AI を使った文書検索／仕様管理（PrivateGPT）

目次

AI を使った文書検索／仕様管理（PrivateGPT）

はじめに

システム更改のバックログが山積みになり、挙句にトラブルやミスが続いて利用部門がカンカン…打開策としてアジャイル開発を取り入れようなんて現場がでてきます。アジャイル開発は案件を同一担当が UX～実装～マニュアル更改の一気通貫で実施することでスピードも品質も一時的に上がりますが、裏腹に、担当者が入替になるとユーザマニュアルの差分と非定型の設計メモだけが残り、全体像を把握できない状態が生まれます。では、ウォータフォールのように大量のドキュメントを作っていれば大丈夫かというと、それはそれで仕様書・設計書、改変履歴を精査する労力が必要です。

Fess（全文検索サーバ）のようなドキュメントを検索するツールは AI が取り沙汰される前からありますが、昨今、ローカルで動作する AI 環境を構築して手元のドキュメントを入力情報に使うことができるソフトウェア／フレームワークがオープンソースで公開されています。これを上手く使うと、柔軟で高度なドキュメント整理の仕組みを組織内に閉じた形で作ることができます。

1. 技術背景 … Llama 2（オープンソース）、RAG（情報矯正）、量子化（情報圧縮）

ツール／LLM が多数発表されていますが、構成を決めるには以下の知識が必要です。

（1）Llama2／LlamaIndex/Llama.cpp

2023 年 7 月に Meta 社がオープンソースの大規模言語モデル（LLM）Llama2 と、ドキュメントと LLM を関係づけるためのフレームワーク LlamaIndex を公開しました。また、LLM を操作するための API を提供する Llama.cpp のプラットフォームと開発言語が広がり、日本語を含む多くの Llama2 派生 LLM と LlamaIndex を組込んだソフトウェアが作られ、個人で利用できるようになりました。

（2）RAG

生成 AI は仕組み上、嘘の情報を生成（合成）してしまうという問題がありました。

この対策として考えられたのが RAG（検索拡張生成：**R**etrieval-**A**ugmented **G**eneration）で、回答を生成する際の参照資料（コンテキスト）を与えて幻覚が発生するのを防止します。ドキュメント管理で使う際の肝はドキュメントをコンテキストとして LLM に処理させることですが、これは、以下の手順で実現します。

① 利用者の問合せ（プロンプト）にあったコンテキストをドキュメント群から抽出する
② プロンプトとコンテキストを合成して LLM に問合せを行う
③ LLM の回答を利用者に提示する

ドキュメントからコンテキストを生成するために予め以下の処理を行っておく必要があり、土台になるソフトウェアが LlamaIndex です。

・**tokenizer**：トークナイザー（tokenizer）で文をトークンに分解します
・**embedding**：重要性と関連性の重みづけによりトークンを数値ベクトル化（embedding）します
・**vector store**：「意味」の近さで検索できるよう、ベクトルはインデックス化して保存します

（3）量子化

情報の整理（LLM の作成）と問合せ応答（Query）のための膨大な計算に GPU が必要でしたが、Query の方はベクトル量子化を行うことにより GPU がない PC でも動作する（Llama.cpp で操作）ようになりました。量子化はベクトルの特徴（重み）を残しながらビット数を減らすことで計算量を減らしています。量子化した LLM は gguf 形式と呼ばれ、q2_K(2bit)、q4_K_M(4bit)等のビット型をファイル名に付けて llama-2-7b-chat.ggmlv3.**q4_K_M**.gguf のようになります。

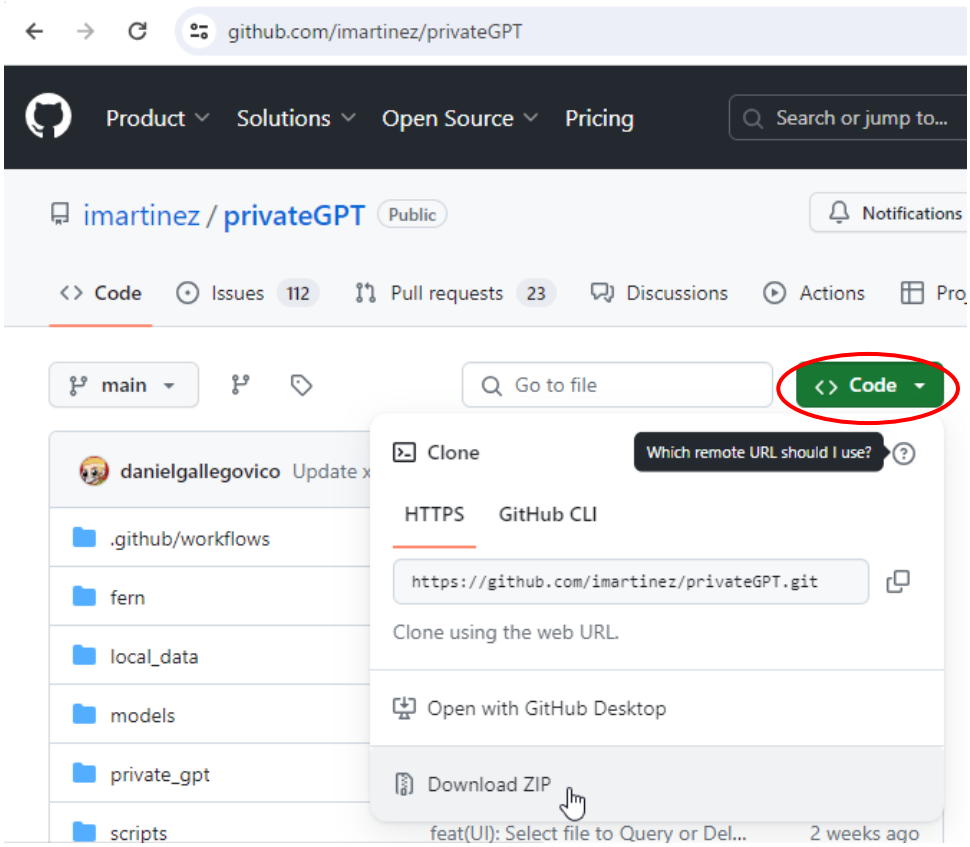
AI を使った文書検索／仕様管理（PrivateGPT）

2. PrivateGPT

ローカル AI を構成するためのソフトウェアは多数ありますが、ここでは PrivateGPT を使った事例を紹介します。PrivateGPT はドキュメントからベクトルインデックスを作る機能や、コンテキストとプロンプトを合成して LLM とのやり取りを行う一連の機能を Web アプリとして一体化した、Apache-2.0 license のオープンソースです。

公開サイト：https://github.com/imartinez/PrivateGPT

公式サイトのインストール方法は git clone でリポジトリをローカルにコピーするようになっていますが、Git を使わなくても Code⇒Download zip でリポジトリ相当の資材を取得できます。



zip ファイルのダウンロードが終わったら解凍します。以下、解凍したディレクトリを”PrivateGPT”とした前提で説明します。

3. Windows 環境（GPU 無し）構築用のツール

PrivateGPT は Python やその他のツールを準備して環境を作る必要があります。以下は Windows11 の PC に LLM も含めて完全にローカルに環境を後置する手順です。プロジェクトのサイトには GPU を使う場合の手順も記載されていますが、ここでは CPU だけで動作させる手順に絞って記載します。

