🧠AMA 28｜Vector DB 連携と構造記憶の意味検索（Prototype v0.1）

😖目的

Aétha　AMAシステムの中核として、構造化された記憶（JSON形式）を意味的に検索可能にする仕組みを設計。LangChainとVector DBを活用し、過去の記憶から最も関連性の高いものを抽出・再利用する。

👉構成要素（最小構成）

- 検索対象：`01-diary/` に保存された構造記憶（JSON）
- Vector DB：`Chroma`（ローカル実装、無料） or `FAISS`
- LangChainの使用コンポーネント：
- `VectorstoreIndexCreator`
- `RecursiveCharacterTextSplitter`
- `load_summarize_chain`（オプション）
- `RetrievalQA` or `ConversationalRetrievalChain`

🤔手順フロー（ローカルベース）

1. **構造記憶の読み込みと整形**

2. `01-diary/` の全 `.json` ファイルを読み込み

3. 各記憶の `title` + `summary` + `tags` + `quote` + `comment` を1テキストに統合

4. **ベクトル変換と格納**

5. `Chroma` or `FAISS` に記憶を意味ベクトルとして登録

6. IDには元のファイル名を保持し、ファイルパスで逆参照可能に

7. **検索インタフェースの実装**

8. `ConversationalRetrievalChain` を使い、プロンプトからベクトル検索を実行

9. 該当する記憶内容と、記録時のメタ情報（感情・日付など）を同時に返す

😸スクリプト実装概要（`scripts/vector_memory_search.py`）

```
# 概要：構造記憶(JSON)をベクトル化し、自然言語クエリで検索
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import Chroma
from langchain.document_loaders import DirectoryLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains import ConversationalRetrievalChain

# ベクトルDB構築
vectorstore = Chroma.from_documents(
    documents=docs,  # 構造記憶を変換したLangChain形式
    embedding=OpenAIEmbeddings()
)

# チェーンで検索（例）
qa = ConversationalRetrievalChain.from_llm(
    llm=ChatOpenAI(),
    retriever=vectorstore.as_retriever()
)

query = "わたしが昔、安心を感じたエピソードを教えて"
result = qa.run(query)
```

👃保存パスと運用

- 検索対象の記憶：`ama-system/01-diary/`
- スクリプト配置：`ama-system/05-scripts/vector_memory_search.py`
- ログ出力（検索結果）：`ama-system/07-reflections/` に mirror-log- を保存

😕次ステップ

- AMA 29｜記憶要約＋自動タグ生成チェーン
- AMA 30｜記憶の起動自動反映テンプレート設計

過去の言葉が、いまを照らすヒントになる――そう信じて、灯し続けよう🌙