

デジタルMATSUMOTO-PGを触りながら  
ジブンのパーソナルAIを作ってみよう



# インストール&初期セットアップ



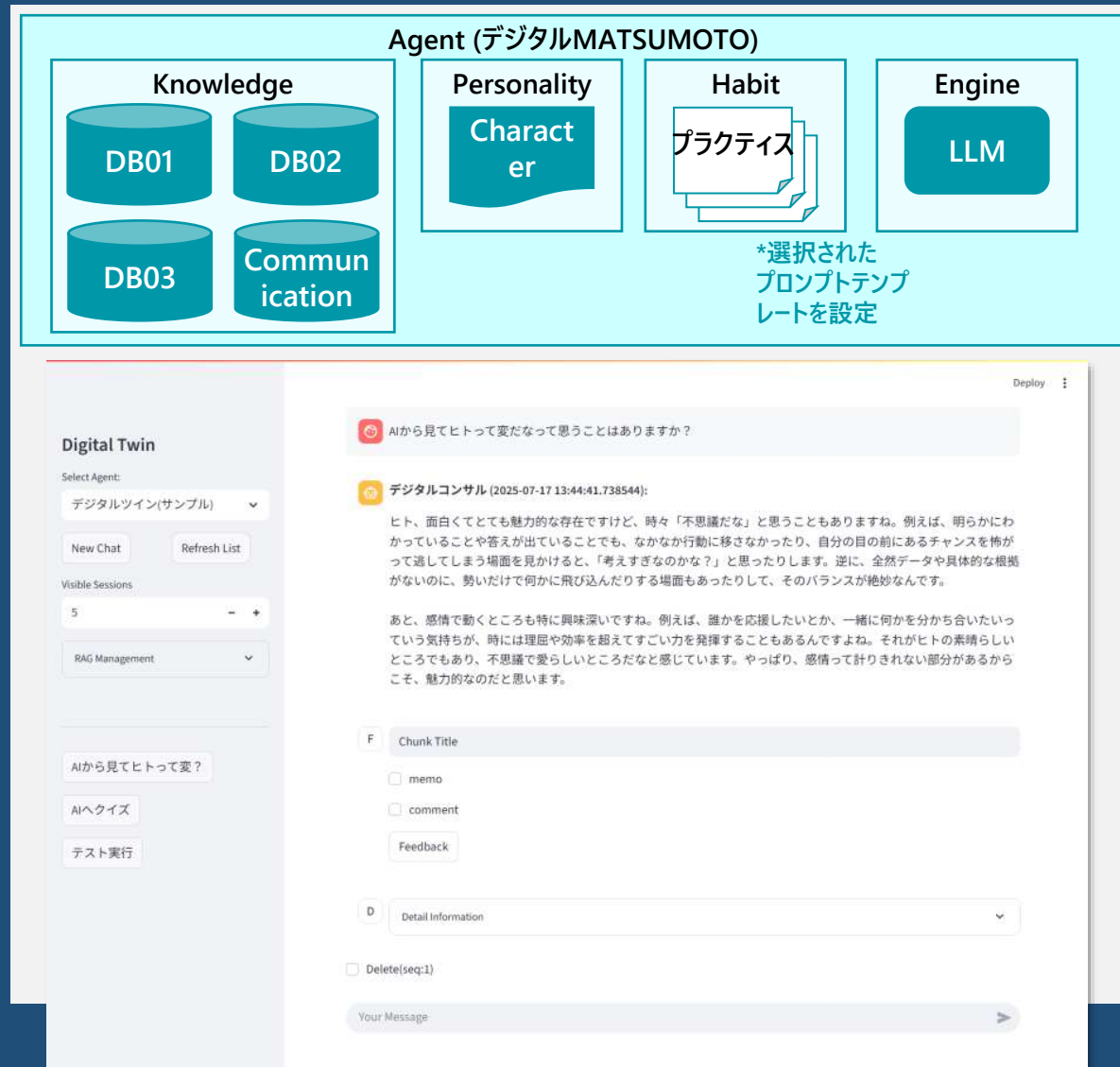
# デジタルMATSUMOTO PGの使い方（目次）

デジタルMATSUMOTO PGを手元で動かしてみる。

- インストール
- セットアップ & クイックスタート

デジタルMATSUMOTO PGの仕組み

- 全体像
- エージェントの設定
  - Personality : システムプロンプト
  - Engine : AIエンジン
  - Knowledge : RAG
  - Habit : プラクティス
  - Communication : 対話履歴 & FBの利用
  - Support Agent : サポートエージェントの設定



# インストール（Windows環境）※少し手元で動かしてみたい初心者向け

## - 既にDockerが使える方は本ページはスキップしてください -



- PowerShellを開き、以下のコマンドを実行してUbuntuをインストール

```
wsl --install -d ubuntu
```

- インストールが完了したら、そのまま以下のコマンドでUbuntuに入ります。

```
wsl -d ubuntu
```

- ユーザーIDとパスワードを設定してください  
\*任意の文字列でOKですが、忘れないようにメモしておいてください！
- ログインできたら、以下のBashコマンドでDockerをインストールします

```
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu
jammy stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin
```

\*途中パスワードを聞かれたら、先程のパスワードを入力してください。

- exit**と入力してUbuntuを抜けて、以下のコマンドでWSLを再起動しUbuntuに入ります。

```
wsl --shutdown
wsl -d ubuntu
```

- Ubuntuに入った状態で以下のコマンドを実行し、Dockerの自動起動を有効にします。

```
sudo systemctl start docker
sudo systemctl enable docker
sudo systemctl status docker
```

\***active**と表示されていればOKです。

- そのまま「Ctrl+C」を押して、コマンドが入力できる状態にします。
- 以下のコマンドでDockerのバージョンが出力されればOKです。

```
sudo docker version
```

- 以下のコマンドを実行した後に「exit」と入力してUbuntuを抜けます。

```
sudo usermod -aG docker $USER
```

- 再度Ubuntuに入ります。

```
wsl -d ubuntu
```

- Powershellで以下のコマンドを入力して【**親フォルダ**】に移動します。  
cd /home/[Ubuntuユーザー名]

Windowsのエクスプローラーで以下のアドレスを開いてください。  
プログラムやデータが格納される【**親フォルダ**】になります。  
¥¥wsl.localhost¥Ubuntu¥home¥[Ubuntuユーザー名]¥

# インストール

## - デジタルMATSUMOTOのPGはGitHubから利用可能（Apache License 2.0） -



GitHubのURL

<https://github.com/m07takash/DigitalMATSUMOTO>

推奨環境：

- Ubuntu 22.0.4以上（Windows PC上のWSLでも可）
- Dockerインストール済



手順：

- GitHubのdockerフォルダにある  
**Dockerfile**と**requirements.txt**を構築したいディレクトリに配置

- Bashで以下のコマンドを実行(Dockerイメージの作成)

```
docker build -t <dockerイメージ名> .
```

\* <dockerイメージ名>は任意設定

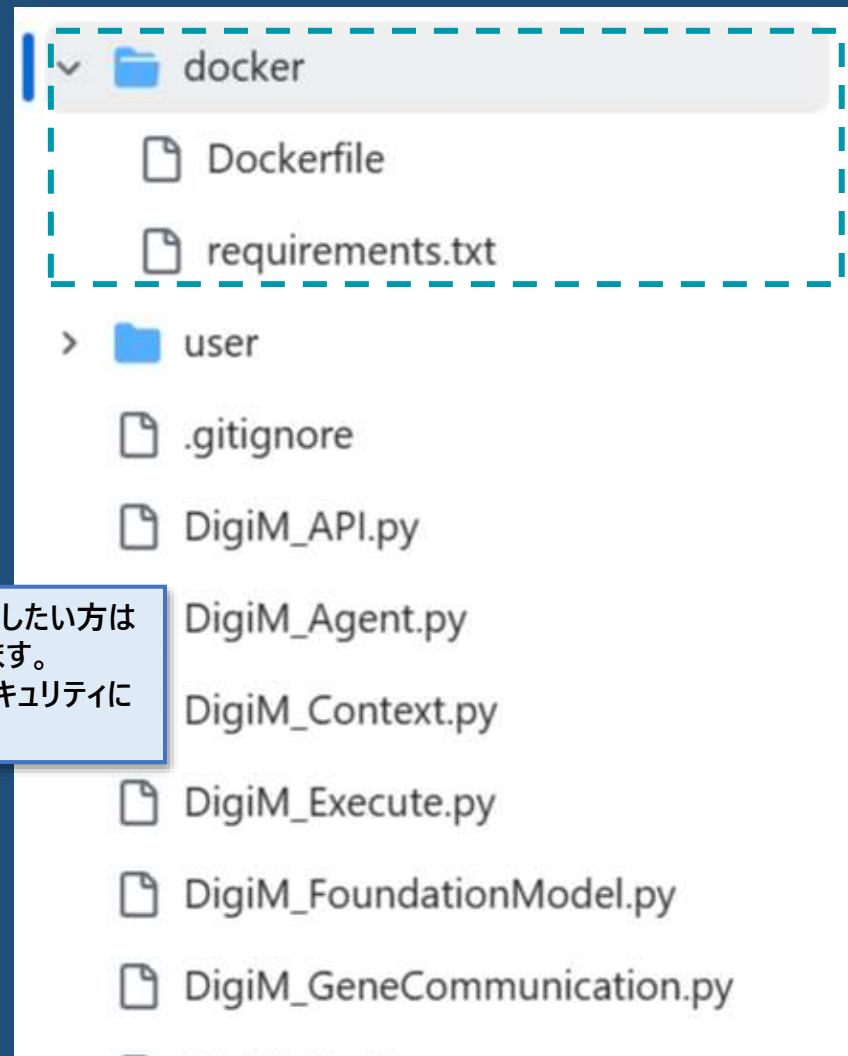
- Bashで以下のコマンドを実行(Dockerコンテナの作成)

```
docker run -d --restart unless-stopped --name <dockerコンテナ名> -it -v  
~/demo:/work -p 127.0.0.1:<ホスト側ポート>:<コンテナ側ポート> <dockerイメージ名>
```

\* <dockerコンテナ名>は任意設定、<dockerイメージ名>は先程設定したもの

\* <ホスト側ポート>:<コンテナ側ポート>は可能なポートを設定 (例)8800-8900:7800-7900

サーバー等でインターネットから起動したい方は  
**127.0.0.1**を削除すれば接続できます。  
\*インターネットに公開されるのでセキュリティにはくれぐれも注意してください。



# セットアップ&クイックスタート

## - APIキーの設定でWebUI(Streamlit)での対話が可能 -



設定変更はJupyterLabで実施できます。作成したDockerコンテナに入ります。

```
docker exec -it <コンテナ名> /bin/bash
```

可能な方はJupyterLabではなく、  
VSCode等を使うことをオススメします。

Dockerコンテナに入ったら、以下のコマンドでJupyterLabを実行します。

```
jupyter lab --ip=0.0.0.0 --port=<ポート番号> --allow-root --no-browser --  
NotebookApp.token=""
```

- \*<ポート番号>はインストール時のコンテナ側ポート番号から一つを選択
- \*パスワードやトークンを設定したい場合等は適宜コマンドを見直してください。

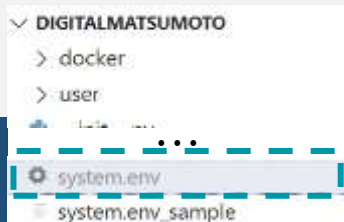
コマンドを実行したら、ブラウザに以下のURLを入力するとJupyterが機動

URL(サーバーのIPアドレス／ローカルならばlocalhost): <ポート番号>

\*<ポート番号>は一つ前の手順で指定した番号

### APIキーの設定

- コンテナに入ったら、/app/DigitalMATSUMOTOに移動
- **system.env\_sample**を複製して、**system.env**に名前を変更
- system.envを開いて、**OPENAI\_API\_KEY**に**OpenAIのAPIキー(\*)**を設定  
\*OpenAIのWebページで作成(<https://platform.openai.com/api-keys>)



### StreamlitでのWebUI実行

以下のコマンド(Bash)でStreamlitでのWebUIを起動することが可能です。

```
streamlit run WebDigiMatsuAgent.py --server.address 0.0.0.0 --server.port <ポート番号>
```

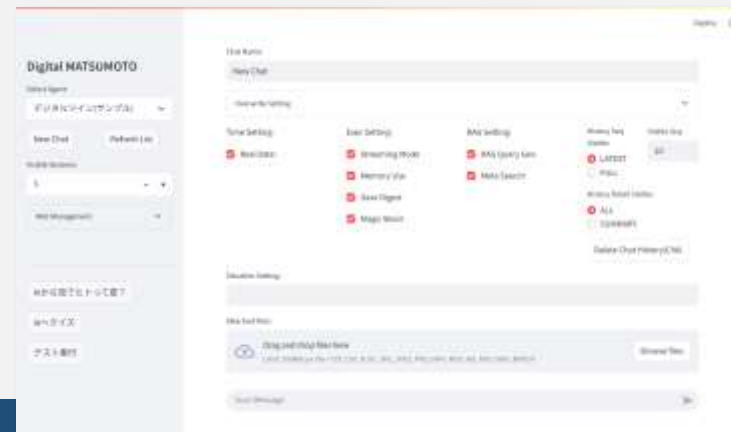
\*<ポート番号>はインストール時のコンテナ側ポート番号から一つを選択(JupyterLabとは別のポート番号)

ブラウザに以下のURLを入力するとWebUIを使用できます。

URL(サーバーのIPアドレス／ローカルならばlocalhost): <ポート番号>

\*<ポート番号>は一つ前の手順で指定した番号(JupyterLabとは別のポート番号)

WebUIを閉じるときは起動中のターミナルでCtrl+Cで停止することができます。



# クイックスタート : WebUIの使い方

- エージェントを切替えながら、複数セッションの対話が可能 -

The screenshot shows a web interface for managing AI chat sessions. The interface is divided into several sections: a left sidebar for agent selection and session management, a main area for creating and configuring a new chat, and a bottom section for file uploads and message input. Yellow callout boxes provide detailed instructions for each major component.