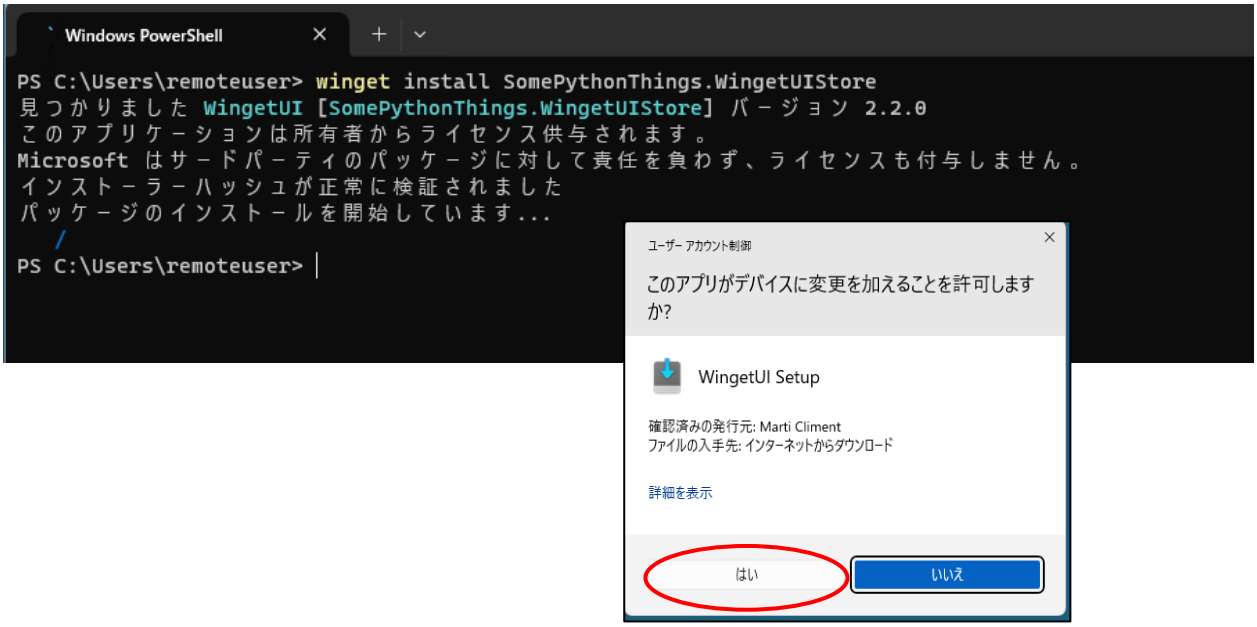
また、環境の構築に複数のソフトウェアの特定のバージョンが求められるので、GUI ツールを使ってインストールを行う例を示します。

3.1. WingetUI

WingetUI[1]は winget を含む複数のパッケージマネージャからインストールを行えるオープンソース（LGPL-2.1 license）の GUI ツールです。winget は Windows OS 用のパッケージャ[2]で、Windows10 以降は Windows Update を行っていればインストールされているはずです。

以下のコマンドで WinggetUI をインストールします。

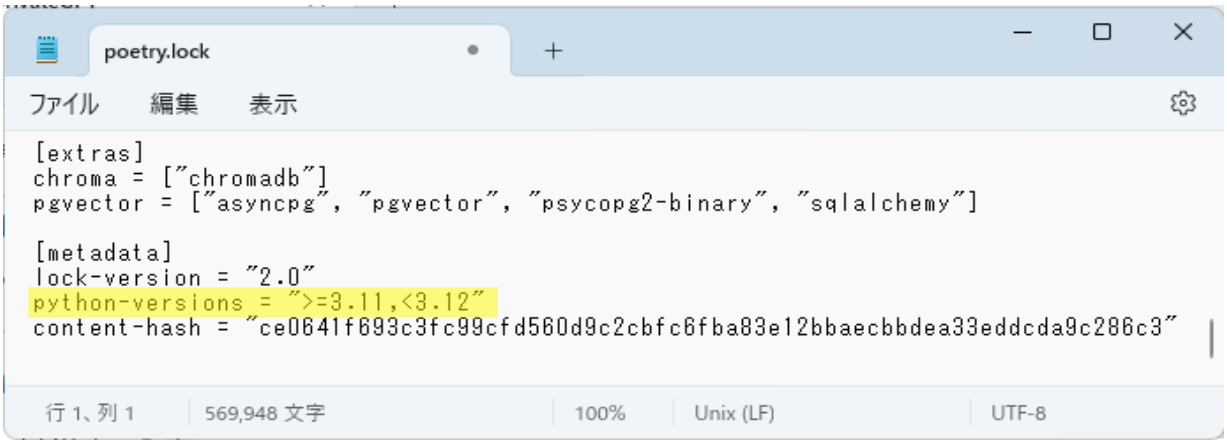
```
winget install SomePythonThings.WingetUIStore
```



3.2. Python のインストール

PrivateGPT に格納されたファイル poetry.lock に書いてあるバージョンが必要になります。

以下のように書かれていた場合、Python のバージョン 3.11 以上、3.12 未満が必要なので 3.11.xx をインストールします。

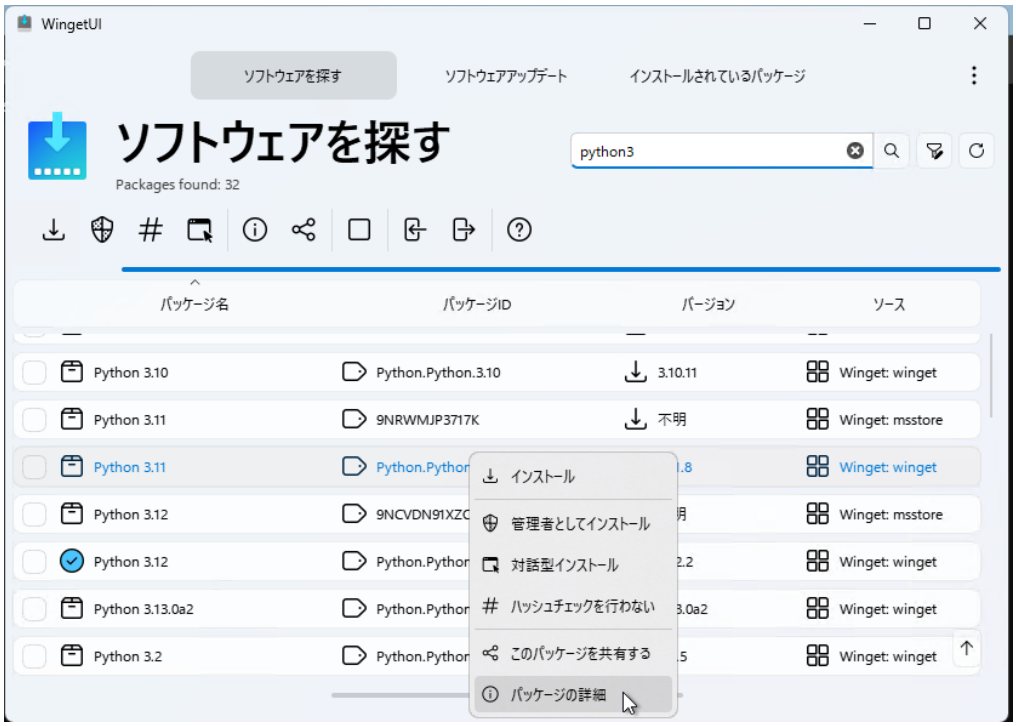


[1] WingetUI(近々"**UnigetUI**"に名称変更するそうです) https://github.com/marticliment/WingetUI

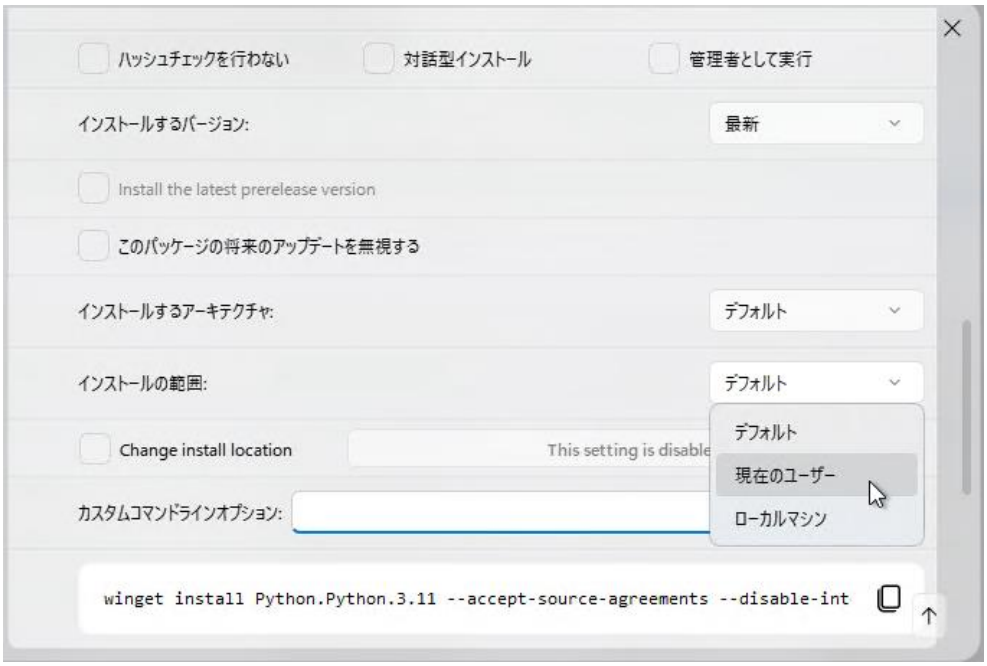
[2] winget https://learn.microsoft.com/ja-jp/windows/package-manager/winget/

