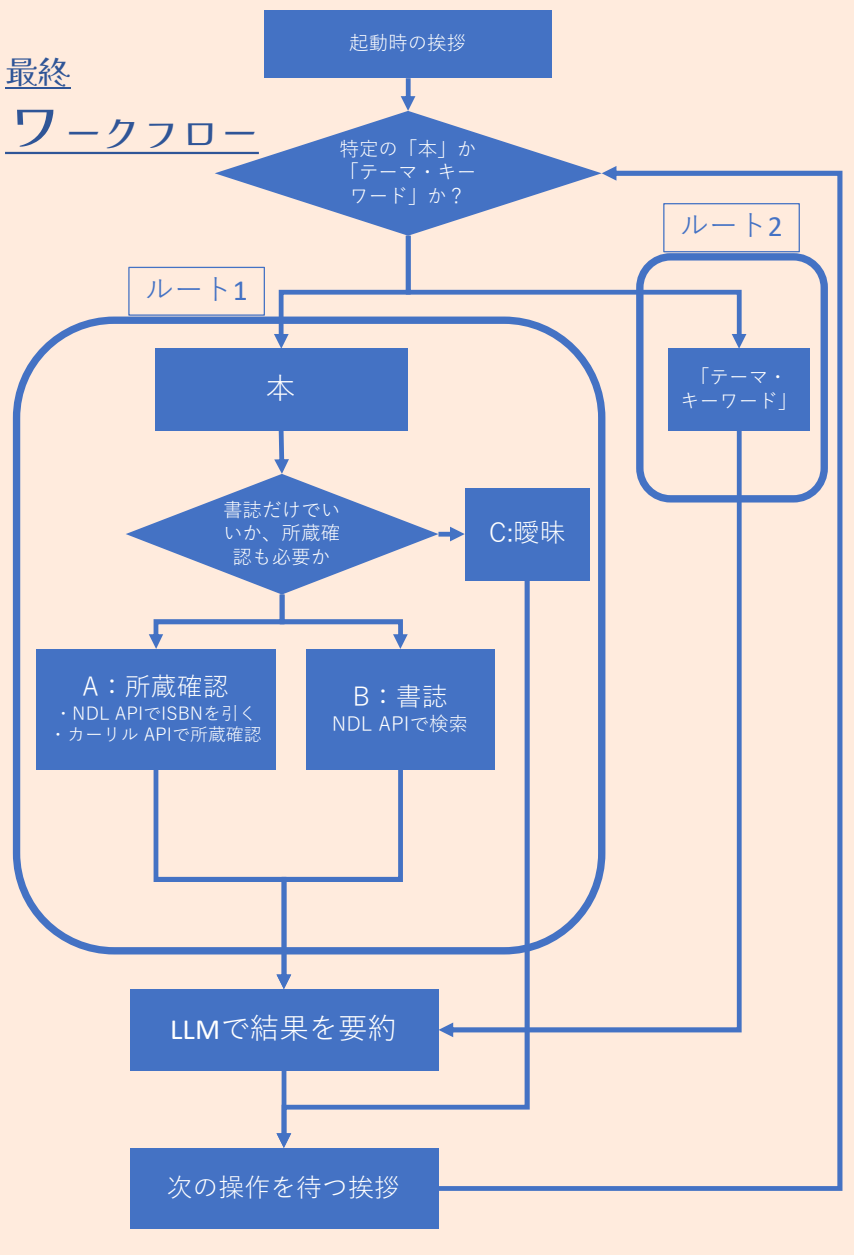
- 検索語の決定にMCPを使うことをやめてPythonで検索語を決定するようにする。
- 検索処理はローカルLLMにやらせず、Pythonから直接、カーリルAPIやNDL APIに問い合わせる形にする。
- ローカルLLMは返ってきたデータの成型にのみ使う。



- 「何でもローカルLLMにやらせる」から「Pythonとの役割分担」へ

最終

ワークフロー



【結果：失敗】

- ワークフローの見直して、動作の軽量化とハルシネーションの抑制はある程度達成できた。
- しかし、実用的なレベルにはできなかった

テスト動画

※応答がいかに遅いかわかりやすくするため
ノーカットです。1質問の応答で7分かかります。



【全体的な課題】

遅延問題

- 参考動画の通り、ある程度は応答が早くなっても実用的（サクサク会話するレベル）な早さとは言い難い。

【ルート1：「蔵書・書籍検索」の課題】

1. 口頭説明の限界。

- 大量の検索結果を口頭で分かりやすく説明させるのは難しい。

2. 目的にあった書籍を正しく選ばせることが難しい。

- ルートAでは「書籍名」+「著者名」で、ルートBでは「書籍名」で検索を行うようにした。
- しかし、例えば「文庫本」か「単行本」かを指定する、「訳者の特定」といった細かいニーズに合わせた検索は、単純な分岐処理や低スペックLLMで対応するのは限界がある。

【ルート2：「キーワード検索」の課題】

1. 単純にユーザーが入力した「キーワード」で検索するだけだと絶対に答えにたどり着けない。

- 今回はとりあえず【ルート1】を正しく動作させることに注力したため、【ルート2】はユーザーの質問をそのまま「キーワード」として各APIに丸投げする状態。
- これだと正しい書籍に到達するのは不可能。

2. レファレンスインタビューの限界

- ユーザーの意図を把握するための高度な対話は、スペックの低いPCのローカルLLMで行うのは不可能。

参考PC SPEC

- OS: Windows 11
- CPU : インテル® Core™ i5-1155G7
- PC4-3200 (DDR4-3200) SDRAM SO-DIMM
ECCなし 260ピン 8GB
- GPU: なし

まとめ

- 結論としては、メモリ8GBのGPU非搭載という限られたハードウェア環境の中で、実用的なローカルLLMベースの検索アシスタントの設計は非常に難しいということが分かりました。

- これらの壁を突破するには

1. PCのスペックの大幅な向上
2. オンラインのLLMの使用

この2つが必要ではないかと思います。ただ、開発の方向自体はあつてだと思いますし、『AItuberKit』を使えばオリジナルキャラクターの検索アシスタントは作れると思います。もし自分の環境なら実現できそうという方がいらっしゃいましたら、私の屍を超えて「夢の図書館検索アシスタント」の完成に挑戦してみてください。

より詳細な説明はnoteに記載してあります。
ご興味のある方は右の二次元コードからアクセスしてみてください。