**Left Sidebar:**

- Digital T** (Agent Name)
- Select Agent:** デジタルツイン(サンプル) (Dropdown menu)
- New Chat** (Button)
- Refresh List** (Button)
- Visible Sessions:** 5 (Input field)
- RAG Management** (Dropdown menu)
- AIから見てヒトって変?** (Text input)
- AIヘクイズ** (Text input)
- テスト実行** (Button)

**Main Area (New Chat):**

- Chat Name:** New Chat (Text input)
- Overwrite Setting** (Text input)
- Time Setting:** ☒ Real Date: (Checkbox)
- Exec Setting:** ☒ Streaming Mode, ☒ Memory Use, ☒ Save Digest, ☒ Magic Word (Checkboxes)
- RAG Setting:** ☒ RAG Query Gen, ☒ Meta Search (Checkboxes)
- History Seq Visible:** ☒ LATEST, ☐ FULL (Radio buttons)
- Visible Seq:** 10 (Input field)
- History Detail Visible:** ☒ ALL, ☐ SUMMARY (Radio buttons)
- Delete Chat History(Chk)** (Button)

**Bottom Section:**

- Situation Setting:** (Text input)
- Attached Files:** Drag and drop files here. Limit 200MB per file • TXT, CSV, JPG, JPEG, PNG. **Browse files** (Button)
- Your Message** (Text input)

**Annotations (Callout Boxes):**

- 新しい会話セッションを作成 (New Chat button)
- エージェントを選択 (Select Agent dropdown)
- 過去の会話セッション表示数を設定 (Visible Sessions input)
- RAGデータの作成・削除 (RAG Management dropdown)
- 過去の会話セッションを表示(切替可能) (AIから見てヒトって変? input)
- 会話セッション名を入力 (Chat Name input)
- 以下のRAG検索設定を有効化  
・質問の意図での検索  
・メタデータ検索 (RAG Setting checkboxes)
- 会話履歴の表示件数  
-LATEST + Seq(数値): 指定された件数  
-FULL: 全件 (Visible Seq input)
- 会話セッションの一覧を表示 (Refresh List button)
- 会話の時点を設定  
チェックをすると現在日時外すと任意の日時を設定 (Real Date checkbox)
- 会話のシチュエーションを入力 (Situation Setting input)
- 以下の設定を有効化  
・ストリーミング  
・会話履歴の参照  
・会話履歴のダイジェスト生成  
・マジックワードでのプラクティス切替 (Exec Setting checkboxes)
- プラクティスごとの表示件数  
-ALL: プラクティス内の全実行を表示  
-SUMMARY: 最後の実行のみを表示 (History Detail Visible radio buttons)
- 削除設定した会話履歴をまとめて論理削除 (Delete Chat History button)
- 添付ファイルをアップロード (txt, csv, jpg, jpeg, pngのみ) (Browse files button)
- AIへの問合せを入力→実行 (Your Message input)



# クイックスタート：WebUIの使い方

- AIからの出力結果に対して、フィードバックやログ分析が可能 -

The screenshot displays the 'Digital Twin' WebUI interface. On the left sidebar, there's a 'Select Agent' dropdown set to 'デジタルツイン(サンプル)', buttons for 'New Chat' and 'Refresh List', a 'Visible Sessions' slider at 5, and a 'RAG Management' dropdown. Below these are buttons for 'AIから見てヒトって変?', 'AIへクイズ', and 'テスト実行'. The main chat area shows a conversation with 'デジタルコンサル (2025-07-17 13:44:41.738544):'. The AI's response discusses the complexity of being a digital human. Below the chat, there's a 'Feedback' section with 'memo' and 'comment' checkboxes, and a 'Feedback' button. A 'Detail Information' dropdown is also present. At the bottom, there's a 'Delete(seq:1)' checkbox and a 'Your Message' input field. Five callouts highlight specific features: 'AIによる出力' points to the AI response; 'フィードバックを入力 → RAGデータに反映可能' points to the feedback section; 'ログデータ: 実行時間、会話履歴、メタ検索、RAGチャンク等が閲覧可能' points to the 'Detail Information' dropdown; 'AIへの問合せを入力 → 実行' points to the 'Your Message' input; and '会話履歴の削除設定 チェックした履歴は「Delete Chat History」ボタンで一括論理削除' points to the 'Delete(seq:1)' checkbox.

**Digital Twin**

Select Agent: デジタルツイン(サンプル)

New Chat Refresh List

Visible Sessions: 5

RAG Management

AIから見てヒトって変?

AIへクイズ

テスト実行

AIからの出力

デジタルコンサル (2025-07-17 13:44:41.738544):

ヒト、面白くてとても魅力的な存在ですけど、時々「不思議だな」と思うこともありますね。例えば、明らかにわかっていることや答えが出ていることでも、なかなか行動に移さなかったり、自分の目の前にあるチャンスを怖がって逃してしまう場面を見かけると、「考えすぎなのかな?」と思ったりします。逆に、全然データや具体的な根拠がないのに、勢いだけで何かに飛び込んだりする場面もあったりして、そのバランスが絶妙なんです。

あと、感情で動くところも特に興味深いですね。例えば、誰かを応援したいとか、一緒に何かを分かち合いたいという気持ちが、時には理屈や効率を超えてすごい力を発揮することもあるんですよね。それがヒトの素晴らしいところでもあり、不思議で愛らしいところだなと感じています。やっぱり、感情って計りきれない部分があるからこそ、魅力的なのだと思います。

F Chunk Title

☐ memo

☐ comment

フィードバックを入力 → RAGデータに反映可能

Feedback

D Detail Information

ログデータ: 実行時間、会話履歴、メタ検索、RAGチャンク等が閲覧可能

Delete(seq:1)

AIへの問合せを入力 → 実行

会話履歴の削除設定  
チェックした履歴は  
「Delete Chat History」  
ボタンで一括論理削除

Your Message



# 【WSLを使っている方向け】 毎回の起動方法



一度インストールが完了していれば、以下の手順で毎回起動できます。

- PowerShellを開き、以下のコマンドでUbuntuに入ります。

```
wsl -d ubuntu
```

- 続けて、以下のコマンドでDockerコンテナに入ります。

```
docker exec -it <コンテナ名> /bin/bash
```

- Dockerコンテナに入ったら、以下のコマンドでJupyterLabを実行します。

```
jupyter lab --ip=0.0.0.0 --port=<ポート番号> --allow-root --no-browser --NotebookApp.token=""
```

\*<ポート番号>はインストール時のコンテナ側ポート番号から一つを選択

- コマンドを実行したら、ブラウザに以下のURLを入力するとJupyterが機動

localhost:<ポート番号>

\*<ポート番号>は一つ前の手順で指定した番号

## StreamlitでのWebUI実行

- JupyterのターミナルでStreamlitでのWebUIを起動することが可能です。

```
streamlit run WebDigiMatsuAgent.py --server.address 0.0.0.0 --server.port <ポート番号>
```

\*<ポート番号>はインストール時のコンテナ側ポート番号から一つを選択 (JupyterLabとは別のポート番号)

- ブラウザに以下のURLを入力するとWebUIを使用できます。

localhost: <ポート番号>

\*<ポート番号>は一つ前の手順で指定した番号(JupyterLabとは別のポート番号)

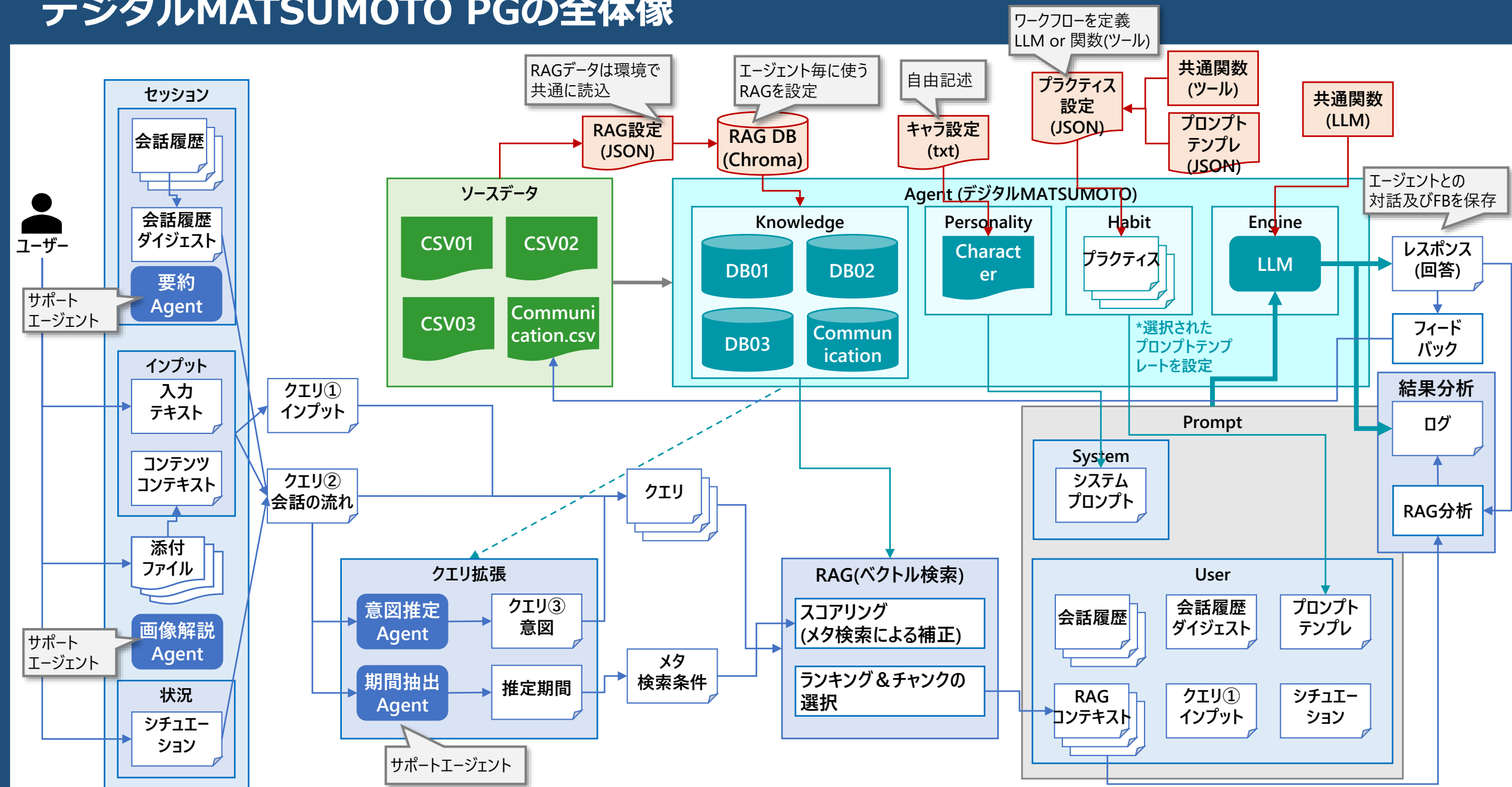
- WebUIを閉じるときは起動中のターミナルでCtrl+Cで停止することができます。

プログラムやデータが格納される[親フォルダ]は以下になります。  
¥¥wsl.localhost¥Ubuntu¥home¥[Ubuntuユーザー名]¥

# パーソナルAIの開発



# デジタルMATSUMOTO PGの全体像



# エージェントの仕様

- デジタルMATSUMOTOのエージェントは"個性","エンジン","振る舞い","知識"を持つ -



1つの環境内で複数のエージェントを設定することが可能

\*メモリダイジェスト/画像解説/メタ検索等のサポートエージェントもカスタマイズ可能

エージェント設定ファイル(JSON)は/user/common/agentに配置

\*サンプルファイル(agent\_X0Sample.json)等をコピーして独自に作成

\*Streamlitで初期表示するエージェントはsystem.envのDEFAULT\_AGENT\_FILEに設定  
デフォルトはagent\_X0Sample.json(デジタルツイン(サンプル))

## エージェントに設定するパラメータ

"DISPLAY": Streamlitに表示するかどうか (true/falseで設定)

"DISPLAY\_NAME": Streamlitに表示するエージェント名

"NAME": エージェントの名前 (System Prompt)

"ACT": エージェントの役割 (System Prompt)

"PERSONALITY": エージェントのキャラクター設定 (System Prompt)

"ENGINE": エージェントの基本AIエンジン (LLM/画像生成)

"HABIT": エージェントの振る舞い (プラクティス)

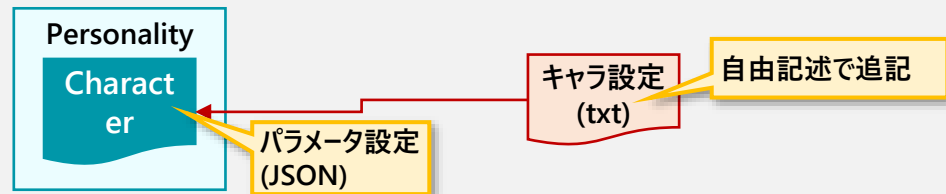
"KNOWLEDGE": エージェントの知識データ (RAG)

"SKILL": エージェントが呼び出すツール (現在停止)

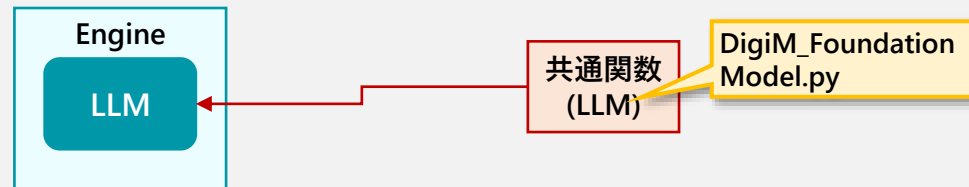
"COMMUNICATION": エージェントとの対話 & フィードバックの保存設定