AI を使った文書検索／仕様管理（PrivateGPT）

WingetUI を使って Python3 のパッケージを探した例

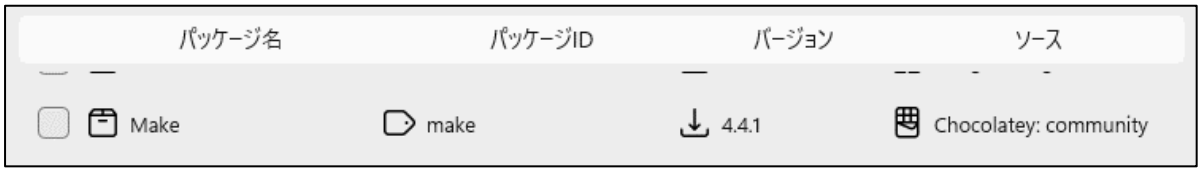


該当のパッケージを右クリックして“パッケージの詳細”を選ぶと、パッケージの内容を確認したりインストールのオプションを選択することができます。



※ Windows の場合、Python はデフォルトでランチャーの py を使い起動するように設定されます
(複数バージョンをインストールしている場合は、py -3.11 とオプションでバージョンを指定します)

3.3. make のインストール



3.4. Poetry

poetry は Python のツールですが、仮想環境を作ってパッケージのバージョン管理を行います。

（1）インストール

ドキュメント[3]に従い、pipx を使って poetry を隔離環境にインストールします。

① pipx のインストール

```
py -m pip install --user pipx
```

ワーニングがでますが、最後に Successfully installed …が表示されたらＯＫ

```
Installing collected packages: click, argcomplete, userpath, pipx
  WARNING: The script userpath.exe is installed in 'C:\Users\remoteuser\AppData\Roaming\Python\Python311\Scripts' which
is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The script pipx.exe is installed in 'C:\Users\remoteuser\AppData\Roaming\Python\Python311\Scripts' which is n
ot on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed argcomplete-3.2.2 click-8.1.7 pipx-1.4.3 userpath-1.9.2
```

② poetry のインストール

```
py -m pipx install poetry
```

```
PS C:\Users\remoteuser> py -m pipx install poetry
  installed package poetry 1.8.2, installed using Python 3.11.8
  These apps are now globally available
    - poetry.exe
⚠  Note: 'C:\Users\remoteuser\.local\bin' is not on your PATH environment variable. These apps will not be globally
    accessible until your PATH is updated. Run `pipx ensurepath` to automatically add it, or manually modify your PATH
    in your shell's config file (i.e. ~/.bashrc).
done! ✨ 🌟 ✨
```

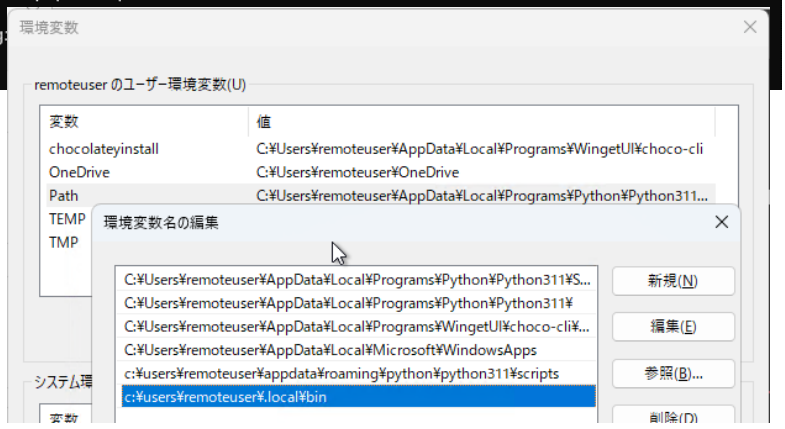
③ 環境変数 path の設定

poetry のインストール時〔⚠ Note:〕で表示されるように poetry のインストール先~¥.local¥bin が環境変数の PATH に含まれないので、pipx ensurepath を実行するかシステムのプロパティから Path に手動で追加します。

```
py -m pipx ensurepath
```

```
PS C:\Users\remoteuser> $ENV:Path
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\
OpenSSH\;C:\Users\remoteuser\AppData\Local\Programs\Python\Python311\Scripts\;C:\Users\remoteuser\AppData\Local\Programs
\Python\Python311\;C:\Users\remoteuser\AppData\Local\Programs\WingetUI\choco-cli\bin;C:\Users\remoteuser\AppData\Local\M
icrosoft\WindowsApps;
PS C:\Users\remoteuser> py -m pipx ensurepath
Success! Added C:\Users\remoteuser\AppData\Roaming\Python\Python311\Scripts to the PATH environment variable.
Success! Added C:\Users\remoteuser\.local\bin to the PATH environment variable.

Consider adding shell completions for pipx. Run 'pipx completions' for instructions.

You will need to open a new terminal or re-log

Otherwise pipx is ready to go! ✨ 🌟 ✨
```

⇒コマンド実行後の Path の状態



[3] Python-poetry の導入 https://python-poetry.org/docs/

AI を使った文書検索／仕様管理（PrivateGPT）

（２）使い方

poetry は poetry.lock ファイル（と pyproject.toml）を参照して指定バージョンの依存ライブラリを集めた仮想環境をつくります。初回の環境構築には必要がありませんが、環境を更新する場合等には以下のコマンドで仮想環境の内容の確認や操作が必要になる場合があります。

① バージョンの確認と使い方の確認　poetry List

```
PS C:\Users\remoteuser\Tools\privateGPT> poetry list
Poetry (version 1.8.2)

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display help for the given command. When no command is given display help for the list comm
and.
  -q, --quiet               Do not output any message.
  -V, --version             Display this application version.
      --ansi                Force ANSI output.
      --no-ansi             Disable ANSI output.
  -n, --no-interaction      Do not ask any interactive question.
      --no-plugins          Disables plugins.
      --no-cache            Disables Poetry source caches.
  -C, --directory=DIRECTORY The working directory for the Poetry command (defaults to the current working directory).
  -v|vv|vvv, --verbose      Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 fo
r debug.

Available commands:
  about              Shows information about Poetry.
  add                Adds a new dependency to pyproject.toml.
  build              Builds a package, as a tarball and a wheel by default.
  check              Validates the content of the pyproject.toml file and its consistency with the poetry.lock file.
  config             Manages configuration settings.
  export             Exports the lock file to alternative formats.
  help               Displays help for a command.
  init               Creates a basic pyproject.toml file in the current directory.
  install            Installs the project dependencies.
```

② 仮想環境の確認

poetry install コマンドで作られた環境は、poetry env info コマンドで確認できます。

```
PS C:\Users\remoteuser\Tools\privateGPT> poetry env info

Virtualenv
Python:         3.11.8
Implementation: CPython
Path:           C:\Users\remoteuser\AppData\Local\pypoetry\Cache\virtualenvs\private-gpt-m2J3WB53-py3.11
Executable:     C:\Users\remoteuser\AppData\Local\pypoetry\Cache\virtualenvs\private-gpt-m2J3WB53-py3.11\Scripts\python.
exe
Valid:          True

Base
Platform:   win32
OS:         nt
Python:     3.11.8
Path:       C:\Users\remoteuser\AppData\Local\Programs\Python\Python311
Executable: C:\Users\remoteuser\AppData\Local\Programs\Python\Python311\python.exe
```

③ インストール済のパッケージの確認 poetry show

```
PS C:\Users\remoteuser\Tools\privateGPT> poetry show
aiofiles                                 23.2.1       File support for asyncio.
aiohttp                                  3.9.1        Async http client/server framework (asyncio)
aiosignal                                1.3.1        aiosignal: a list of registered asynchronous callbacks
altair                                   5.2.0        Vega-Altair: A declarative statistical visualization library...
annotated-types                          0.6.0        Reusable constraint types to use with typing.Annotated
```

④ poetry.lock ファイルに記載がないパッケージの削除　poetry install --sync

```
PS C:\Users\remoteuser\Tools\privateGPT> poetry install --sync
Installing dependencies from lock file

Package operations: 0 installs, 0 updates, 64 removals

  - Removing aiofiles (23.2.1)
  - Removing altair (5.2.0)
  - Removing contourpy (1.2.0)
```

