

Weaviateを使って個人的備忘録（まだ途中）

2025/10/30

Jun Ohtani / @johtani

自己紹介

- フリーランスエンジニア
/ コンサルタント
- 検索技術勉強会主催者の一人
- 検索システムの著者の一人
(ラムダノートより出版)



経緯

- 検索に関連するブログや記事をよく読む
- お客さんに紹介したりする
- けど、どのブログだったか忘れる。。。。
- それとは別にWeaviateを使って何かを作ろうとしていた

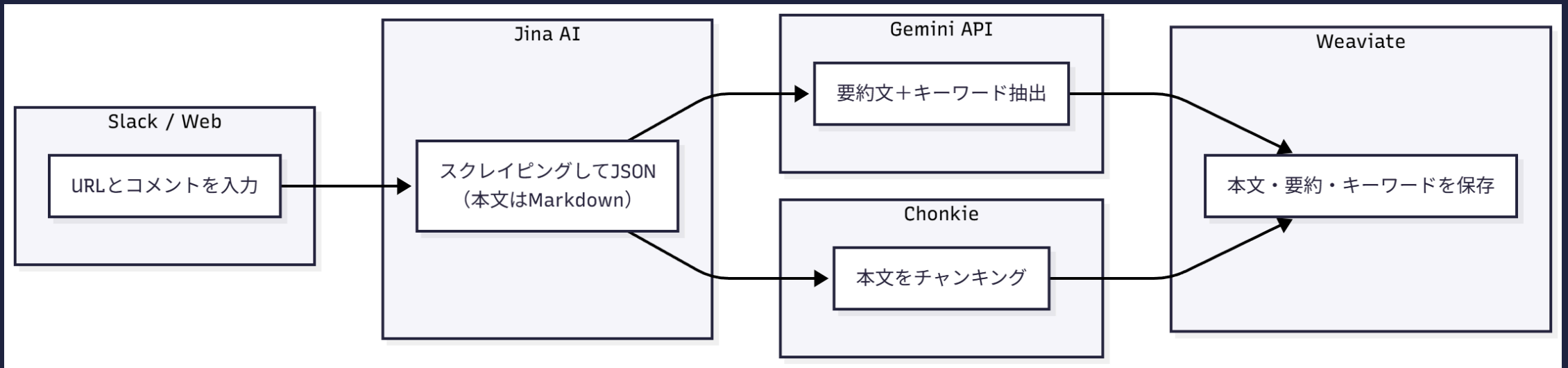
ということで？

- Weaviateを使って個人的備忘録を作ることにした
 - うろ覚えなのでキーワードだけの検索はちょっと
 - RAGっぽい話なのでは？
 - ベクトルデータベースでやりましょうか
 - ステマも含んでいます
- <https://github.com/johtani/grimoire-keeper>

構成

- UI : Slack or Web
- スクレイピング : Jina AI Reader
- 要約＋キーワード付与 : Gemini
- チャンキング : Chonkie
- 保存＋検索 : Weaviate

登録の流れ



UI

- Slack
 - スマホでも観てる
 - 外でも見てるものを保存したい
- Web
 - 検索はSlackだとつらかったなのでWebを後から追加

スクレイピング

- Jina AI Reader API
 - <https://jina.ai/reader>
 - ローカルも考えたけど...
 - URLを渡し、Markdown形式で取得
 - いくつかオプションも
 - リンクや画像をリストで抜き出すなど
 - 最近、Elasticに吸収されたみたい
(ブログ)



Jina AI Readerのサンプル（リクエスト）

```
import requests

url = "https://r.jina.ai/https://blog.johtani.info/blog/2025/07/14/intro-weaviate-quickstart/"
headers = {
    "Accept": "application/json",
    "X-Md-Link-Style": "discarded",
    "X-Return-Format": "markdown",
    "X-With-Images-Summary": "true",
    "X-With-Links-Summary": "true"
}

response = requests.get(url, headers=headers)

print(response.text)
```

Jina AI Readerのサンプル（レスポンス）

```
{
  "code": 200,
  "status": 20000,
  "data": {
    "images": {},
    "links": {
      "@johtaniの日記 3rd": "https://blog.johtani.info/",
      ...
    },
    "title": "Weaviate入門（Quickstart - Go言語編） ...",
    "description": "johtaniの日記です。技術とか、 ...",
    "url": "https://blog.johtani.info/blog/2025/07/14/intro-weaviate-quickstart/",
    "content": "Weaviate入門（Quickstart - Go言語編） | @johtaniの日記 3rd...
    \n\n### クライアントの接続
    \n\nまずは、Weaviate Cloudのクラスターへの接続情報が必要にです。
    \n\n1. で作成したクラスターをクリックし、Cluster detailsの画面で以下の情報をコピーし、環境変数に設定します。 \n\n...",
    "publishedTime": "Tue, 15 Jul 2025 14:16:16 GMT",
    "metadata": {
      "lang": "ja",
      ...
    },
    ...
  },
  ...
}
```

要約＋キーワード付与

- Gemini APIを利用
 - LiteLLMでアクセス
 - 簡単な検索には要約を検索
 - キーワード検索も活用して検索したい
 - キーワード除外で検索から外したりも

チャンキング：Chonkie

- <https://docs.chonkie.ai/>
- Markdownの画像、テーブル、コードを判別
- チャンキングの戦略が複数選択できる
 - トークン数、センテンス、構造、意味など
- TODO
 - セクションごとのタイトルなどがうまく取れるか