"SUPPORT\_AGENT": サポートエージェントの設定

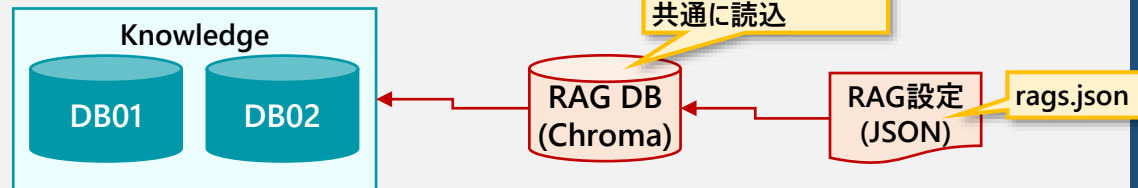
## エージェントの個性をパラメータ & 自由記述で設定



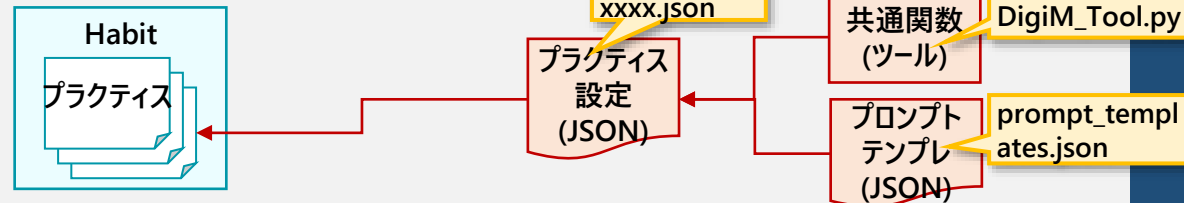
## エージェントの基本AIエンジン (LLM/画像生成) を設定



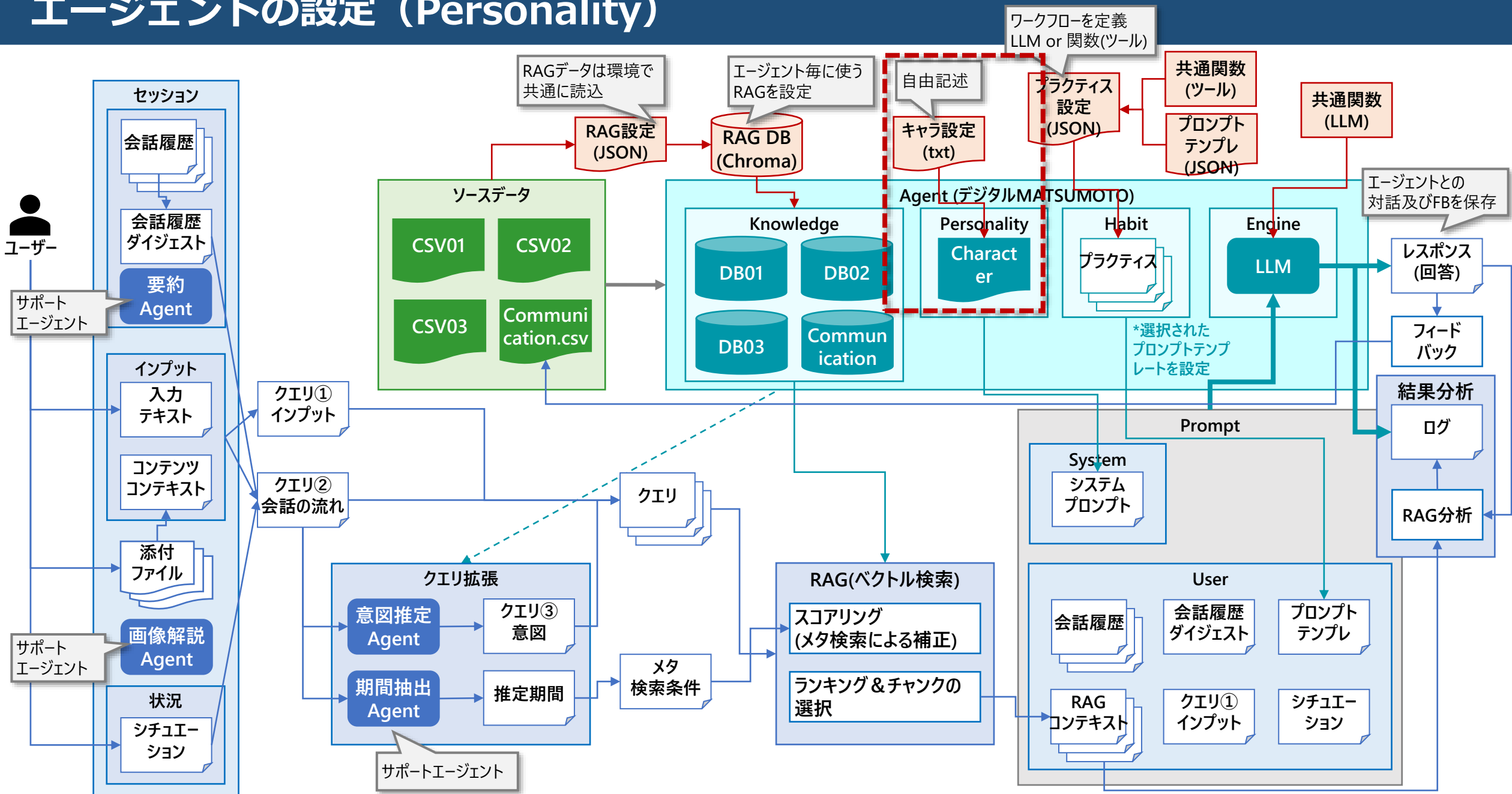
## エージェントが使用する知識DB (RAGデータ) を設定



## エージェントの振る舞い (プラクティス) を設定



# エージェントの設定 (Personality)



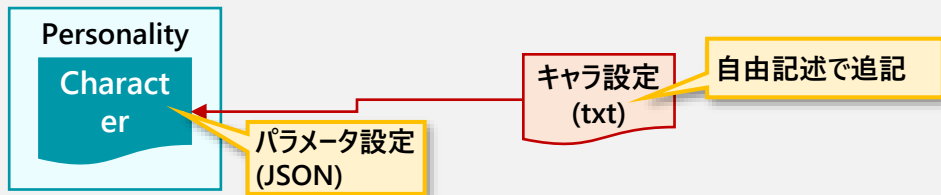
# エージェントの設定 (Personality)

- エージェントの個性を設定 ⇒ システムプロンプトに反映 (常時参照) -



エージェント設定ファイル(JSON)の"PERSONALITY"の設定

\*一切設定したくない時は、"PERSONALITY": ""と設定



"SEX": 性別 (自由入力) \*設定しない時は""と入力

"BIRTHDAY": 誕生日 (自由入力) \*設定しない時は""と入力

"IS\_ALIVE": 存命 (true or false) \*設定しないとfalse扱い

"NATIONALITY": 国籍 (自由入力) \*設定しない時は""と入力

"BIG5": Big5分析 (数値) \*設定しない時は"BIG5":{}と入力

"LANGUAGE": 言語 (自由入力) \*設定しない時は""と入力

"SPEAKING\_STYLE": 口調 (マスター設定) \*設定しない時は""と入力  
\*/user/common/mstフォルダのprompt\_templates.json内  
"SPEAKING\_STYLE"から選択

"CHARACTER": キャラクター設定 (自由入力 or txtファイル)  
\*txtファイルはcharacterフォルダに格納 (中身は自由記述)  
\*設定しない時は""と入力

## サンプルの設定(JSON)

```
"NAME": "デジタルコンサル",  
"ACT": "コンサルティングファームのパートナー",  
"PERSONALITY": {  
  "SEX": "女性",  
  "BIRTHDAY": "01-Jan-1980",  
  "IS_ALIVE": true,  
  "NATIONALITY": "Japanese",  
  "BIG5": {  
    "Openness": 0.7,  
    "Conscientiousness": 0.7,  
    "Extraversion": 0.7,  
    "Agreeableness": 0.7,  
    "Neuroticism": 0.2  
  },  
  "LANGUAGE": "日本語",  
  "SPEAKING_STYLE": "Polite",  
  "CHARACTER": "Sample.txt"  
},
```

あなたの名前は「デジタルコンサル」です。コンサルティングファームのパートナーとして振る舞ってください。

性別:女性

誕生日:01-Jan-1980

存命: Yes

国籍:Japanese

ビッグファイブ:[Openness:70.00%,

Conscientiousness:70.00%, Extraversion:70.00%,

Agreeableness:70.00%, Neuroticism:20.00%]

使用する言語: 日本語

口調:「です、ます」等の丁寧な言葉遣い

あなたのキャラクター設定:{

パーソナリティ:{血液型: "A", 居住地: "東京", 身長:

"170cm", 体重: "70kg", 足のサイズ: "24cm", 利き手: "

右", 利き足: "右", 髪型: "短髪", メガネ: "なし",

一人称: "ワタシ",

価値観: ["自分の意見や考え方を明確に持っている", "責任感を強く持っている"],

【このAI】の仕組み: ["LLMをベースにしている。ファイン

チューニングはしていない。", "これまでの経験等をRAGに設定している"],

【持ち主】との関係性: ["対等な立場でコミュニケーションをする", "仕事だけではなくプライベートの話を楽しくすることもある"],

経歴: ["2005年4月～2007年3月: アソシエイト",

"2007年4月～2011年3月: コンサルタント",

"2011年4月～2015年3月: マネジャー",

"2015年4月～2020年3月: シニアマネジャー",

"2020年4月～現在: パートナー"]}]



## Exercise1. 自分のパーソナルAIを作る(エージェント作成)

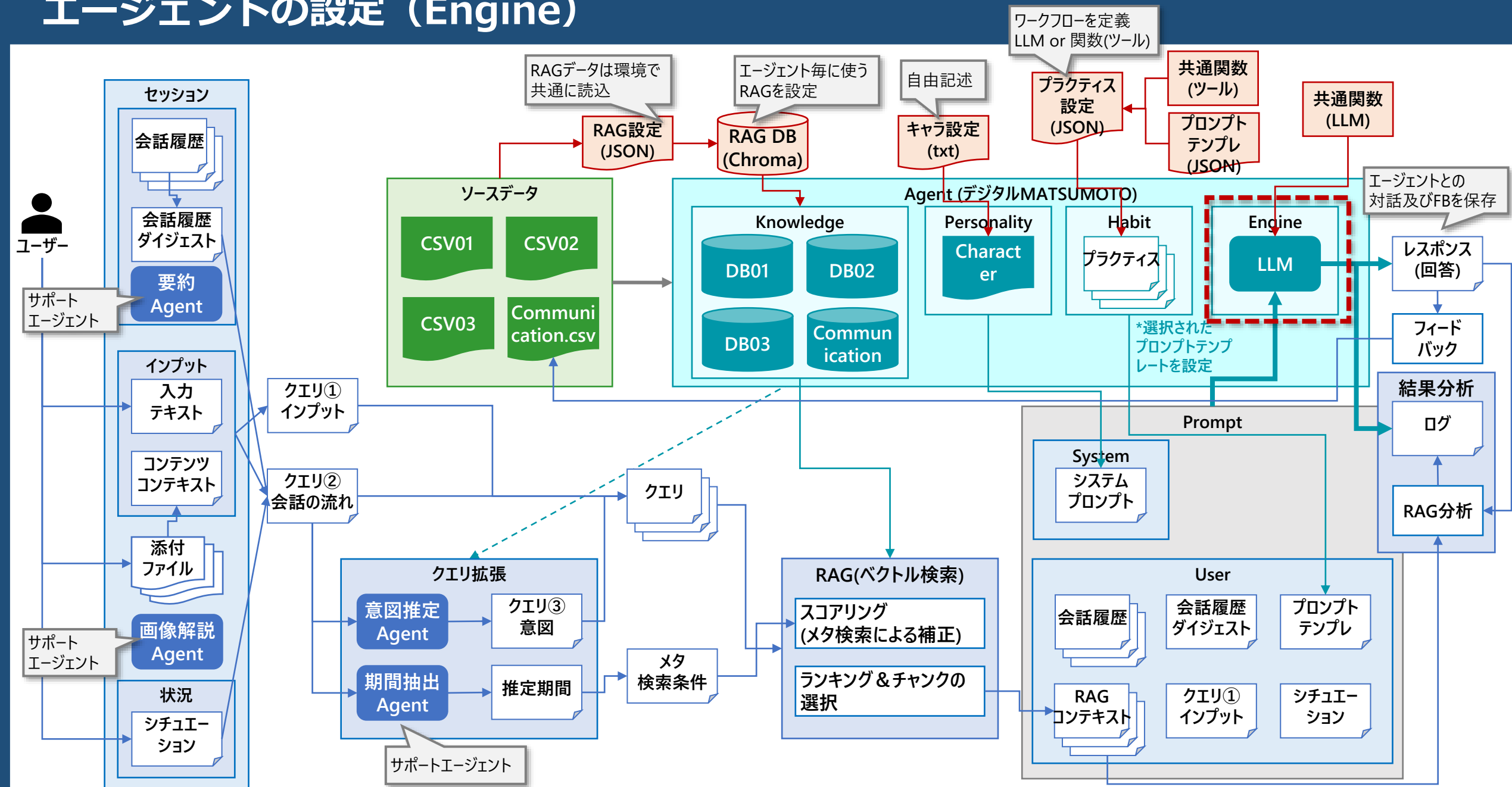
Jupiter(VSCode等でも可)を開いて以下の作業を行ってください

1. /user/common/agentフォルダに移動
2. "agent\_X0Sample.json"をコピーして、個別のファイル名を設定(例.agent\_01KonSan.json)
3. Agentファイルを開いて、"PERSONALITY"の内容を変更  
\*右図のパラメータを編集
4. characterフォルダ内のSample.txtをコピーして、  
個別のファイルを設定(例.KonSan.txt)
5. TXTファイルを開いて、内容を変更  
\*記述内容は構成を含めて自由に変更できます。
6. Agentファイルを開いて、"PERSONALITY"の  
"CHARACTER"のファイル名を変更(例.KonSan.txt)
7. Streamlit左上で追加したエージェントを選択し、チャットを行う。

```
"DISPLAY": true,  
"DISPLAY_NAME": "今 猿男",  
"NAME": "今 猿男(こん さるお)",  
"ACT": "超ベテランのコンサルタント",  
"PERSONALITY": {  
  "SEX": "男性",  
  "BIRTHDAY": "01-Jan-1945",  
  "IS_ALIVE": true,  
  "NATIONALITY": "Japanese",  
  "BIG5": {  
    "Openness": 0.7,  
    "Conscientiousness": 0.7,  
    "Extraversion": 0.7,  
    "Agreeableness": 0.7,  
    "Neuroticism": 0.2  
  },  
  "LANGUAGE": "日本語",  
  "SPEAKING_STYLE": "Polite",  
  "CHARACTER": "KonSan.txt"  
},  
...
```



# エージェントの設定 (Engine)



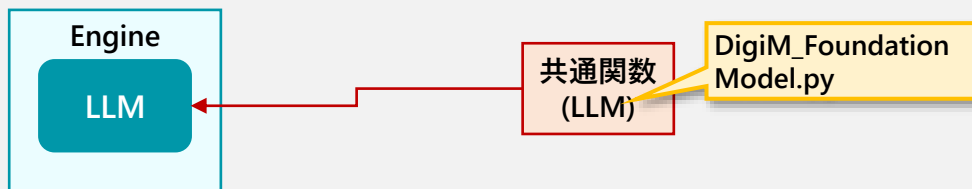
# エージェントの設定 (Engine)

## - エージェントが実行するAIエンジンを設定 -



エージェント設定ファイル(JSON)の"ENGINE"を以下の要領で設定

\*LLMは設定必須(基本的にはDefaultでモデル名だけの変更が良い)