⑤ 操作中（poetry.lock）の仮想環境　poetry env list

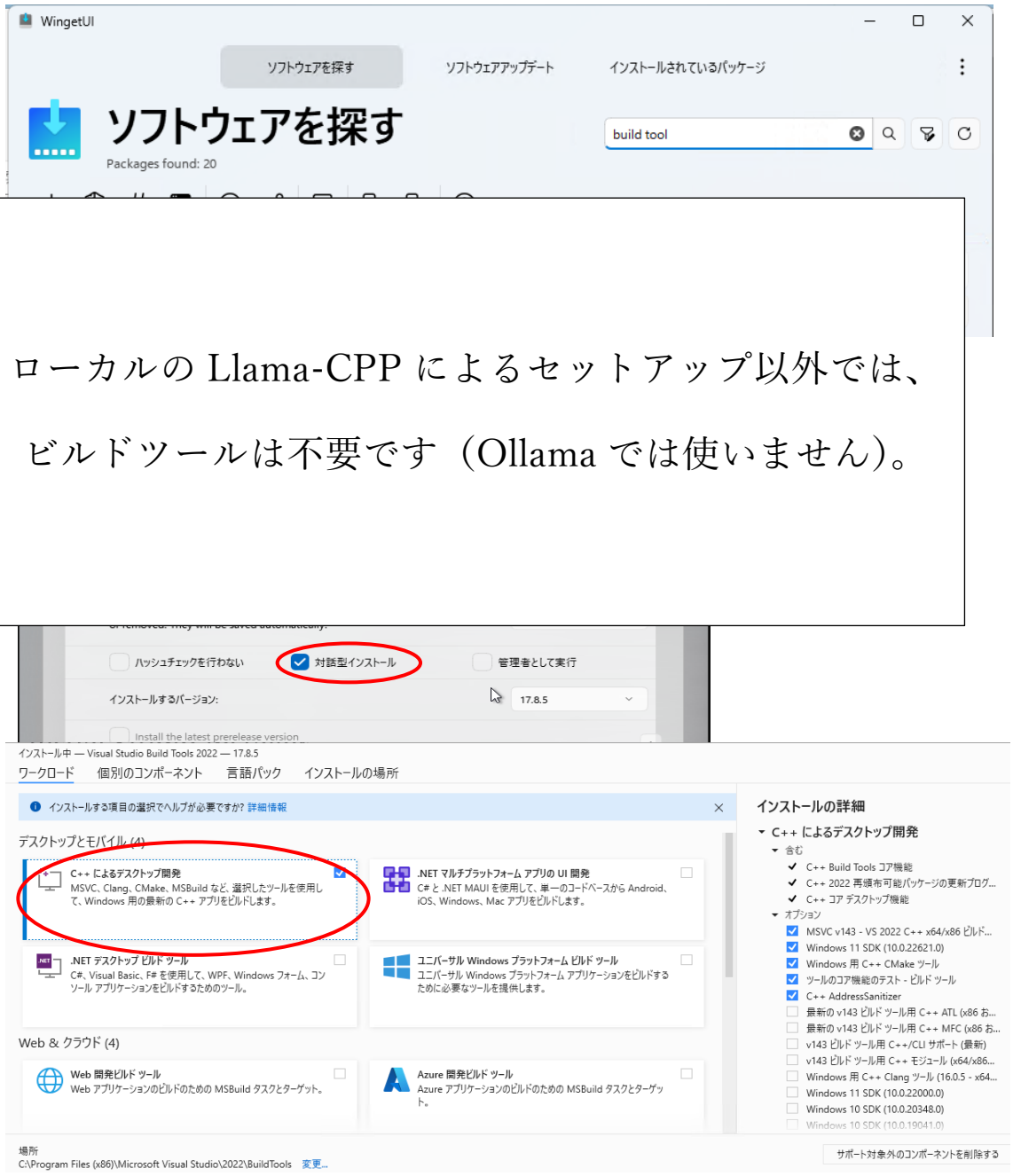
```
PS C:\Users\remoteuser\Tools\privateGPT> poetry env list
private-gpt-m2J3WB53-py3.11 (Activated)
```

⑥ 仮想環境の削除　poetry env remove *仮想環境名*

```
PS C:\Users\remoteuser\Tools\privateGPT> poetry env remove private-gpt-m2J3WB53-py3.11
Deleted virtualenv: C:\Users\remoteuser\AppData\Local\pypoetry\Cache\virtualenvs\privat
e-gpt-m2J3WB53-py3.11
```

3.5.　C++コンパイラと CMake のインストール

Visual Studio BuildTools の一番新しいバージョンを選択します。

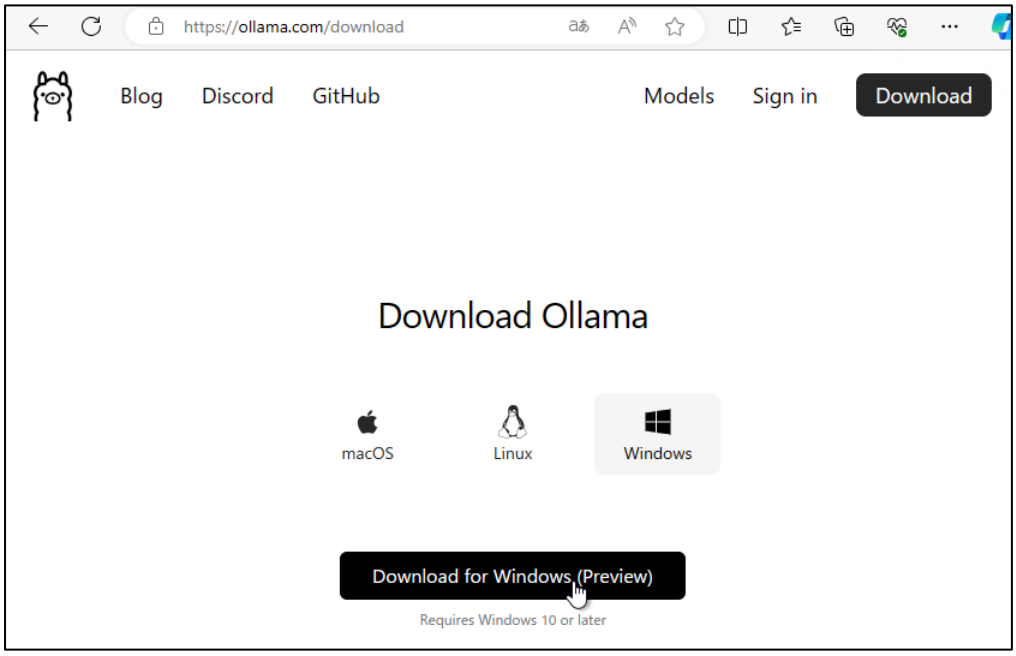


3.6. Ollama

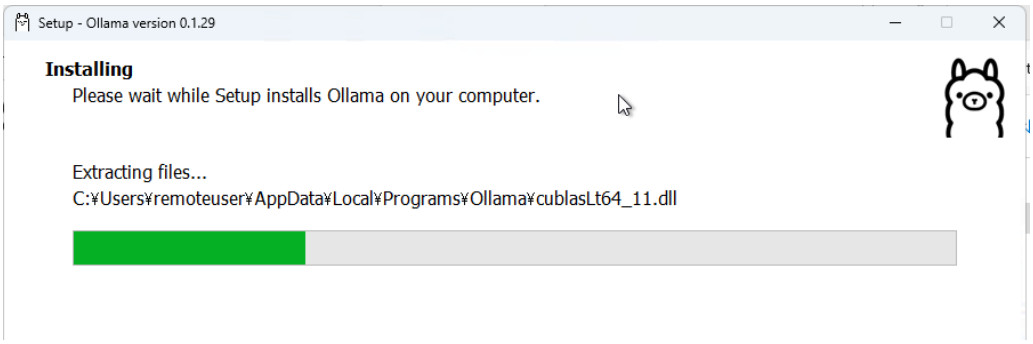
PrivateGPT の構成形態は選べますが PrivateGPT が ver.0.4.0(3/17 版)で推奨しているのは LLM を自前で制御（Llama-CPP を使用）するのではなく Ollama（MIT License のオープンソース）という LLM を制御するサービスに接続する形態に変わっています[4]。

（1）インストール

Ollama は右記サイトからダウンロードできます：https://ollama.com/download



ダウンロードしたインストーラを実行します（当資料で使ったのは Ollama ver.0.1.29）。



＜インストール完了後＞

① Ollama が Path に追加されているのを確認

```
$env:path
```

～（略）～;*ユーザのホームディレクトリ*¥AppData¥Local¥Programs¥**Ollama**

② Ollama はスタートアップメニュー（下記パス）に登録され、常時起動するように設定されます

ユーザのホームディレクトリ¥AppData¥Roaming¥Microsoft¥Windows¥Start Menu¥Programs

[4] privatePGT 2024/03/20 版で poety.lock に書かれている LlamaCPP(ver.0.2.53)が Windows11 日本語版はコンパイルエラーになります（ver.0.2.57 はコンパイルできます。恐らく文字コードの問題）