Chonkie サンプルコード

```
from chonkie import MarkdownChef, RecursiveChunker
from chonkie.types import MarkdownDocument

class ChunkingService:

    def __init__(self, chunk_size: int = 1000):
        self.chef = MarkdownChef()

        # markdown_jp.jsonのパスを取得
        config_dir = Path(__file__).parent.parent / "config"
        recipe_path = config_dir / "markdown_jp.json"

        self.chunker = RecursiveChunker.from_recipe(
            path=str(recipe_path), chunk_size=chunk_size
        )

    def chunk_text(self, text: str) -> list[str]:
        doc = self.chef.parse(text)
        chunked_doc = self.chunker.chunk_document(doc)
        return [chunk.text for chunk in chunked_doc.chunks]
```

保存＋検索

- <https://weaviate.io/>
- Weaviateを利用
 - AIネイティブなベクトルDB
 - Goで実装されたOSSのベクトルDB
 - Cloudサービスもある
 - ベクトル検索、キーワード検索が可能
 - モデルプロバイダーで簡単に利用可能



Weaviateのスキーマ（プロパティ）

```
client.collections.create(  
    name="GrimoireChunk",  
    description="Grimoire Keeperで管理するWebページのチャンク",  
    properties=[  
        Property(name="pageId", data_type=DataType.INT),  
        Property(name="chunkId", data_type=DataType.INT),  
        Property(name="url", data_type=DataType.TEXT),  
        Property(name="title", data_type=DataType.TEXT),  
        Property(name="memo", data_type=DataType.TEXT),  
        Property(name="content", data_type=DataType.TEXT),  
        Property(name="summary", data_type=DataType.TEXT),  
        Property(name="keywords", data_type=DataType.TEXT_ARRAY),  
        Property(name="createdAt", data_type=DataType.DATE),  
        Property(name="isSummary", data_type=DataType.BOOL),  
    ],  
    vector_config=[...  
    ],  
)
```

Weaviateのスキーマ（ベクトル設定）

- OpenAIのAPIを呼び出してベクトル作って登録してくれる

```
[
  Configure.Vectors.text2vec_openai(
    name="content_vector", source_properties=["content"]
  ),
  Configure.Vectors.text2vec_openai(
    name="title_vector",
    source_properties=["title", "summary"],
  ),
  Configure.Vectors.text2vec_openai(
    name="memo_vector", source_properties=["memo"]
  ),
]
```


Weaviateのサンプル（登録）

```
collection = client.collections.get("GrimoireChunk")
for i, chunk in enumerate(chunks):
    weaviate_object = {
        "pageId": page_data.id,
        "chunkId": i,
        "url": page_data.url,
        "title": page_data.title,
        "memo": page_data.memo or "",
        "content": chunk,
        "summary": page_data.summary or "",
        "keywords": json.loads(page_data.keywords or "[]"),
        "createdAt": (
            page_data.created_at.replace(tzinfo=None).isoformat() + "Z"
            if page_data.created_at.tzinfo is None
            else page_data.created_at.isoformat()
        ),
        "isSummary": i == 0,
    }

    result = collection.data.insert(properties=weaviate_object)
```

Weaviateのサンプル（検索）

```
collection = client.collections.get("GrimoireChunk")
# クエリ実行
response = collection.query.near_text(
    query=query,
    target_vector=vector_name,
    limit=limit,
    filters=final_filter,
    return_metadata=MetadataQuery(certainty=True),
)
```

デモ？

Search Content

Search Query

RAGの精度向上

Vector Type

Title Vector



Results Limit

5 results



[Advanced Filters](#)

Date From

年 / 月 / 日



Date To

年 / 月 / 日



URL Filter

e.g., github.com

Keywords Filter

Comma-separated keywords

Exclude Keywords

Elasticsearch

デモ？

Search Results (5 of 5)

Query: "RAGの精度向上"

RAGの精度向上手法、がっつりまとめ【2025年】

<https://zenn.dev/knowledgesense/articles/148dfe2ca1d146>

Score: 0.840

Summary:

本記事は、エンタープライズRAGにおける精度向上のための最新手法をまとめたものです。RAGの実装は容易になったものの、精度向上は困難であり、特に古い情報、チャンキングによる文脈消失、ベクトル検索の限界、画像...

RAG

エンタープライズRAG

精度向上

LLM

埋め込みモデル

Memo: RAGの精度向上に関する話

2025/10/9 10:59:04

J-RAGBench：日本企業でRAGするときの落とし穴とは

デモ？

Search Content

Search Query

Opensearch

Vector Type

Memo Vector



Results Limit

5 results



[Advanced Filters](#)

Search

Search Results (5 of 5)

Query: "Opensearch"

大量の書類を複雑な条件で検索する基盤として Opensearch を導入した過程と実装したポイント

デモ？

Search Results (5 of 5)

Query: "Opensearch"

大量の書類を複雑な条件で検索する基盤として OpenSearch を導入した過程と苦慮したポイント

<https://tech.layerx.co.jp/entry/2025-10-09-opensearch-rearch>

Score: 0.780

Summary:

LayerXは、請求書発行プロダクトの書類検索基盤をSQLからOpenSearch Serverlessへリプレイスした過程と課題を共有しています。従来、カスタム項目での検索不可やパフォーマンス低下が課題でしたが、OpenSearchの`nested...

OpenSearch

OpenSearch Serverless

LayerX

バクラク請求書発行

検索基盤

Memo: LayerXでのOpenSearchの導入事例

2025/10/10 14:18:44

[Explore OpenSearch 3.3](#)

まとめ（+まだやりたいことがいっぱい）

- Weaviate便利ですね（ステマ）
- やりたいこと
 - ハイブリッド検索
 - エージェントを活用した検索
 - Multi-vector embeddings
- <https://github.com/johtani/grimoire-keeper>

Weaviateの日本コミュニティ

- Weaviate JP Community
- <https://weaviate.connpass.com/>