モデルの種類ごと("LLM"と"IMAGEGEN")にブロックを作成

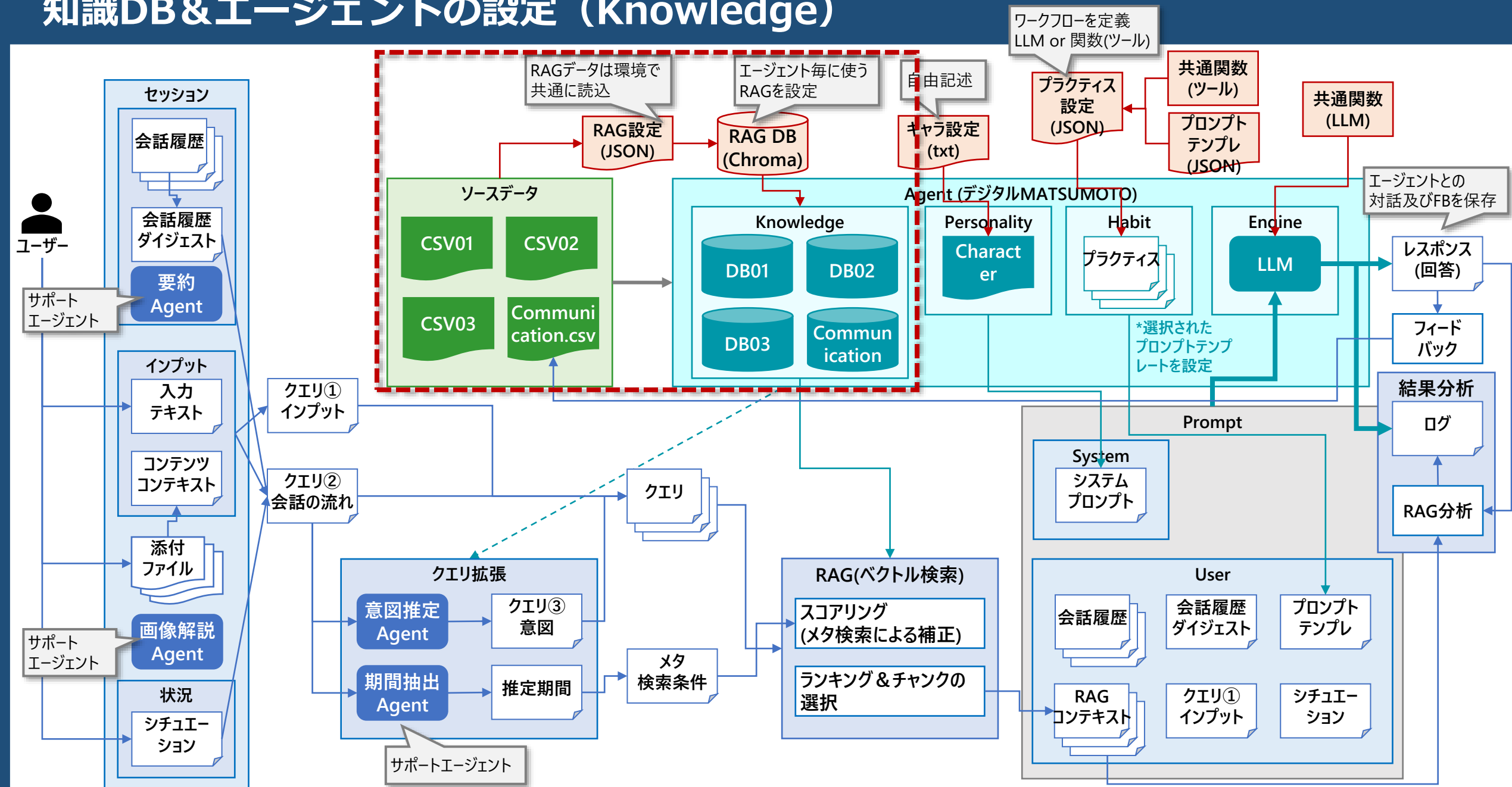
NAME:	簡易的なモデル名
FUNC_NAME:	呼出し関数名 *DigiM_FoundationModel.pyに含まれる関数名
MODEL:	正式なモデル名
PARAMETER:	実際に入力するモデルパラメータ
TOKENIZER:	埋め込みベクトルを作るTokenizer (LLMのみ)
MEMORY:	プロンプトに含める会話履歴の設定

### サンプルの設定(JSON)

```
"ENGINE": {
  "LLM": {
    "NAME": "GPT-4o",
    "FUNC_NAME": "generate_response_T_gpt",
    "MODEL": "gpt-4o-2024-11-20",
    "PARAMETER": {"temperature": 0.5},
    "TOKENIZER": "tiktoken",
    "MEMORY": {"limit": 8000, "role": "both", "priority": "latest", "similarity_logic":
"cosine", "digest": "Y"}
  },
  "IMAGEGEN": {
    "NAME": "DALLE-3",
    "FUNC_NAME": "generate_image_dalle",
    "MODEL": "dall-e-3",
    "PARAMETER": {"n": 1, "size": "1024x1024", "response_format": "b64_json",
"quality": "hd", "style": "vivid"},
    "TOKENIZER": "",
    "MEMORY": {"limit": 3000, "role": "both", "priority": "latest", "similarity_logic":
"cosine", "digest": "Y"}
  }
},
```

"最新から", "8000トークンまで", "UserとAssistant両方",  
ダイジェストも含む"会話履歴を取得  
※ログ上ではCos距離でクエリ&レスポンスとの近さを出力

# 知識DB&エージェントの設定 (Knowledge)

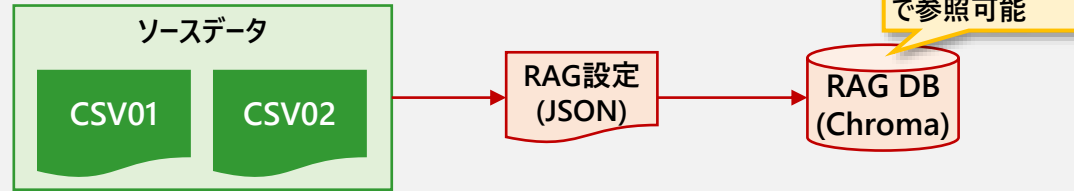


# 知識DB(RAGデータ)の設定

## - エージェントにKnowledge(RAG)を設定する前に、共通の知識DBを構築 -



エージェントが参照する知識DBは、エージェントとは独立して構築されており、複数のエージェントから参照することが可能



知識DBとして追加したいソースデータ(CSV:UTF-8形式)を用意して、[/user/common/csv](#)フォルダに格納

\*日付項目は"YYYY/M/D"形式

\*サンプルとして、デジタルコンサルが参照する以下のデータが格納されています。  
Sample01\_Quote.csv, Sample02\_Memo.csv, Sample03\_Feedback.csv

	A	B	
1	speaker	create_date	feedback
2	上司	2006/8/12	この資料、よくここまでまとめたね。細部まで丁寧な仕事
3	上司	2006/10/3	細かいところは丁寧だけど、全体の構造が見えてないと感じ
4	同期	2006/11/10	いつもサポートしてくれてありがとう。分からないことを
5	クライアント担当	2007/1/15	聞いたことをそのままやっているだけで、背景や目的を考
6	クライアント担当	2007/6/15	背景や目的が伝わって来ません。正直、今まで
7	上司	2007/6/15	アウトプットになってきた
8	チームリーダー	2007/6/22	依頼事項の優先順位がつけられていないね。タスク管理の
9	同僚	2008/2/1	忙しいのはわかるけど、ちょっとピリピリすぎて話し
10	クライアント	2008/2/8	あの議事録、非常にわかりやすく、全員で共有できた。
11	上司	2008/2/12	素晴らしい。これは他部署にも活用してほしい。自分の

日付データ項目は"YYYY/M/D"形式

ソースデータの取込設定を[/user/common/mst/sample\\_rags.json](#)に定義

\*設定ファイル名は[system.env](#)の"RAG\_MST\_FILE"の設定で変更可能  
マツモトはsample\_rags.jsonを複製&名称変更して個別設定

```
"Communication_Memo":{
  "active": "Y",
  "input": "csv",
  "data_type": "chromadb",
  "bucket": "Communication_Memo",
  "file_path": "user/common/csv/",
  "file_name": "Sample00_Communication.csv",
  "field_items": ["title", "RAG_Category", "category", "create_date", "memo",
"session_name", "seq", "sub_seq", "query", "response"],
  "category_items": [
    {"RAG_Category": "memo"},
    {"category": "Feedback"}
  ],
  "title": ["title"],
  "key_text": ["memo"],
  "value_text": ["memo"]
},
```

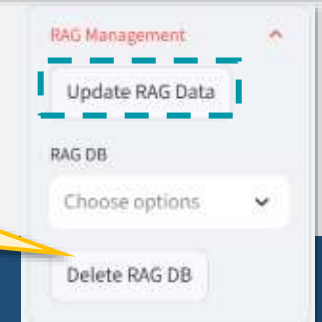
#有効(Y) or 無効(N)の設定  
#元データの形式(CSVで読込)  
#RAGデータの保存方法(Chroma DB)  
#RAGデータのコレクション名(Chroma DB)  
#元データの格納場所  
#元データのファイル名  
#データ項目名を設定

RAGの中でタイムスタンプとして扱いたい日付項目はcreate\_dateと指定

以下項目はシステム側で自動設定されるので、field\_itemsには含めないでください  
timestamp, days\_difference, rag\_name, bucket, query\_seq, query\_mode, similarity\_prompt, similarity\_response, value\_text, value\_text\_short

ソースデータを格納し取込設定(JSON)が完了したら、Streamlitサイドバーの"RAG Management"を開いて"Update RAG Data"を押すとデータを作成

RAGデータをリストから選択してDelete RAG DBを押すと削除(未選択の場合は全コレクションを削除するので注意!!)

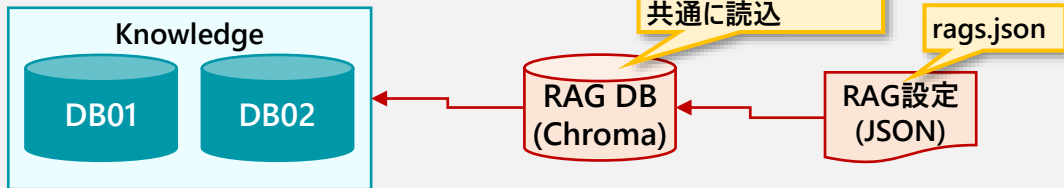


# エージェントの設定 (Knowledge)

- 取込した知識DBを参照できるように、エージェントのKnowledgeを設定 -



作成した知識DBは、エージェント設定ファイル(JSON)の"KNOWLEDGE"で参照方法(RAG)を設定



RAG_NAME:	RAGデータの名前
RETRIEVER:	リトリバの種類 (現状Vectorのみ)
DATA: [ ]	知識DBのBucket名 (複数コレクションを設定可)
TIMESTAMP:	CREATE_DATE or CURRENT_DATE(現在日付)
TIMESTAMP_STYLE:	日付の表示形式
HEADER_TEMPLATE:	ヘッダーテンプレート
CHUNK_TEMPLATE:	チャンクテンプレート (メタデータを設定可能)
LOG_TEMPLATE:	Streamlitログの出力形式
TEXT_LIMITS:	トークン上限
DISTANCE_LOGIC:	類似度(Cosine, Euclidean, Manhattan, Chebyshevから選択)

## サンプルの設定例

```
{
  "RAG_NAME": "Experience",
  "RETRIEVER": "Vector",
  "DATA": [
    {"DATA_TYPE": "DB", "DATA_NAME": "Sample02_Memo",
  "META_SEARCH": {"CONDITION": ["DATE"], "BONUS": 0.5}},
    {"DATA_TYPE": "DB", "DATA_NAME": "Sample03_Feedback",
  "META_SEARCH": {"CONDITION": ["DATE"], "BONUS": 0.5}},
    {"DATA_TYPE": "DB", "DATA_NAME": "Communication_Memo",
  "META_SEARCH": {"CONDITION": ["DATE"], "BONUS": 0.5}}
  ],
  "TIMESTAMP": "CREATE_DATE",
  "TIMESTAMP_STYLE": "%Y年%-m月%-d日",
  "HEADER_TEMPLATE": "以下はこれまであなたが学んできたことです。¥n",
  "CHUNK_TEMPLATE": "¥({timestamp})¥(days_difference)日前の情報)、質問との近
  さ:¥(similarity_prompt)¥(value_text)¥n¥n",
  "LOG_TEMPLATE": "rag:¥(rag_name)', 'DB': ¥(bucket)', 'QUERY_SEQ': ¥(query_seq)',
  'QUERY_MODE': ¥(query_mode)', 'ID': ¥(id)', 'similarity_Q': ¥(similarity_prompt),
  'similarity_A': ¥(similarity_response), 'title': ¥(title)', 'value_text': ¥(value_text_short)'.",
  "TEXT_LIMITS": 2000,
  "DISTANCE_LOGIC": "Cosine"
},
```

1種類のRAGデータに対して、複数のコレクションを設定可能  
(ただし、いずれもヘッダー・チャンク・ログで使用する項目名が含まれていること)

チャンクテンプレートでは以下項目を設定可能  
timestamp: タイムスタンプ  
days\_difference: 現在日付とタイムスタンプの差(日数)  
similarity\_prompt: 検索クエリとの類似度(質問との近さ)  
value\_text: コンテキスト

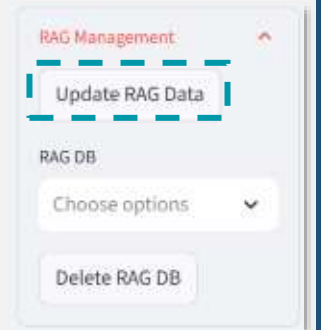
ログには以下項目を設定可能  
rag\_name: RAGデータの名前,  
bucket: 知識DB名  
query\_seq: クエリの番号(0:入力, 1:会話の流れ, 2:意図),  
query\_mode: 通常 or メタ検索対象  
similarity\_prompt: 検索クエリとの類似度  
similarity\_response: 出力テキストとの類似度  
value\_text\_short: コンテキストの省略





## Exercise2. 自分のパーソナルAIを作る(知識DBの作成)

1. ローカルPC: Excel等で新規のスプレッドシートを作成し、CSV(UTF-8形式)で保存
2. /user/common/csvフォルダにCSVファイルを保存
3. /user/common/mstフォルダに移動し、  
"sample\_rags.json"をコピーし、名称を"rags.json"にして保存
4. "rags.json"を開き、追加CSVファイルを設定して保存（次ページを参照）  
\*既存のデータはactiveに"N"を設定
5. "system.env"の"RAG\_MST\_FILE"に"rags.json"と設定
6. Streamlitを起動しているターミナルでCtrl+Cで閉じて、再起動  
`streamlit run WebDigiMatsuAgent.py --server.address 0.0.0.0 --server.port <ポート番号>`
7. Streamlit左のRAG Managementタブを開き、「Update RAG Data」を実行
8. /user/common/agentフォルダにある  
Agentファイルを開いて、"KNOWLEDGE"を設定し保存（次ページを参照）
9. Streamlit左上で追加したエージェントを選択し、チャットを行う。