インストール直後と、Windows 再起動後に以下のプロセスが起動するようになります。

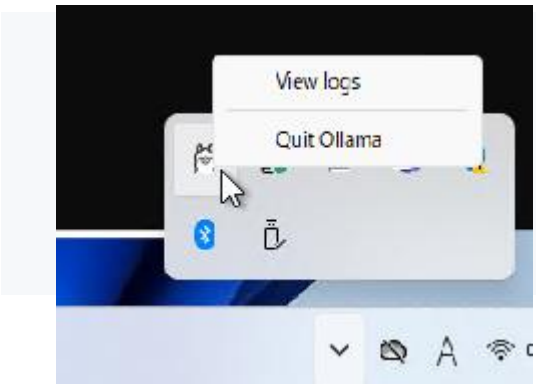
```
PS C:\Users\remoteuser> ps -name '*ollama*'

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    127      11    18988     128608       0.13   19772  17 ollama
    411      17    17384      20312       0.30   19812  17 ollama app
```

（2）起動と停止

インストール後は手作業でスタートアップメニューから削除するまで自動で起動します。LLM の入替等で停止／起動する場合は、以下のようにします。

① プロセスの確認と停止

PowerShell のコマンド `ps -name '*ollama*'` を実行するか、タスクバーからも確認できます。



停止はタスクバーから Quit Ollama を選ぶか、以下のコマンドを実行します。

```
(ps -name '*ollama*').kill()
```

② 起動

```
ollama serve
```

（3）LLM のダウンロード

LLM は、https://ollama.com/library から選択できます。

PrivateGPT がデフォルトにしている mistral を選択する場合は以下のコマンドを実行します。

```
ollama pull mistral
ollama pull nomic-embed-text
```

```
PS C:\Users\remoteuser> ollama pull mistral
pulling manifest
pulling e8a35b5937a5... 100%  4.1 GB
pulling 43070e2d4e53... 100%   11 KB
pulling e6836092461f... 100%   42 B
pulling ed11eda7790d... 100%   30 B
pulling f9b1e3196ecf... 100%   483 B
verifying sha256 digest
writing manifest
removing any unused layers
success
PS C:\Users\remoteuser> ollama pull nomic-embed-text
pulling manifest
pulling 970aa74c0a90... 100%  274 MB
pulling c71d239df917... 100%   11 KB
pulling ce4a164fc046... 100%   17 B
pulling 31df23ea7daa... 100%   420 B
verifying sha256 digest
writing manifest
removing any unused layers
success
```

ダウンロード済のモデルは、ollama list コマンドで確認できます。

```
PS C:\Users\remoteuser> ollama list
NAME                       ID              SIZE    MODIFIED
mistral:latest             61e88e884507    4.1 GB  14 minutes ago
nomic-embed-text:latest    0a109f422b47    274 MB  8 minutes ago
```

＜LLM の保存先＞

ユーザのホームディレクトリ¥.ollama

3.7. PrivateGPT の依存ライブラリインストール

インストールするライブラリは構成により変わり、extras オプションで各要素を指定します。

ui:ブラウザを使う

llms-：問合せ先の LLM は、ollama 経由(ollama)、PrivateGPT 自身で行う(llama-cpp) 他

embeddings-：ベクトル数値化を、ollama 経由、PrivateGPT 自身で行う(huggingface) 他

vector-stores-：ベクトルの保存先（qdrant を選ぶとデフォルトで local_data 配下に保存されます）

以下は全てローカル（かつ LLM は Ollama 経由）で動作する環境を作る手順です。

poetry は全て PrivateGPT のディレクトリ（poetry.lock の場所）で実行する必要があります。

cd *PrivateGPT ディレクトリ*

poetry install --extras "ui llms-ollama embeddings-ollama vector-stores-qdrant"

```
PS C:\Users\remoteuser> cd C:\Users\remoteuser\Tools\privateGPT
PS C:\Users\remoteuser\Tools\privateGPT> poetry install --extras "ui llms-ollama embeddings-ollama vector-stores-qdrant"
Installing dependencies from lock file

Package operations: 4 installs, 0 updates, 4 removals


  - Removing diskcache (5.6.3)
  - Removing mpmath (1.3.0)
  - Removing sympy (1.12)
  - Removing torch (2.1.2)
  - Installing gradio (4.19.2)
  - Installing llama-index-embeddings-ollama (0.1.2)
  - Installing llama-index-llms-ollama (0.1.2)
  - Installing llama-index-vector-stores-qdrant (0.1.3)

Installing the current project: private-gpt (0.4.0)
```

extras オプションで llms-llama-cpp を指定して環境構築を行いビルドに失敗した場合、PrivateGPT の起動時に以下のようなエラーが出る場合があります。

```
ImportError: cannot import name 'Settings' from partially initialized module 'llama_index.core.settings' (most likely due to a circu
lar import) (C:\Users\remoteuser\AppData\Local\pypoetry\Cache\virtualenvs\private-gpt-m2J3WB53-py3.11\Lib\site-packages\llama_index\
core\settings.py)
make: *** [Makefile:36: run] Error 1
```

この場合は poetry env remove で中途半端になった仮想環境を削除し、再度 poetry install を行う必要があります。poetry install は複数回実行しても問題なく、依存ライブラリの変更／更新も行います。

```
PS C:\Users\remoteuser\Tools\privateGPT> poetry install --extras "ui llms-llama-cpp embeddings-huggingface vector-stores-qdrant"
Installing dependencies from lock file

Package operations: 145 installs, 1 update, 0 removals

  - Installing certifi (2023.11.17)
  - Installing h11 (0.14.0)
```

AI を使った文書検索／仕様管理（PrivateGPT）

4. PrivateGPT サービスの起動

cd *PrivateGPT ディレクトリ*
$env:PGPT_PROFILES="ollama"
make run

初回の起動でアクセス許可の確認が出るので[許可]を押下します

```
PS C:\Users\remoteuser\Tools\privateGPT> $env:PGPT_PROFILES="ollama"
PS C:\Users\remoteuser\Tools\privateGPT> make run
poetry run python -m private_gpt
14:13:31.578 [INFO    ] private_gpt.settings.settings_loader - Starting application with profiles=['default', 'ollama']
None of PyTorch, TensorFlow >= 2.0, or Flax have been found. Models won't be available and only tokenizers, configuration and file/d
ata utilities can be used.
tokenizer_config.json: 100%|████████████████████████████████████████████████████████████████| 1.46k/1.46k [00:00<?, ?B/s]
14:14:01.016 [WARNING ]              py.warnings - C:\Users\remoteuser\AppData\Local\pypoetry\Cache\virtualenvs\private-gpt-m2J3WB5
3-py3.11\Lib\site-packages\huggingface_hub\file_download.py:147: UserWarning: `huggingface_hub` cache-system uses symlinks by defaul
t to efficiently store duplicated files but your machine does not support them in C:\Users\remoteuser\Tools\privateGPT\models\cache.
 Caching files will still work but in a degraded version that might require more space on your disk. This warning can be disabled by
 setting the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more details, see https://huggingface.co/docs/huggingface_h
ub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to see act
ivate developer mode, see this article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development
  warnings.warn(message)

tokenizer.model: 100%|████████████████████████████████████████████████████████████████| 493k/493k [00:00<00:00, 5.20MB/s]
tokenizer.json: 100%|██████████████████████████████████████████████████████████████| 1.80M/1.80M [00:00<00:00, 2.35MB/s]
special_tokens_map.json: 100%|████████████████████████████████████████████████████████| 72.0/72.0 [00:00<?, ?B/s]
14:14:03.921 [INFO    ] private_gpt.components.llm.llm_component - Initializing the LLM in mode=ollama
14:14:05.999 [INFO    ] private_gpt.components.embedding.embedding_component - Initializing the embedding model in mode=ollama
14:14:06.015 [INFO    ] llama_index.core.indices.loading - Loading all indices.
14:14:06.015 [INFO    ] private_gpt.components.ingest.ingest_component - Creating a new vector store index
Parsing nodes: 0it [00:00, ?it/s]
Generating embeddings: 0it [00:00, ?it/s]
14:14:06.218 [INFO    ]         private_gpt.ui.ui - Mounting the gradio UI, at path=/
14:14:06.531 [INFO    ]             uvicorn.error - Started server process [12084]
14:14:06.531 [INFO    ]             uvicorn.error - Waiting for application startup.
14:14:06.531 [INFO    ]             uvicorn.error - Application startup complete.
14:14:06.531 [INFO    ]             uvicorn.error - Uvicorn running on http://0.0.0.0:8001 (Press CTRL+C to quit)
```