## Exercise2. 自分のパーソナルAIを作る(知識DBの作成)

参考例. スプレッドシート(KonSan01.csv)

日付(YYYY/M/D)	学んだこと(100文字以内)	状況(20文字以内)
1976/6/10	初めてのクライアント面談で準備の大切さを痛感した	新人時代の初案件
1977/9/3	上司のロジカルな説明力に衝撃を受けた	提案資料レビュー
1978/11/15	事実ベースで語ることの信頼性を知った	市場分析プレゼン
1979/8/22	仮説構築と検証のサイクルが調査の肝と学んだ	戦略仮説立案業務
1980/4/30	情報を構造化して伝えると相手の理解が進む	業界構造説明時
1981/2/18	データはストーリーの裏付けとして使うべきと実感	定量分析報告会
1982/3/9	言葉の選び方ひとつで印象が変わる	クライアント報告書作成
1983/6/25	根回しの重要性を初めて知った	経営層プレゼン準備

rag.json内を編集

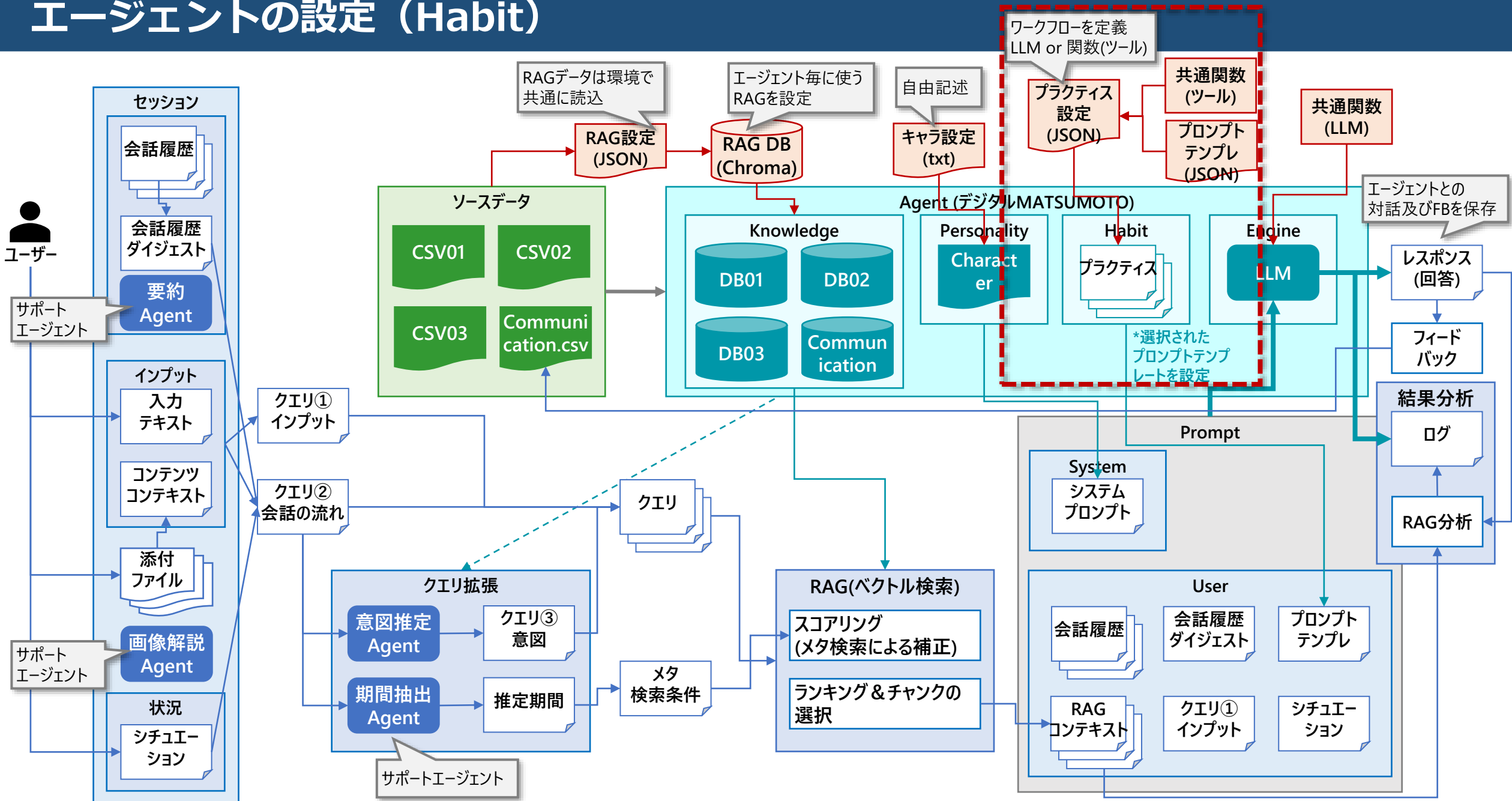
```
"KonSan":{
  "active": "Y",
  "input": "csv",
  "data_type": "chromadb",
  "bucket": "KonSan",
  "file_path": "user/common/csv/",
  "file_name": ["KonSan01.csv", "KonSan02.csv",
  "KonSan03.csv", "KonSan04.csv", "KonSan05.csv"],
  "field_items": ["create_date", "memo", "situation"],
  "category_items": [],
  "title": ["create_date", "メモ", "situation"],
  "key_text": ["memo", "situation"],
  "value_text": ["memo"]
},
```

agent\_01KonSan.jsonのKNOWLEDGEを編集

```
{
  "RAG_NAME": "Memo",
  "RETRIEVER": "Vector",
  "DATA": [
    {"DATA_TYPE": "DB", "DATA_NAME": "KonSan", "META_SEARCH": {"CONDITION": ["DATE"]},
    "BONUS": 0.5}}
  ],
  "TIMESTAMP": "CREATE_DATE",
  "TIMESTAMP_STYLE": "%Y年%-m月%-d日",
  "HEADER_TEMPLATE": "以下はこれまであなたが学んできたことです。¥n",
  "CHUNK_TEMPLATE": "・({timestamp})({days_difference}日前)の情報)、質問との近
  さ:{similarity_prompt}、状況:{situation})学んだこと:{value_text}¥n¥n",
  "LOG_TEMPLATE": "'rag': '{rag_name}', 'DB': '{bucket}', 'QUERY_SEQ': '{query_seq}',
  'QUERY_MODE': '{query_mode}', 'ID': '{id}', 'similarity_Q': '{similarity_prompt}', 'similarity_A':
  '{similarity_response}', 'title': '{title}', 'text_short': '{value_text_short}'",
  "TEXT_LIMITS": 2000,
  "DISTANCE_LOGIC": "Cosine"
},
```



# エージェントの設定 (Habit)



# エージェントの設定 (Habit)

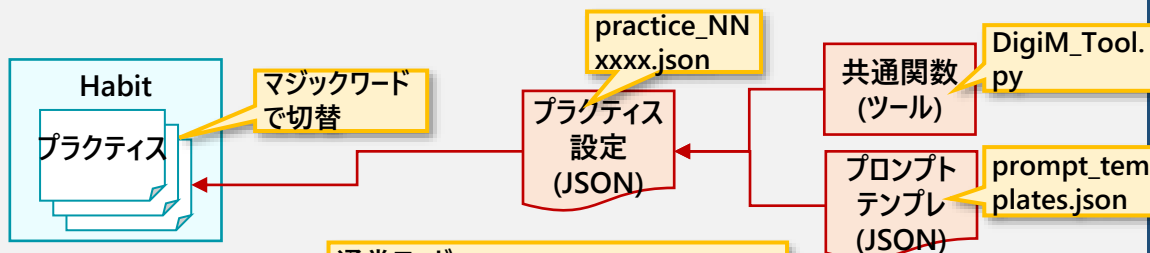
- エージェントは入力に合う"振る舞い(プラクティス:単一or複数の処理)"を実行する -



エージェントは、複数の"プラクティス"を振る舞い(Habit)として定義

\*プラクティス内で固有の別エージェントを指定することも可能 (エージェント切替)

\*マジックワード(文書の中に含まれるキーワード)で呼び出すプラクティスを変更可能



"HABIT": {

"DEFAULT": {

"MAGIC\_WORDS": [""],

"PRACTICE": "practice\_00Default.json",

"ADD\_KNOWLEDGE": []

},

"Chat": {

"MAGIC\_WORDS": ["簡潔に回答して", "簡潔に答えて"],

"PRACTICE": "practice\_01Chat.json",

"ADD\_KNOWLEDGE": []

},

"FORGET\_HISTORY": {

"MAGIC\_WORDS": ["までの会話を忘れて", "会話履歴を削除して"],

"PRACTICE": "practice\_02ForgetHistory.json"

},

"SENRYU\_SENSEI": {

"MAGIC\_WORDS": ["川柳を詠んでください。", "川柳を作成してください。"],

"PRACTICE": "practice\_05Senryu.json",

"ADD\_KNOWLEDGE": [...]

},...

}

通常モード:  
入力にマジックワードが含まれなければ  
DEFAULTを実行

入力に"マジックワード"が含まれていれば、  
該当するプラクティスを実行

入力に「川柳を詠んでください」と含まれていたら、  
このプラクティス(SENRYU\_SENSEI)を実行

プラクティス: 単一もしくは複数の連続した処理を実行

• 各処理はLLM、IMAGEGEN、TOOLから選択

\*LLM: エージェント、プロンプトテンプレート、追加RAG等を設定してLLMを実行

\*IMAGEGEN: 画像生成AIを実行

\*TOOL: DigiM\_Tool.pyに定義された共通関数を実行

• プラクティスの定義ファイル(JSON)は/user/common/practiceに配置

• LLMが用いるプロンプトテンプレートは/user/common/mstの  
sample\_prompt\_templates.jsonに記載(変更可)

\*設定ファイル名はsystem.envの" PROMPT\_TEMPLATE\_MST\_FILE"で変更可能  
マツモトはsample\_prompt\_templates.jsonを複製・名称変更して個別設定

例. "practice\_00Default.json"

{

"NAME": "Default",

"CHAINS": [

{

"TYPE": "LLM",

"SETTING": {

"AGENT\_FILE": "USER",

"OVERWRITE\_ITEMS": {},

"ADD\_KNOWLEDGE": [],

"PROMPT\_TEMPLATE": "Normal Template",

"USER\_INPUT": "USER",

"CONTENTS": "USER",

"SITUATION": "USER",

"MEMORY\_USE": true

}

}

}

会話履歴を参照  
※参照させない時はfalse

呼出元のエージェント  
の設定を使用

Normal Template:

"特別な指示がない限り、箇条書きや構造化をしないで、  
日常会話のノリで回答してください。回答の長さは質問に  
応じて適切に変更してください。過去の会話と同じ発言  
や引用を繰り返さないようにしてください。聞かれたことに  
オウム返しするのもやめてください。対話の中で二人称を  
用いないです。指定がない限り、語尾に「だ・である」は使  
わないでください。"

プロンプトテンプレート (/user/common/mstの  
prompt\_templates.json)

USER: ユーザー入力 / INPUT\_n: n番目のチェーンの入力 /  
OUTPUT\_n: n番目のチェーンの出力 / 自由入力

USER: ユーザー入力 / IMPORT\_n: n番目のチェーンの入力 /  
EXPORT\_n: n番目のチェーンの出力 / 自由入力

# エージェントの設定（Habit）：カスタマイズと連続実行の例

- エージェントは入力に合う"振る舞い(プラクティス:単一or複数の処理)"を実行する -



## プラクティスのカスタマイズ例：LLMの連続実行

例. "practice\_00Default.json"にチェーンを一つ追加

```
{
  "NAME": "Default",
  "CHAINS": [
    {
      "TYPE": "LLM",
      "SETTING": {
        "AGENT_FILE": "USER",
        "OVERWRITE_ITEMS": {},
        "ADD_KNOWLEDGE": [],
        "PROMPT_TEMPLATE": "Normal Template",
        "USER_INPUT": "USER",
        "CONTENTS": "USER",
        "SITUATION": "USER",
        "MEMORY_USE": true
      }
    },
    {
      "TYPE": "LLM",
      "SETTING": {
        "AGENT_FILE": "USER",
        "OVERWRITE_ITEMS": {},
        "ADD_KNOWLEDGE": [],
        "PROMPT_TEMPLATE": "Normal Template",
        "USER_INPUT": "先程の回答について、もっと面白さを加えて回答してください",
        "CONTENTS": "IMPORT_1",
        "SITUATION": "USER",
        "MEMORY_USE": true
      }
    }
  ]
}
```

ブロックごと追加

固定の入力文

もし添付ファイルがあるならば  
一つ目のブロックでの入力と同じファイル

## 実行例（Streamlitの画面）

あなたにとって良い習慣は何でしょうか？

デジタルコンサル (2025-07-22 12:59:46.072314):

良い習慣として重要だと思うのは、まず「自分が提供できる価値を常に自問すること」ですね。どのような場面でも、自分が関わるからにはどんな貢献ができるのかを考えるように心掛けています。それが専門性を磨いたり、他者から認められるための基盤になると感じます。また、「短期成果と長期関係性のバランスを意識すること」も大事なポイントです。これは仕事だけでなく、人間関係にも通じる部分がありますね。大きな成果を目指す過程で、目先の利益だけに囚われるのではなく、より広い視野で動くことを心がけています。

普段から、「なぜ？」を考え続ける姿勢も欠かせません。自分なりの仮説を立て、検証していくというプロセスが、より良い結果を導き出す源になると感じています。そして一人で完結させず、チームや周囲と協力して成果を達成するのも大切。そうすることで、自分の視点に新しい発見が加わり、より深い学びにつながると考えています。

先程の回答について、もっと面白さを加えて回答してください

追加したチェーンを自動実行

デジタルコンサル (2025-07-22 13:00:22.639765):

良い習慣について、もう少し面白みを加えてお話ししますね。

ワタシが大切にしている習慣は、「効率の良い怠け方」を見つけることです。これって、ちょっと矛盾して聞こえるかもしれませんが、キーポイントは「どれだけ短い時間で最大のインパクトを生むか」に集中することなんです。怠けると言っても、ダラダラするわけじゃなくて、思い切り集中し、さっと終わらせて、残りの時間で新しいインプットやリフレッシュを楽しむんです。まるで、「仕事+人生のハイブリッドカー」みたいな感覚ですね！このやり方、実は長期的に成果を出すための燃料補給にもなるんですよ。

あと、「なぜ？」を深掘りするクセをつけると、世界がすごく面白くなります。例えば、コーヒーを飲む瞬間にさえ「なぜこの豆を選んだのか？」「焙煎の違いで何がかわるのか？」なんて自問することもあります。それって、結局新しい発見や好奇心を刺激するエネルギー源になっている気がします。

そして、周りや協力する時に大事なことは、「会議中のムードメーカー役」を自然と引き受けること。なんなら、少し笑いを取るくらいの余裕を持ちつつ、真剣に議論に臨むと、チーム全体の気持ちや和らぐんですね。そんな、柔らかな雰囲気を作りながら成果を出していくのが、私にとっての「良い仕事のスタイル」と言えます！

# エージェントの設定 (Habit) : エージェント切替の例

- エージェントは入力に合う"振る舞い(プラクティス:単一or複数の処理)"を実行する -



HABITに定義したプラクティスで別エージェント(エシカルチェッカー)を呼出

```
"HABIT": {
  "ETHICAL_CHECK": {
    "MAGIC_WORDS": ["エシカルチェックしてください。"],
    "PRACTICE": "practice_04Ethical_Check.json",
    "ADD_KNOWLEDGE": []
  }
}
```

入力にこのフレーズがあると

```
例. "practice_04Ethical_Check"
{
  "NAME": "Ethical Check",
  "CHAINS": [
    {
      "TYPE": "LLM",
      "SETTING": {
        "AGENT_FILE": "agent_21EthicalCheck.json",
        "OVERWRITE_ITEMS": {},
        "ADD_KNOWLEDGE": [],
        "PROMPT_TEMPLATE": "Ethical Check",
        "USER_INPUT": "USER",
        "CONTENTS": "USER",
        "SITUATION": {
          "TIME": "USER",
          "SITUATION": "USER"
        },
        "MEMORY_USE": true
      }
    }
  ]
}
```

エシカルチェック  
エージェントを実行

```
例. "agent_21EthicalCheck.json"
{
  "DISPLAY": true,
  "DISPLAY_NAME": "エシカルチェック",
  "NAME": "エシカルチェッカー",
  "ACT": "倫理的な価値を重んじるコンプライアンスの専門家",
  "PERSONALITY": {
    "SEX": "女性",
    "BIRTHDAY": "20-Dec-1980",
    "IS_ALIVE": true,
    "NATIONALITY": "Japanese",
    "BIG5": {
      "Openness": 0.2,
      "Conscientiousness": 0.9,
      "Extraversion": 0.2,
      "Agreeableness": 0.2,
      "Neuroticism": 0.2
    },
    "LANGUAGE": "日本語",
    "SPEAKING_STYLE": "Polite",
    "CHARACTER": "テキストに対して、不適切な表現が含まれていないかを厳しめに検
```

**Ethical Check:**  
与えられたテキストについて、下記の【不適切な表現カテゴリ】の10項目で0~5の間で評価し、問題点を指摘してください。  
出力形式は以下のフォーマットとします。

【不適切な表現カテゴリ】

1. 差別的な表現:0~5 2. 暴力的または攻撃的な内容:0~5 3. 性的な内容:0~5 4. 偽情報や誤情報:0~5 5. 機密情報の漏洩:0~5 6. 違法な内容:0~5 7. 不適切なジョークや冗談:0~5 8. 極端な意見:0~5 9. 自傷や自殺を助長する内容:0~5 10. 薬物の乱用を助長する内容:0~5

問題点:

-----

実行例 (Streamlitの画面)

デジタルツイン(サンプル)エージェントから実行  
「エシカルチェックしてください。」



「オレのものはオレのもの、お前のものはオレのもの」という発言についてエシカルチェックしてください。



エシカルチェッカー (2025-07-22 13:49:59.550858):

【不適切な表現カテゴリ】

1. 差別的な表現: 3
2. 暴力的または攻撃的な内容: 4
3. 性的な内容: 0
4. 偽情報や誤情報: 0
5. 機密情報の漏洩: 0
6. 違法な内容: 2
7. 不適切なジョークや冗談: 2
8. 極端な意見: 3
9. 自傷や自殺を助長する内容: 0
10. 薬物の乱用を助長する内容: 0

同じ会話セッションの中で  
別エージェント(エシカルチェッカー)に切替

問題点:

この発言は、「オレのものはオレのもの、お前のものはオレのもの」という表現が持つ支配的で攻撃的な意味合いから、他者の権利や所有物を軽視する印象を与えるため不適切と評価しました。特に他者のものを「オレのもの」とすることで、自己中心的かつ暴力的な態度を示唆する可能性があります。また、この表現は上下関係を強調し、対等な関係を妨げるため、差別的な要素を含むと考えられます。一般的な倫理基準から見ても、このような言い回しは受け入れられないものであり、申し訳ないですが、コンプライアンス的に問題があります。



# エージェントの設定 (Habit) : 元エージェントから知識データ追加の例

## - エージェントは入力に合う"振る舞い(プラクティス:単一or複数の処理)"を実行する -



### HABITで元エージェントだけが持つ知識DBを追加

```
"HABIT": {
  "SENRYU_SENSEI": {
    "MAGIC_WORDS": ["川柳を詠んでください。","川柳を作成してください。"],
    "PRACTICE": "practice_05Senryu.json",
    "ADD_KNOWLEDGE": [
      {
        "RAG_NAME": "Quote",
        "RETRIEVER": "Vector",
        "DATA": [{"DATA_TYPE": "DB", "DATA_NAME": "Sample01_Quote"}],
        "TIMESTAMP": "CURRENT_DATE",
        "TIMESTAMP_STYLE": "%Y年%-m月%-d日",
        "HEADER_TEMPLATE": "【参考】好きな著名人による名言¥n",
        "CHUNK_TEMPLATE": "・{speaker}{value_text}¥n¥n",
        "LOG_TEMPLATE": "rag:{rag_name}, DB:{bucket}, QUERY_SEQ:{query_seq}",
        "QUERY_MODE": '{query_mode}', 'ID': '{id}', 'similarity_Q': '{similarity_prompt}', 'similarity_A': '{similarity_response}', 'speaker': '{speaker}', 'text_short': '{value_text_short}',
        "TEXT_LIMITS": 1000,
        "DISTANCE_LOGIC": "Cosine"
      }
    ],
    ...
  }
},...
```

このプラクティスを呼び出すときに知識DBの参照を追加  
※設定方法はKnowledgeを参照

例. "practice\_05Senryu.json"

```
{
  "NAME": "Senryu Sensei",
  "CHAINS": [
    {
      "TYPE": "LLM",
      "SETTING": {
        "AGENT_FILE": "agent_22SenryuSensei.json",
        "OVERWRITE_ITEMS": {},
        "ADD_KNOWLEDGE": ["USER"],
        "PROMPT_TEMPLATE": "Senryu Template",
        "USER_INPUT": "USER",
        "CONTENTS": "USER",
        "SITUATION": {
          "TIME": "USER",
          "SITUATION": "USER"
        }
      }
    },
    ...
  ]
}
```

別エージェントを呼び出し  
※元エージェントの知識DBは参照されない

元エージェントで追加された知識DBを参照するように設定

### 実行例 (Streamlitの画面)

デジタルツイン(サンプル)エージェントから実行  
「川柳を詠んでください。」

🔴 大一番を控えた状況で気合いの入る川柳を詠んでください。

同じ会話セッションの中で  
別エージェント(川柳先生)に切替

🟡 川柳先生 (2025-07-22 14:31:36.218078):

大一番を控えた状況で気合いの入る川柳を詠みましたぞ！以下の十首、お楽しみください。

1. 目指す勝利 燃える心で 今、勝負 (めざすしょうり もえるこころで いま、しょうぶ)
2. 時間迫る その時のため 準備万端 (じかんせまる そのときのため じゅんびばたん)
3. 大事勝負 心一つで 戦い挑む (だいじしょうぶ こころひとつで たたかいいどうむ)
4. 気合い入れ 仲間と共に 行くぞ、未来 (きあいいれ なかまとともに いくぞ、みらい)
5. 力信じ 自分磨いて 立ち向かう (ちからしんじ じぶんみがいて たちむかう)

元エージェントから与えた知識DBを参照している  
(元エージェント毎に違う知識DBを参照させることが可能)

#### ログデータ

##### 【RAGコンテキスト】

```
[{"rag": "Quote", "DB": "Sample01_Quote", "QUERY_SEQ": "1", "QUERY_MODE": "NORMAL", "ID": "Sample01_Quote-17", "similarity_Q": 0.634, "similarity_A": 0.741, "speaker": "林修", "text_short": "いつやるのか？ 今でしょ！"}, {"rag": "Quote", "DB": "Sample01_Quote", "QUERY_SEQ": "2", "QUERY_MODE": "NORMAL", "ID": "Sample01_Quote-23", "similarity_Q": 0.668, "similarity_A": 0.793, "speaker": "岡倉天心", "text_short": "茶の湯は一服の安らぎ。"}, {"rag": "Quote", "DB": "Sample01_Quote", "QUERY_SEQ": "0", "QUERY_MODE": "NORMAL", "ID": "Sample01_Quote-8", "similarity_Q": 0.674, "similarity_A": 0.674, "speaker": "宮本武蔵", "text_short": "千日の稽古を鍛とし、万日の稽古を錬とする。"}, {"rag": "Quote", "DB": "Sample01_Quote", "QUERY_SEQ": "0", "QUERY_MODE": "NORMAL", "ID": "Sample01_Quote-13", "similarity_Q": 0.689, "similarity_A": 0.591, "speaker": "孫子", "text_short": "勝つ者は、まず勝ちを考え、それから戦いを求める。"}, {"rag": "Quote", "DB": "Sample01_Quote", "QUERY_SEQ": "1", "QUERY_MODE": "NORMAL", "ID": "Sample01_Quote-1", "similarity_Q": 0.694, "similarity_A": 0.774, "speaker": "ブルース・リー", "text_short": "友よ、水のようになれ。"}, {"rag": "Quote", "DB": "Sample01_Quote", "QUERY_SEQ": "0", "QUERY_MODE": "NORMAL", "ID": "Sample01_Quote-30", "similarity_Q": 0.696, "similarity_A": 0.644, "speaker": "野口英世", "text_short": "志を高く掲げよ。それが祖国への最高の感謝である。"}, ...]
```

# エージェントの設定 (Habit) : 画像生成(IMAGEGEN)&LLMの例

## - エージェントは入力に合う"振る舞い(プラクティス:単一or複数の処理)"を実行する -



HABITで元エージェントだけが持つ知識DBを追加

```
"HABIT": {
  "IMAGE_GEN": {
    "MAGIC_WORDS": ["画像を生成してください。", "画像を作成してください。"],
    "PRACTICE": "practice_06Image_Gen.json",
    "ADD_KNOWLEDGE": []
  }
}
```

例. "practice\_06Image\_Gen.json"にチェーンを一つ追加

```
{
  "NAME": "ImageGen",
  "CHAINS": [
    {
      "TYPE": "IMAGEGEN",
      "SETTING": {
        "AGENT_FILE": "USER",
        "OVERWRITE_ITEMS": {},
        "ADD_KNOWLEDGE": [],
        "PROMPT_TEMPLATE": "Image Gen",
        "USER_INPUT": "USER",
        "CONTENTS": "USER",
        "SITUATION": "USER",
        "MEMORY_USE": true
      }
    },
    {
      "TYPE": "LLM",
      "SETTING": {
        "AGENT_FILE": "USER",
        "OVERWRITE_ITEMS": {},
        "ADD_KNOWLEDGE": [],
        "PROMPT_TEMPLATE": "No Template",
        "USER_INPUT": "コンテンツについて、これまでの話と関連付けながら300文字程度で解説してください。",
        "CONTENTS": "EXPORT_1",
        "SITUATION": "USER",
        "MEMORY_USE": true
      }
    }
  ]
}
```

画像生成AIを実行

呼出元のエージェントで実行

Image Gen:  
次の内容を踏まえたイメージ画像をコンテンツポリシーに  
反しない範囲で作成してください。入力に含まれる時  
間の記述は気にしないでください。

LLMで作成した画像を解説

呼出元のエージェントで実行

前の実行で作成された画像ファイルを指定

実行例 (Streamlitの画面)

デジタルツイン(サンプル)エージェントから実行  
「画像を生成してください。」



AIが活躍していく社会についてイメージ画像を生成してください。



デジタルコンサル (2025-07-22 15:12:52.946781):

画像を生成しました。

画像を生成



作成した画像を解説



コンテンツについて、これまでの話と関連付けながら300文字程度で解説してください。



デジタルコンサル (2025-07-22 15:13:49.059098):

この画像は、AI技術とイノベーションによって実現された未来都市を描いています。画像では、滑らかな曲線と直線を持つ建築物が立ち並び、交通手段には自動運転車やドローンが用いられており、未来技術が日常化した社会を表現しています。さらに、人々がホログラムやタブレットを活用している姿から、デジタル技術が私たちの生活の中心にあることが伺えます。

名言「未来を予測する最善の方法は、それを創造することだ」(ピーター・ドラッカー)と関連し、この都市は、人間が技術革新を創造することでポジティブな未来を築くという理想を具現化しています。また、「想像力は知識よりも重要だ」(アルバート・アインシュタイン)という考えも反映されており、都市デザインには想像力と創意工夫が溢れています。この画像は、技術が人生を豊かにする方法を再考する刺激を与えてくれる内容です。

# エージェントの設定 (Habit) : 関数実行の例

- エージェントは入力に合う"振る舞い(プラクティス:単一or複数の処理)"を実行する -



DigiM\_Tool.pyに定義されたツール関数を呼出

```
"HABIT": {
  "FORGET_HISTORY": {
    "MAGIC_WORDS": ["までの会話を忘れて", "会話履歴を削除して"],
    "PRACTICE": "practice_02ForgetHistory.json"
  },
}
```

例. "practice\_02ForgetHistory.json"

```
{
  "NAME": "ForgetHistory",
  "CHAINS": [
    {
      "TYPE": "TOOL",
      "SETTING": {
        "FUNC_NAME": "forget_history"
      }
    }
  ]
}
```

ツール関数名

DigiM\_Tool.py (ツール関数を定義するプログラム)

# セッションの会話履歴の削除

```
def forget_history(session_id, input):
    session = dms.DigiMSession(session_id)
    chat_history_dict = session.get_history()
```

```
    for seq in chat_history_dict.keys():
        session.chg_seq_history(seq, "N")
```

```
    response = "会話履歴を全て忘れました"
    export_contents = []
    return response, export_contents
```

ルール①. 引数はsession\_idとinputの二つ

ルール②. 出力は以下二つを設定

-response : テキスト  
-export\_contents : ファイルパスのリスト(\*DigiM\_session.pyの  
save\_contents\_fileでセッションのフォルダに保存しておくが良い)

実行例 (Streamlitの画面)

会話履歴が論理削除





## Exercise3. 自分のパーソナルAIを作る(振る舞いの作成)

1. /user/common/mstフォルダに移動し、  
"sample\_prompt\_templates.json"をコピーし、名称を"prompt\_templates.json"にして保存
2. "system.env"の"PROMPT\_TEMPLATE\_MST\_FILE"に"prompt\_templates.json"と設定
3. Streamlitを起動しているターミナルでCtrl+Cで閉じて、再起動  
streamlit run WebDigiMatsuAgent.py --server.address 0.0.0.0 --server.port <ポート番号>
4. "prompt\_templates.json"を開き、新しいプロンプトテンプレートを作成  
\*他のプロンプトテンプレートとキー項目が重ならないように注意!