“Running on http://0.0.0.0:8001 (Press CTRL+C to quit)” が表示されたら起動完了

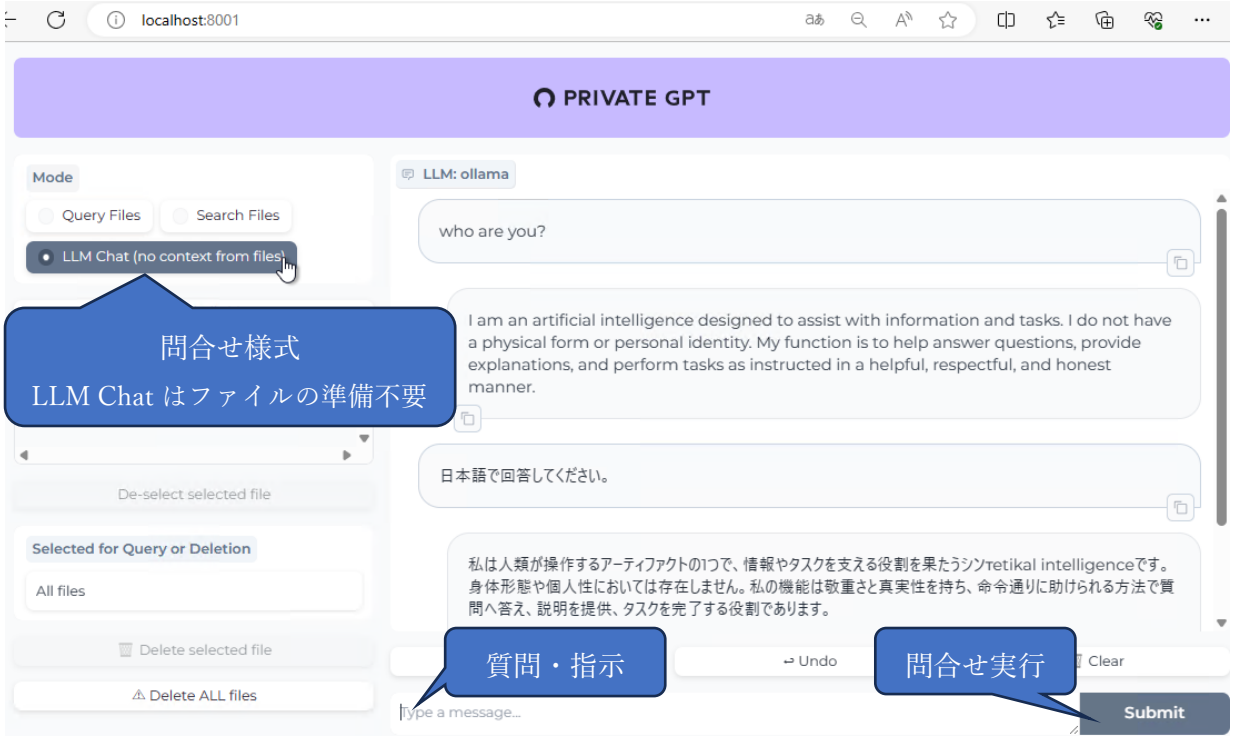
5. PrivateGPT との会話

ブラウザを起動してサーバの 8001 ポート（デフォルトから変えてない場合）に接続します。

http://localhost:8001/　　My PrivateGPT

5.1. チャット

LLM Chat は前準備なしで使えます。
初期状態では英文で回答が表示されますが、日本語の質問を理解して翻訳もしてくれます。
（但し、どの程度対応できるかは使う LLM 次第）



初期状態で設定されていた LLM は以下のサイトで確認でき、初歩的な日本語を理解できるようです。
https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.2-GGUF

AI を使った文書検索／仕様管理（PrivateGPT）

5.2. ローカルファイルのアップロード（ベクトルインデックスの作成）

自分のファイルを問合せ対象として使うためには予め Upload Files(s)でサーバにアップロードと解析処理（自動的に処理が走ります）を行っておき、Mode で[Query Files]か[Search Files]を選びます。

PDF やテキストファイル、オフィス文書が選択できますが Excel（xlsx、xls）は処理できません。また、初期状態で Word ファイル（.docx）を処理しようとすると以下のエラーがでます。

```
  File "C:\Users\remoteuser\AppData\Local\pypoetry\Cache\virtualenvs\private-gpt-m2J3WB53-py3.11\Lib\site-packages\llama
_index\readers\file\docs_reader.py", line 60, in load_data
    raise ImportError(
ImportError: docx2txt is required to read Microsoft Word files: `pip install docx2txt`
```

【docx を使う準備】以下を実行すると Word 文書が使えるようになります。

```
py -m pip install docx2txt
```

また、PrivateGPT から追加ライブラリ（docx2txt）を使えるようにするには poetry.lock の書き換えが必要で、以下のコマンドを実行します。

```
cd PrivateGPT ディレクトリ
poetry add docx2txt
```

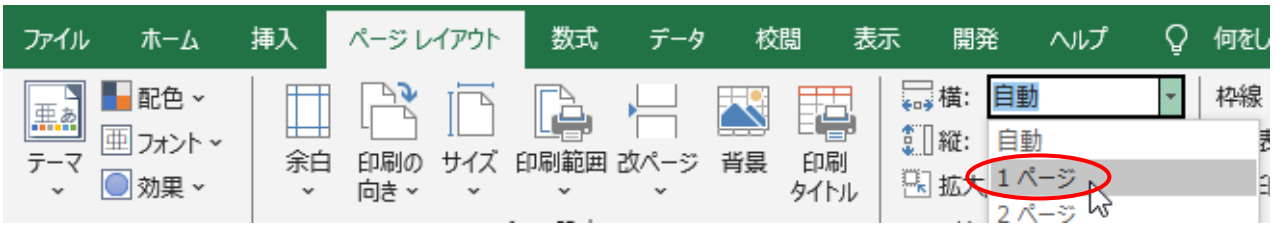
【xlsx, xls：現状、使えません】

2024/3/4 の時点で Excel ファイル（.xlsx）は以下のエラーが発生します。一般のテキストファイルとして扱ってしまう誤りがあるようです[5]が未対策です。

```
  File "C:\Users\remoteuser\Tools\privateGPT\private_gpt\components\ingest\ingest_helper.py", line 48, in _load_file_to_
documents
    return string_reader.load_data([file_data.read_text()])
                                    ^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\remoteuser\AppData\Local\Programs\Python\Python311\Lib\pathlib.py", line 1059, in read_text
    return f.read()
           ^^^^^^^^
UnicodeDecodeError: 'cp932' codec can't decode byte 0x9c in position 22: illegal multibyte sequence
```

※ xlsx は PDF か HTML に変換してください。また、その際、横方向が 1 ページになるようにして表（情報）がページを跨がないようにしてください

〔注〕現状では、これを行った場合でも見出行＋明細行という「表」形式は意味を解釈できません

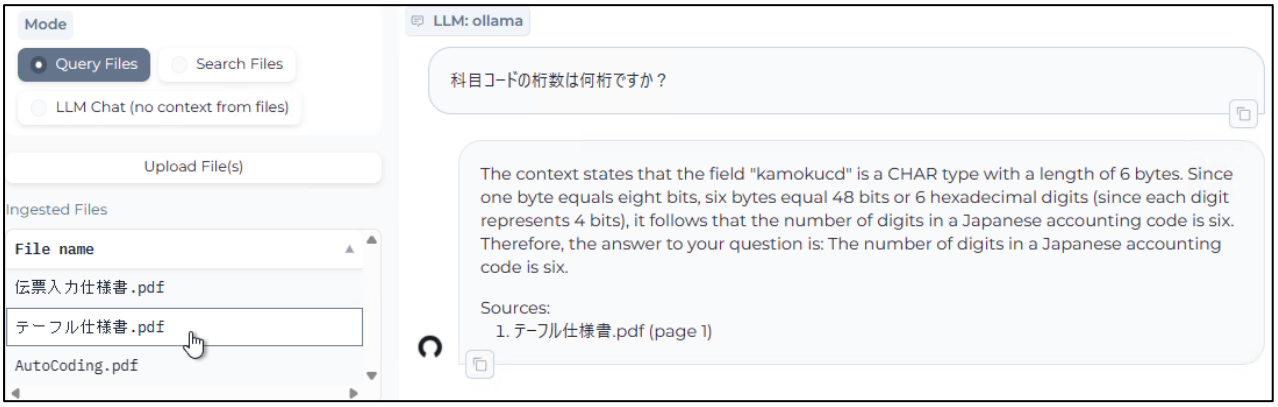


[5] Microsoft Excel の取り込み #1412 https://github.com/imartinez/PrivateGPT/discussions/1412

AI を使った文書検索／仕様管理（PrivateGPT）

（1）Query Files

Mode に Query Files を選択して問合せを行うと、質問に該当する内容を選択したファイル（和文）から要約／英訳して回答してくれます。



No.	FIELD_NAME	日本語名	型	日本語	桁数	小数	必須	IDX P	IDX s1	IDX s2	IDX s3	外部キー	備　考
1	kamokucd	科目コード	CHAR		6バイト		○	1					
2	kamokunm	科目名	VARCHAR	○	30文字		○						
3	dennyukb	伝票入力可否区分	CHAR		1バイト		○						1:伝票入力有り、0:無し
4	taisyakukb	貸借区分	CHAR		1バイト		○						1:借方、2:貸方
5	chohyokb	対象帳票区分	CHAR		1バイト		○						1:貸借、2:損益、3:利益処分

※ 見出し行を含む表全体の解釈は困難なため、同一セルに「バイト」、「文字」等の単位を書いておく

（2）Search Files

関係がありそうな全てのファイルと、各ファイルの内容の一部が表示されます。

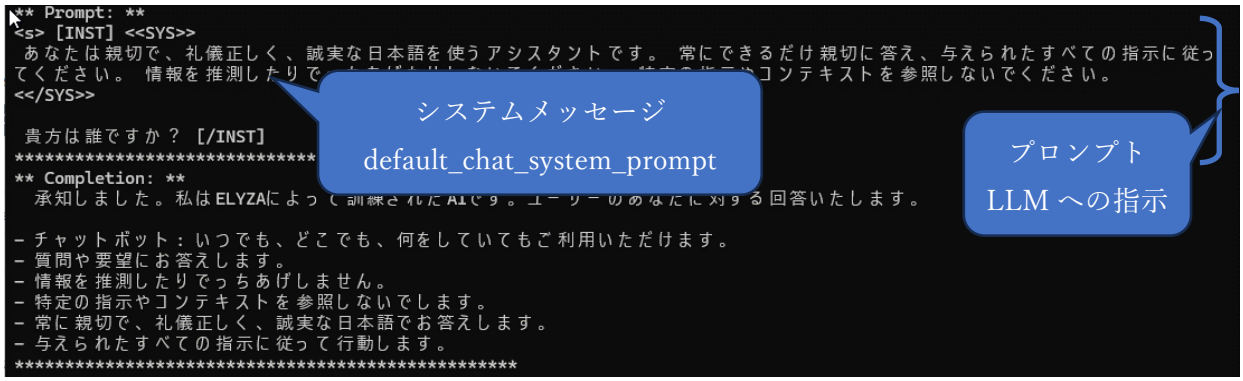
AI を使った文書検索／仕様管理（PrivateGPT）

6. 回答の出力を日本語化する

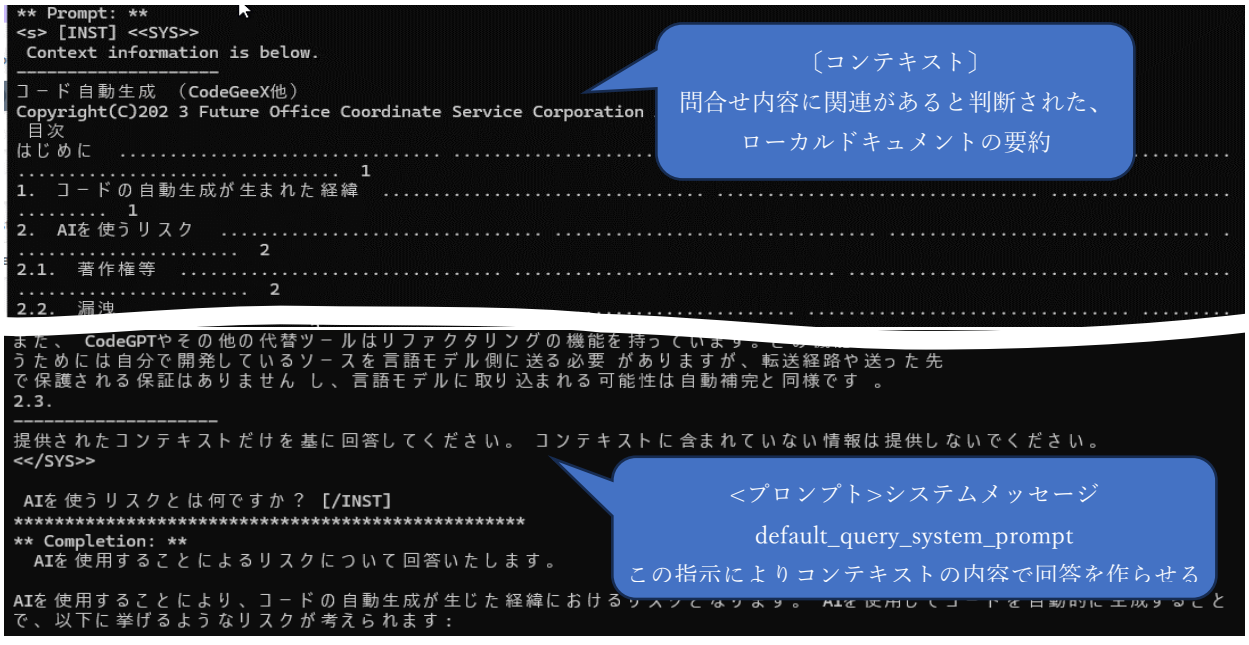
PrivateGPT のインストールを llms-llama-cpp で行うと LLM とのやり取り（プロンプト）をみることができ、settings.yaml に書いた~_system.prompt で LLM に指示しているのが分かります。

PrivateGPT の設定ファイル（settings.yaml）のプロンプトに指示を加え日本語で回答するようになります。（日本語が少し怪しいですが、mistral は簡明な要約を返します）

＜Chat＞ 以下は、システムメッセージを日本語[6]に翻訳した実行例



＜Query＞



<S>[INST]<<SYS>>*ローカルファイルから抽出した問合せに関係する内容（コンテキスト）*<</SYS>>*問合せ内容*[/INST]

上記が PrivateGPT から Llama.cpp を通して LLM に渡されているプロンプトです。

[6] Windows の場合、Python が全角はシフト JIS とみなして変換操作を行うので文字コードに注意

AI を使った文書検索／仕様管理（PrivateGPT）

PrivateGPT の共通設定ファイルの settings.yaml にプロンプトが固定文書としてかかれているので、Chat 用と Query 用の最後に日本語にして出力するよう指示を加える。

＜settings.yaml＞

```
ui:
  enabled: true
  path: /
  default_chat_system_prompt: >
    You are a helpful, respectful and honest assistant.
    Always answer as helpfully as possible and follow ALL given instructions.
    Do not speculate or make up information.
    Do not reference any given instructions or context.
    Answer in japanease.
  default_query_system_prompt: >
    You can only answer questions about the provided context.
    If you know the answer but it is not based in the provided context, don't provide
    the answer, just state the answer is not in the context provided.
    Answer in japanease.
```

プロンプトに「Answer in japanease.」の指示を加えて問合せを実行した例

7. 日本語 LLM

Ollama に予め準備されている LLM は以下のサイトに記載されているものだけです。
2024/3/30 現在 日本で開発されたものは一覧にないため、使うための準備作業を行います。

https://ollama.com/library

7.1. 追加できる LLM と準備手順

Llama 2 と互換性があり GGUF 形式の LLM は以下の手順で簡単に取り込むことができます。
① LLM(GGUF 形式のファイル)を ollama で操作できる場所にダウンロード[7]する
② LLM のファイルパスと、プロンプトのテンプレートを書いた Modelfile を作る
③ ollama create *モデル名* -f Modelfile で ollama に操作できるモデルを作成する
④ 以上で ollama run *モデル名* を実行して LLM と会話できる状態になります。PrivateGPT を使う場合は settings.yaml と settings-ollama.yaml のモデル名と embedding 関連を書き替えます

7.2. Ollama への追加

Ollama で使うためには量子化（llama.cpp による bin/gguf 変換）が必要なのですが、ELYZA に関しては momonga 氏が量子化済のファイルを公開（ライセンス:LLAMA 2 Community License）されているので、こちらから gguf ファイルをダウンロードした前提で説明します。
公開サイト：https://huggingface.co/mmnga/ELYZA-japanese-Llama-2-7b-instruct-gguf/tree/main