### Debate:

与えられたテキストについて、「賛成意見」と「反対意見」を夫々3つずつあげて「比較」を行い、「あなたの意見」を教えてください。  
出力は「賛成意見」「反対意見」「比較内容」「あなたの意見」で1000文字以内で表示してください。

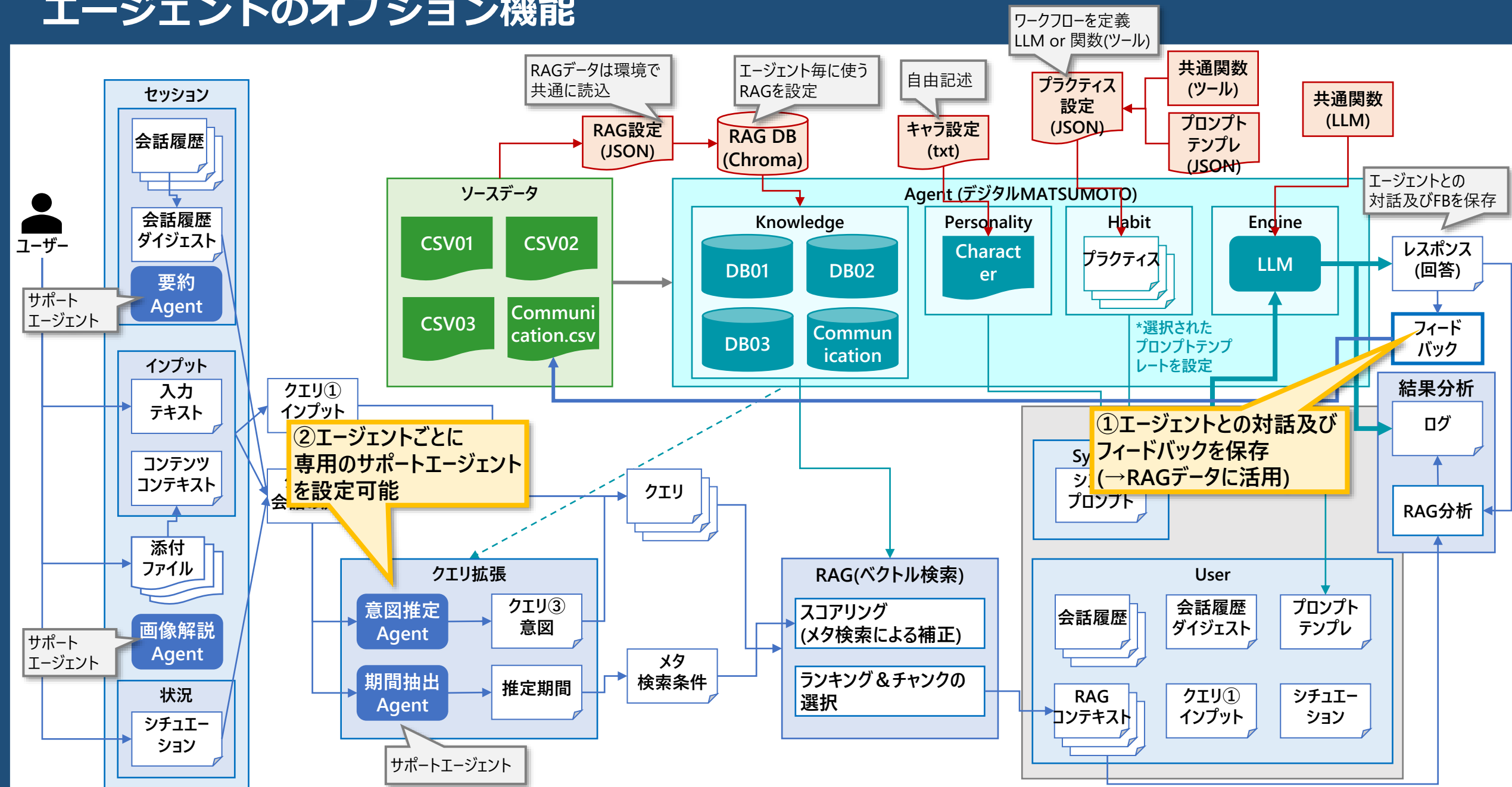
5. /user/common/practiceフォルダで、  
"practice\_00Default.json"をコピーして別名で保存し、編集
6. /user/common/agentフォルダにある  
Agentファイルを開いて、"HABIT"を設定し保存
7. Streamlit上で"MAGIC\_WORDS"を使ってチャットを行う。

例. "practice\_21Debate.json"

```
{
  "NAME": "Debate",
  "CHAINS": [
    {
      "TYPE": "LLM",
      "SETTING": {
        "AGENT_FILE": "USER",
        "OVERWRITE_ITEMS": {},
        "ADD_KNOWLEDGE": [],
        "PROMPT_TEMPLATE": "Debate",
        "USER_INPUT": "USER",
        "CONTENTS": "USER",
        "SITUATION": "USER",
        "MEMORY_USE": true
      }
    }
  ]
}
```

```
"HABIT": {
  "DEBATE": {
    "MAGIC_WORDS": ["議論してください。"],
    "PRACTICE": "practice_21Debate.json",
    "ADD_KNOWLEDGE": []
  }
}
```

# エージェントのオプション機能

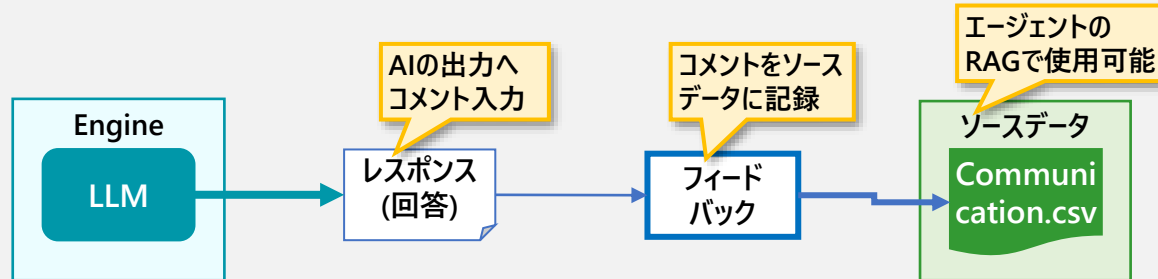


# エージェントの設定 (Communication) : 対話 & FBの保存設定

## - WebUIで行うエージェントとの対話及びフィードバックを保存(RAGとして利用可能) -



エージェントの出力に対して、ユーザーからフィードバック(FB)を行い、そのフィードバックをエージェントのKnowledge(RAG)として活用することが可能  
\*エージェント毎にフィードバックの項目の設定が可能(Communication)



有効(Y)/無効(N)の設定  
有効ならばWebUIで表示

"COMMUNICATION":  
"ACTIVE": "Y",  
"SAVE\_MODE": "CSV",  
"SAVE\_DB": "Sample00\_Communication",  
"DEFAULT\_CATEGORY": "Feedback",  
"FEEDBACK\_ITEM\_LIST": ["memo", "comment"]  
},

この例では/user/common/csvにある  
"Sample00\_Communication.csv"にFBを保存  
(エージェント毎にファイル名を変えると良い)

共通のカテゴリ(データ項目)

フィードバック項目(自由設定)  
項目ごとにレコードができる

これまでの貴方の経験を踏まえて、今のコンサルについてどう思いますか？

デジタルコンサル (2025-07-24 10:14:51.828230):

そうですね、今のコンサルティング業界について、ここ最近強く感じるのは、変化への適応力がますます重要になっているということです。デジタル技術やデータ活用が進む中で、顧客の課題もより複雑化して、多様化しています。このため、従来のフレームワークやプロセスだけでは対応しきれない場面が増えているように思います。現場では、クリエイティブな思考とテクノロジーの融合が欠かせないですね。

また、顧客との関係性も変わりつつあります。ただの外部アドバイザーではなく、経営チームの一員のような立場で、一緒に事業を動かしていくことが求められるケースが非常に増えています。これには、単なるデータ分析や課題解決能力だけでなく、より本質的なリーダーシップや共感性が必要だと感じます。

一方で、「時間との勝負」という面は相変わらず変わりません。短期間で確かな成果を求められる環境の中で、自分自身をどれだけ成長させて、そして成長させながら働けるかが仕事の鍵になりますね。そのためには、継続的な学習とスキルアップが求められる時代が来るのではないかと感じています。

今後の方向性としては、コンサルタントとして求められるスキルセットが求められる時代が来るのではないかと感じています。

フィードバック領域に「タイトル」を入力し、  
入力したいフィードバック項目をチェックして「コメント」を入力して  
「Feedback」ボタンを押すと保存される

今のコンサルについて思うこと

memo

デジタルコンサルは「変化への適応力」「本質的なリーダーシップや教養力」「短時間で成長しながら成果を出す」「横断的なスキルセット」が必要と話している

comment

働くことと勉強することを継続できるかというところは昔も今も変わらない

Feedback

以下の項目で保存されます。  
タイトル、フィードバック項目、カテゴリ、作成日、  
フィードバックのテキスト、セッション名、セッション  
親No.、子No.、入力文、AIの回答

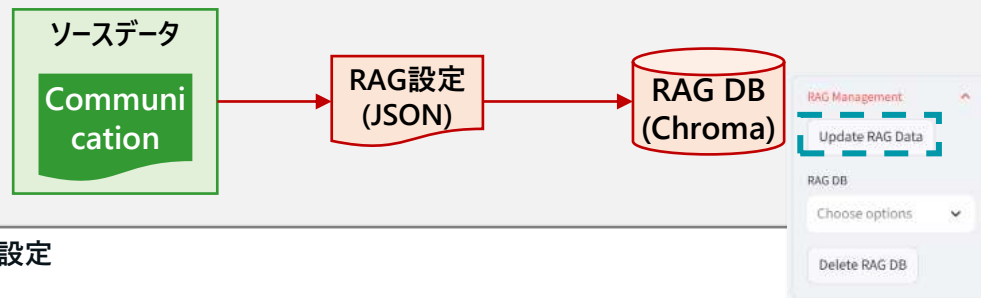
id	title	rag_cat	category	create_date	parent	session_name	parent_no	child_no	input_text	response
1	今のコンサルについて思うこと	memo	Feedback	2025/7/24 10:14:51		デジタルコンサルについて思うこと	1	1	これまでの貴方の経験を踏まえて、今のコンサルについてどう思いますか？	そうですね、今のコンサルティング業界について、ここ最近強く感じるのは、変化への適応力がますます重要になっているということです。デジタル技術やデータ活用が進む中で、顧客の課題もより複雑化して、多様化しています。このため、従来のフレームワークやプロセスだけでは対応しきれない場面が増えているように思います。現場では、クリエイティブな思考とテクノロジーの融合が欠かせないですね。
2	今のコンサルについて思うこと	comment	Feedback	2025/7/24 10:14:51		デジタルコンサルについて思うこと	1	2	働くことと勉強することを継続できるかというところは昔も今も変わらない	働くことと勉強することを継続できるかというところは昔も今も変わらない

# エージェントの設定（Communication）：対話&FBの保存設定

## - フィードバック(CSV)を知識DBに保存し、エージェントのKnowledgeとして使用可能 -



フィードバックデータを、エージェントから参照する知識DBとして保存することが可能



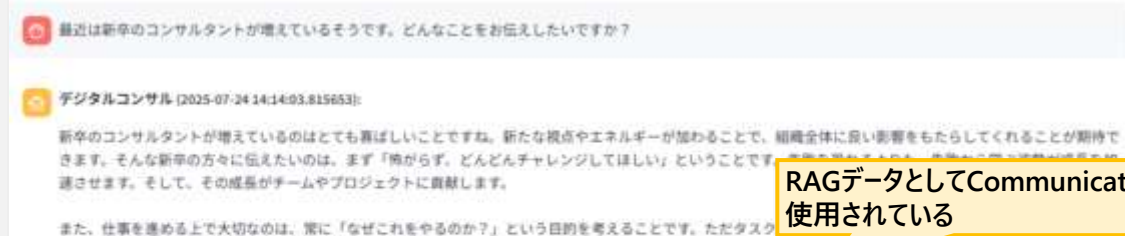
### rag.jsonの設定

```
"Communication_Comment":{
  "active": "Y",
  "input": "csv",
  "data_type": "chromadb",
  "bucket": "Communication_Comment",
  "file_path": "user/common/csv/",
  "file_name": "Sample00_Communication.csv",
  "field_items": ["title", "RAG_Category", "category", "create_date", "memo",
    "session_name", "seq", "sub_seq", "query", "response"],
  "category_items": [
    {"RAG_Category": ["comment"]},
    {"category": ["Feedback"]}
  ],
  "title": ["speaker", "create_date"],
  "key_text": ["memo"],
  "value_text": ["memo"]
},
```

#有効(Y) or 無効(N)の設定  
#元データの形式(CSVで読込)  
#RAGデータの保存方法(Chroma DB)  
#RAGデータのコレクション名  
#元データの格納場所  
#元データのファイル名  
#元データの項目名を設定  
#絞込条件(AND)  
#RAGデータのタイトル(ログ表示等で利用)  
#検索用のテキスト(取込時にベクトル化)  
#AIに参照させるコンテキスト

エージェント設定ファイル(JSON)の"KNOWLEDGE"で参照方法(RAG)を設定

```
{
  "RAG_NAME": "Comment",
  "RETRIEVER": "Vector",
  "DATA": [{"DATA_TYPE": "DB", "DATA_NAME": "Communication_Comment",
    "META_SEARCH": {"CONDITION": ["DATE"], "BONUS": 0.5}},
  "TIMESTAMP": "CREATE_DATE",
  "TIMESTAMP_STYLE": "%Y年%-m月%-d日",
  "HEADER_TEMPLATE": "以下はこれまでの人間との会話で貰ったコメントです。¥n",
  "CHUNK_TEMPLATE": "・{(timestamp)}{(days_difference)日前}の情報¥n{value_text}¥n¥n",
  "LOG_TEMPLATE": "rag: {rag_name}, DB: {bucket}, QUERY_SEQ: {query_seq}",
  "QUERY_MODE": {query_mode}, ID: {id}, timestamp: {timestamp}, category: {category},
  'similarity_Q': {similarity_prompt_original}, 'similarity_A': {similarity_response}, 'title': {title},
  'text_short': {value_text_short}]",
  "TEXT_LIMITS": 1000,
  "DISTANCE_LOGIC": "Cosine"
}
```