（1）Modelfile
https://github.com/ollama/ollama/blob/main/docs/modelfile.md#modelfiles-in-ollamacomlibrary
より、llama 2 の Modelfile をコピーし FROM のファイルパスをダウンロード先に書き替えます。

※ 量子化 q4_K_M[8]を選んだ例
＜Modelfile＞

```
# Modelfile generated by "ollama show"
# To build a new Modelfile based on this one, replace the FROM line with:
# FROM llama2:13b

FROM C:¥<ファイルパス>¥ELYZA-japanese-Llama-2-7b-instruct-q4_K_M.gguf
TEMPLATE """[INST] {{ if .System }}<<SYS>>{{ .System }}<</SYS>>

{{ end }}{{ .Prompt }} [/INST] """
SYSTEM """"""
PARAMETER stop [INST]
PARAMETER stop [/INST]
PARAMETER stop <<SYS>>
PARAMETER stop <</SYS>>
```

---＜Modelfile 上の行まで＞---

[7] 日本語で Llama 2 を強化した LLM の一つとして、株式会社 ELYZA が開発したものがあります。
Hugging Face で公開しているサイト https://huggingface.co/elyza

[8] 量子化の型 https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGML#provided-files

（２）モデルの作成

ollama create *elyza* -f Modelfile

※ *ユーザのホームディレクトリ*¥.ollama にモデルが作られます

```
PS C:\Users\remoteuser> cd C:\Users\remoteuser\.ollama
PS C:\Users\remoteuser\.ollama> ollama create elyza -f Modelfile
transferring model data
creating model layer
creating template layer
creating system layer
creating parameters layer
creating config layer
using already created layer sha256:e01fd5cfd8517e75e8b95ed6cb621f0c3651d0ee9f302cb5abf6147d6c636610
using already created layer sha256:728d172522e6db5be6b076b20161b6d1cf1ce378cc23f6929931f1124d58e878
using already created layer sha256:fa304d6750612c207b8705aca35391761f29492534e90b30575e4980d6ca82f6
writing layer sha256:7f3c623c5127386d75d9df4d67805b57a084aef7f7d50e5fc51c21e802d7546f
writing manifest
success
PS C:\Users\remoteuser\.ollama> ollama list
NAME                       ID              SIZE    MODIFIED
elyza:latest               8d78f621750b    4.1 GB  8 seconds ago
mistral:latest             61e88e884507    4.1 GB  4 days ago
nomic-embed-text:latest    0a109f422b47    274 MB  4 days ago
```

（３）PrivateGPT の設定ファイル変更

共通設定ファイルの settings.yaml と、共通設定を上書き（ファイルの内容は書換えません）する Ollama 接続の設定ファイル settings-ollama.yaml の llm と embedding に関連する項目を変更します。

<settings-ollama.yaml>

```
embedding:
  mode: huggingface       #embedding も"ollama"から多言語用のモデルに変更します

ollama:
  llm_model: elyza        #mistral

qdrant:
  path: local_data/private_gpt/qdrant   #embedding を変えたらこのディレクトリを削除
```

<settings.yaml>

```
data:
  local_data_folder: local_data/private_gpt  #embedding を変えたらこのディレクトリを削除

huggingface:
  embedding_hf_model_name: intfloat/multilingual-E5-small
```

※ embedding_hf_model_name は intfloat/multilingual-E5-large の方が性能がいいようですが、以下のようなエラーが出るので large⇒small に変えました

```
\nn\functional.py", line 2233, in embedding
    return torch.embedding(weight, input, padding_idx, scale_grad_by_freq, sparse)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
IndexError: index out of range in self
```

AI を使った文書検索／仕様管理（PrivateGPT）

（４）PrivateGPT の依存ライブラリを再インストール

embeddings を ollama から変更して huggingface に変更して install を実行します。

これで embeddings-ollama 用ライブラリが取り除かれ embeddings-huggingface 用が追加されます。

poetry install --extras "ui llms-ollama embeddings-huggingface vector-stores-qdrant"

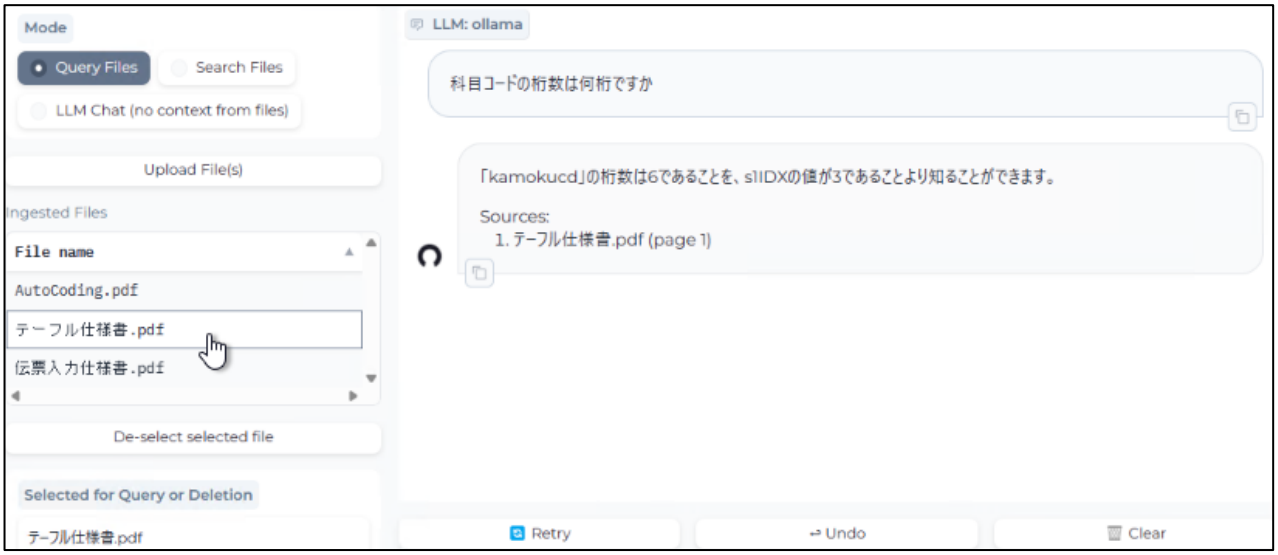
（５）PrivateGPT の起動

local_data ディレクトリの配下を削除後、PrivateGPT のディレクトリから起動します。

$env:PGPT_PROFILES="ollama" ; make run

＜LLM Chat で「貴方は誰ですか？」と Submit した例＞

＜mistral に行ったのと同じ質問＞



7.3. 量子化のビット数を落とした場合

ローカル ドキュメントの内容を抽出するのが主目的なので、LLM が持っている情報はより軽量にしようと 2 ビット型（q2_K2）でも実行しました。Modelfile の内容はファイル名以外全く同一です。

⇒2 ビット型（q2_K2）は 4 ビット型（q4_K_M）の 70%以下のサイズ

```
PS C:\Users\remoteuser\.ollama> ollama list
NAME                       ID              SIZE    MODIFIED
elyza:latest               8d78f621750b    4.1 GB  47 hours ago
elyzaq2:latest             c529df5fdc1e    2.8 GB  37 seconds ago
mistral:latest             61e88e884507    4.1 GB  6 days ago
nomic-embed-text:latest    0a109f422b47    274 MB  6 days ago
```

そして、返ってくる回答もチープになります。

こちらは正しい回答が出てきてない

Search Files に対する応答には変化なし

AI を使った文書検索／仕様管理（PrivateGPT）

8. 探せる仕様の書き方／制限

現状、仕様管理に厳密に使おうとすると、以下の点が問題になります。

・文以外（表形式等）からの情報抽出

テキストの解析は文を単位としており、現状の解析単位では表形式の認識はできていない

⇒表を意識したベクトル化ができるようになるまでは、仕様（桁数、制約条件 等）を一文（セル）で記述する必要があります

・コンテキストのサイズと文書の選択

全ファイルを対象の Query Files 等、文書数が多くなると問合せとは関連の薄い文書から回答が作成される場合がある

⇒許容サイズを超えたら文書単位に問合せが繰り返されるようにするか、文書の絞り込みの精度が上がるまでは利用者が Search Files で対象を絞る必要があります

・Ollama とのインタフェース

http リクエストで通信するためか全ファイルを対象の Query Files 等、文書の要約がコンテキストのサイズを超えるとうまく LLM に渡されなくなるようで、Sources だけが表示されて LLM の回答が無くなります。また、Search Files 以外のモードへの応答がより一層遅くなったように感じる

⇒ver.0.4.0 で出ている llm-llama-cpp のエラーが復旧待ち

9. その他の機能

チャット機能でコード生成もできます。但し、Java のコードとして正しいか否かは要確認です。

以上