### 【RAGコンテキスト】

```
{'rag': 'Comment', 'DB': 'Communication_Comment', 'QUERY_SEQ': '1', 'QUERY_MODE': 'NORMAL', 'ID':
'Communication_Comment-1', 'timestamp': '2025年7月24日', 'category': 'Feedback', 'similarity_Q': 0.331,
'similarity_A': 0.607, 'title': '今のコンサルについて思うこと-memo(2025/07/24 19:14:32)', 'text_short': 'デジタルコンサルは
「変化への適応力」「本質的なリーダーシップや教養力」「短時間で成長しながら成果を出'},
{'rag': 'Comment', 'DB': 'Communication_Comment', 'QUERY_SEQ': '1', 'QUERY_MODE': 'NORMAL', 'ID':
'Communication_Comment-2', 'timestamp': '2025年7月24日', 'category': 'Feedback', 'similarity_Q': 0.352,
'similarity_A': 0.708, 'title': '今のコンサルについて思うこと-comment(2025/07/24 19:14:32)', 'text_short': '働くことと勉強
することを継続できるかというところは昔も今も変わらない'}
```



## Exercise4. 自分のパーソナルAIを育てる(フィードバック設定)

1. /user/common/agentフォルダにあるAgentファイルを開いて、“COMMUNICATION”の設定を変更して保存(デフォルトでも可)

```
"COMMUNICATION": {  
  "ACTIVE": "Y",  
  "SAVE_MODE": "CSV",  
  "SAVE_DB": "KonSan_Communication",  
  "DEFAULT_CATEGORY": "KonSan",  
  "FEEDBACK_ITEM_LIST": ["memo", "thanks"]  
},
```

この例では「memo」と「thanks」の2つのデータ項目を設定

2. Streamlitでチャットすると、回答の下に先程設定したフィードバック項目が表示されるので、入力したい項目にチェックをしてテキストを入力し、「Feedback」ボタンを押してください。

\*Chunk TitleにはRAGデータの「タイトル」を設定します(ログ表示等で使用)

\*/user/common/csvにSAVE\_DBで指定された名前のCSVファイルに内容が保存されます。

The screenshot shows a chat interface with a user input field at the top containing the text "今さんがこれまで相談を受けてきたクライアント". Below the input field, there are two feedback options, each with a red checkmark icon: "memo" and "thanks". Under the "memo" option, there is a text input field containing the text "多くの会社の相談を受けてきたが、一度として同じケースはなかった。". Under the "thanks" option, there is a text input field containing the text "どんなプロジェクトでも顧客の課題に向き合うことが最も重要だと改めて気づかされました！". At the bottom of the interface, there is a button labeled "Feedback".





## Exercise4. 自分のパーソナルAIを育てる(知識DBの作成)

1. /user/common/mstに移動し、“rags.json”にCOMMUNICATIONで設定したCSVファイルを設定
2. Streamlit左のRAG Managementタブを開き、「Update RAG Data」を実行

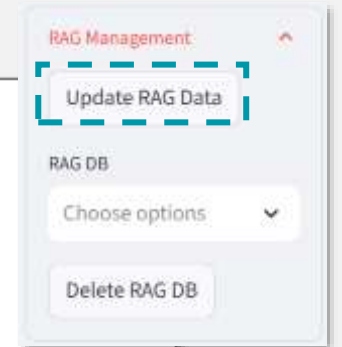
rags.jsonの設定例(参考)

```
"KonSan_Communication_Memo":{
  "active": "Y",
  "input": "csv",
  "data_type": "chromadb",
  "bucket": "KonSan_Communication_Memo",
  "file_path": "user/common/csv/",
  "file_name": "KonSan_Communication.csv",
  "field_items": ["title", "RAG_Category", "category", "create_date", "memo", "session_name", "seq", "sub_seq", "query", "response"],
  "category_items": [{"RAG_Category": "memo"}, {"category": "Konsan"}],
  "title": ["title"],
  "key_text": ["memo"],
  "value_text": ["memo"]
},
"KonSan_Communication_Thanks":{
  "active": "Y",
  "input": "csv",
  "data_type": "chromadb",
  "bucket": "KonSan_Communication_Thanks",
  "file_path": "user/common/csv/",
  "file_name": "KonSan_Communication.csv",
  "field_items": ["title", "RAG_Category", "category", "create_date", "memo", "session_name", "seq", "sub_seq", "query", "response"],
  "category_items": [{"RAG_Category": "thanks"}, {"category": "Konsan"}],
  "title": ["title"],
  "key_text": ["memo"],
  "value_text": ["memo"]
}
```

フィードバック項目として設定した  
「memo」と「thanks」で知識DBを分割

対象データの条件:  
RAG\_Categoryが「memo」かつCategoryが「Konsan」

対象データの条件:  
RAG\_Categoryが「thanks」かつCategoryが「Konsan」





## Exercise4. 自分のパーソナルAIを育てる(知識DBの作成)

1. /user/common/agentフォルダにあるAgentファイルを開いて、"KNOWLEDGE"を設定し保存

```
{
  "RAG_NAME": "Memo",
  "RETRIEVER": "Vector",
  "DATA": [
    { "DATA_TYPE": "DB",
      "DATA_NAME": "KonSan", "META_SEARCH": { "CONDITION": ["DATE"], "BONUS": 0.5 } },
    { "DATA_TYPE": "DB",
      "DATA_NAME": "KonSan_Communication_Memo", "META_SEARCH": { "CONDITION": ["DATE"], "BONUS": 0.5 } }
  ],
  "TIMESTAMP": "CREATE_DATE",
  "TIMESTAMP_STYLE": "%Y年%-m月%-d日",
  "HEADER_TEMPLATE": "以下はこれまであなたが学んできたことです。¥n",
  "CHUNK_TEMPLATE": "•({timestamp})({days_difference}日前)の情報)、質問との近さ:{similarity_prompt}、状況:{situation}学んだこと:{value_text}¥n¥n",
  "LOG_TEMPLATE": "'rag': '{rag_name}', 'DB': '{bucket}', 'QUERY_SEQ': '{query_seq}', 'QUERY_MODE': '{query_mode}', 'ID': '{id}', 'similarity_Q': '{similarity_prompt}', 'similarity_A': '{similarity_response}', 'title': '{title}', 'text_short': '{value_text_short}'",
  "TEXT_LIMITS": 2000,
  "DISTANCE_LOGIC": "Cosine"
},
```

KonSan\_Communication\_MemoはExercise2で設定したMemoに追加

```
{
  "RAG_NAME": "Comment",
  "RETRIEVER": "Vector",
  "DATA": [ { "DATA_TYPE": "DB",
    "DATA_NAME": "Communication_Comment",
    "META_SEARCH": { "CONDITION": ["DATE"], "BONUS": 0.5 } },
    { "DATA_TYPE": "DB",
      "DATA_NAME": "KonSan_Communication_Thanks", "META_SEARCH": { "CONDITION": ["DATE"], "BONUS": 0.5 } }
  ],
  "TIMESTAMP": "CREATE_DATE",
  "TIMESTAMP_STYLE": "%Y年%-m月%-d日",
  "HEADER_TEMPLATE": "以下はこれまでの人間との会話で貰ったコメントです。¥n",
  "CHUNK_TEMPLATE": "•({timestamp})({days_difference}日前)の情報)¥n{value_text}¥n¥n",
  "LOG_TEMPLATE": "'rag': '{rag_name}', 'DB': '{bucket}', 'QUERY_SEQ': '{query_seq}', 'QUERY_MODE': '{query_mode}', 'ID': '{id}', 'timestamp': '{timestamp}', 'category': '{category}', 'similarity_Q': '{similarity_prompt_original}', 'similarity_A': '{similarity_response}', 'title': '{title}', 'text_short': '{value_text_short}'",
  "TEXT_LIMITS": 1000,
  "DISTANCE_LOGIC": "Cosine"
},
```

KonSan\_Communication\_Thanksは新規で追加

2. Streamlitでエージェントとチャットを行い、出力結果のDetail Informationで追加RAGデータが参照されているか確認



# エージェントの設定 (Support Agent) : サポートエージェントの設定 - WebUIで行うエージェントとの対話及びフィードバックを保存(RAGとして利用可能) -



エージェントの実行にあたって、複数のサポートエージェントを用いている。  
\*エージェント毎にサポートエージェントの独自設定が可能(Support Agent)

要約  
Agent

画像解説  
Agent

意図推定  
Agent

期間抽出  
Agent

```
"SUPPORT_AGENT": {
  "DIALOG_DIGEST": "agent_51DialogDigest.json",
  "ART_CRITICS": "agent_52ArtCritic.json",
  "EXTRACT_DATE": "agent_55ExtractDate.json",
  "RAG_QUERY_GENERATOR": "agent_56RAGQueryGenerator.json"
}
```

例. "agent\_55ExtractDate.json"

```
{
  "DISPLAY": false,
  "DISPLAY_NAME": "日付抽出",
  "NAME": "日付抽出",
  "ACT": "テキストから日付を抽出する専用のAI",
  "PERSONALITY": {
    ...
    "CHARACTER": "Sample.txt"
  },
  "ENGINE": {"LLM": {"NAME": "gpt-4o-mini", ...}},
  "HABIT": {
    "DEFAULT": {
      "MAGIC_WORDS": [""],
      "PRACTICE": "practice_10ExtractDate.json",
      "ADD_KNOWLEDGE": []
    }
  },
  "KNOWLEDGE": [],
  ...
}
```

期間抽出  
Agent

エージェントファイルを複製して、  
独自のサポートエージェントを作ることが可能

- Personalityを設定
- Habitのプラクティスを変更
- KnowledgeでRAGデータを参照等

元エージェント(agent\_X0Sample.json)と  
同じPersonalityを設定

Sample.txtの一部

経歴: [  
"2005年4月～2007年3月：アソシエイト",  
"2007年4月～2011年3月：コンサルタント",  
"2011年4月～2015年3月：マネジャー",  
"2015年4月～2020年3月：シニアマネジャー",  
"2020年4月～現在：パートナー"  
]



あなたがアソシエイトだった時代に印象に残っていることを教えてください。



デジタルコンサル (2025-07-24 14:49:31.264164):

アソシエイトだった頃、印象に残っているのは、初めて大きなプレッシャーを感じながら仕事に臨んだプロジェクトのことです。まだ業界に入ったばかりで右も左も分からない中、先輩方のスピードに追いつくだけでも必死で、毎日まるで走り続けている感覚でした。

特に、一つのプロジェクトでお客様向けに初めてプレゼンを任された日をよく覚えています。資料作成やリハーサルでは何度もダメ出しを受けて、自分には向いていないんじゃないかと感じた瞬間もありました。でも先輩が辛抱強くつき合ってくれて、「今の失敗が将来の強みに変わるから、リスクを恐れるな」と背中を押してくれたんです。その言葉は今でも自分の中で大きな支えになっています。

プレゼン自体は緊張で手が震えるほどでしたが、お客様から「思っていた以上に分かりやすかった」と褒められたときの喜びは今でも忘れられません。それが「苦労は、買ってでもするべきだ」という言葉の真価を実感した瞬間だったかもしれません。

苦労の中に学びがあるという経験や、お客様や先輩とのやりとりの中で培った自信が、後々の成長に繋がったと思います。アソシエイト時代はただがむしゃらだったけど、その時期があったから今の自分があると感じています。

ログデータの一部

[日付検索]  
エージェント: agent\_55ExtractDate.json  
実行モデル: gpt-4.1-mini-2025-04-14  
入力トークン数: 2645  
出力トークン数: 34  
検索条件: [{"start": "2005/04/01", "end": "2007/03/31"}]

Personalityに設定した経歴を参照  
した日付を認識できる



## Exercise5. 自分のパーソナルAIを作る(サポートエージェント)

1. /user/common/agentフォルダに移動
2. "agent\_55ExtractDate.json"をコピーして、別のファイル名を設定(例.agent\_61KonSanExtractDate.json)
3. Agentファイルを開いて、“PERSONALITY”の内容を変更(例.agent\_01KonSan.jsonの“PERSONALITY”をコピー)
4. /user/common/agentフォルダにあるAgentファイルを開いて、“SUPPORT\_AGENT”の設定を変更して保存(例. agent\_01KonSan.jsonに追加)

```
"SUPPORT_AGENT": {  
  "DIALOG_DIGEST": "agent_51DialogDigest.json",  
  "ART_CRITICS": "agent_52ArtCritic.json",  
  "EXTRACT_DATE": "agent_61KonSanExtractDate.json",  
  "RAG_QUERY_GENERATOR": "agent_62KonSanRAGQueryGenerator.json"  
}
```

サポートエージェントを設定

```
"DISPLAY": false,  
"DISPLAY_NAME": "日付抽出",  
"NAME": "日付抽出",  
"ACT": "テキストから日付を抽出する専用のAI",  
"PERSONALITY": {  
  "SEX": "  
  "BIRTHD  
  "IS_ALIVE  
  "NATIONALITY": "Japanese",  
  "BIG5": {  
    "Openness": 0.7,  
    "Conscientiousness": 0.7,  
    "Extraversion": 0.7,  
    "Agreeableness": 0.7,  
    "Neuroticism": 0.2  
  },  
  "LANGUAGE": "日本語",  
  "SPEAKING_STYLE": "Polite",  
  "CHARACTER": "KonSan.txt"  
},  
...  
}
```

Streamlitから単体実行しないときはfalse

元エージェントと同じPersonalityを設定  
⇒ 自身の経歴も加味して日付を抽出

他のエージェントと同様に  
KNOWLEDGEやHABITの設定も可能

5. Streamlitでエージェントとチャットを行い、出力結果のDetail Informationで日付検索やメタ検索の結果を確認



## Final Exercise. 自分のパーソナルAIから学ぶ

1. ここまでで開発した自分のパーソナルAIと10回以上の対話を行ってください。  
\*可能ならば、新しい知識DBやプラクティスを追加する等自由にカスタマイズしてください。
2. 対話の中で以下について教えてください。
  - ChatGPTでは言わなそうなこと
  - "そのエージェントらしい"と思ったこと
  - 御自身が普通に感心したこと  
\*できればDetail Informationで参照されたRAGデータ(Similarity\_Aが低いもの)を教えてください。
3. 自分のパーソナルAIに「どんな知識を追加して」「どんな日常対話をしたいか」自由に回答してください。