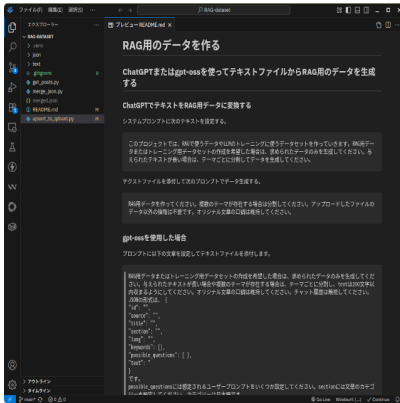


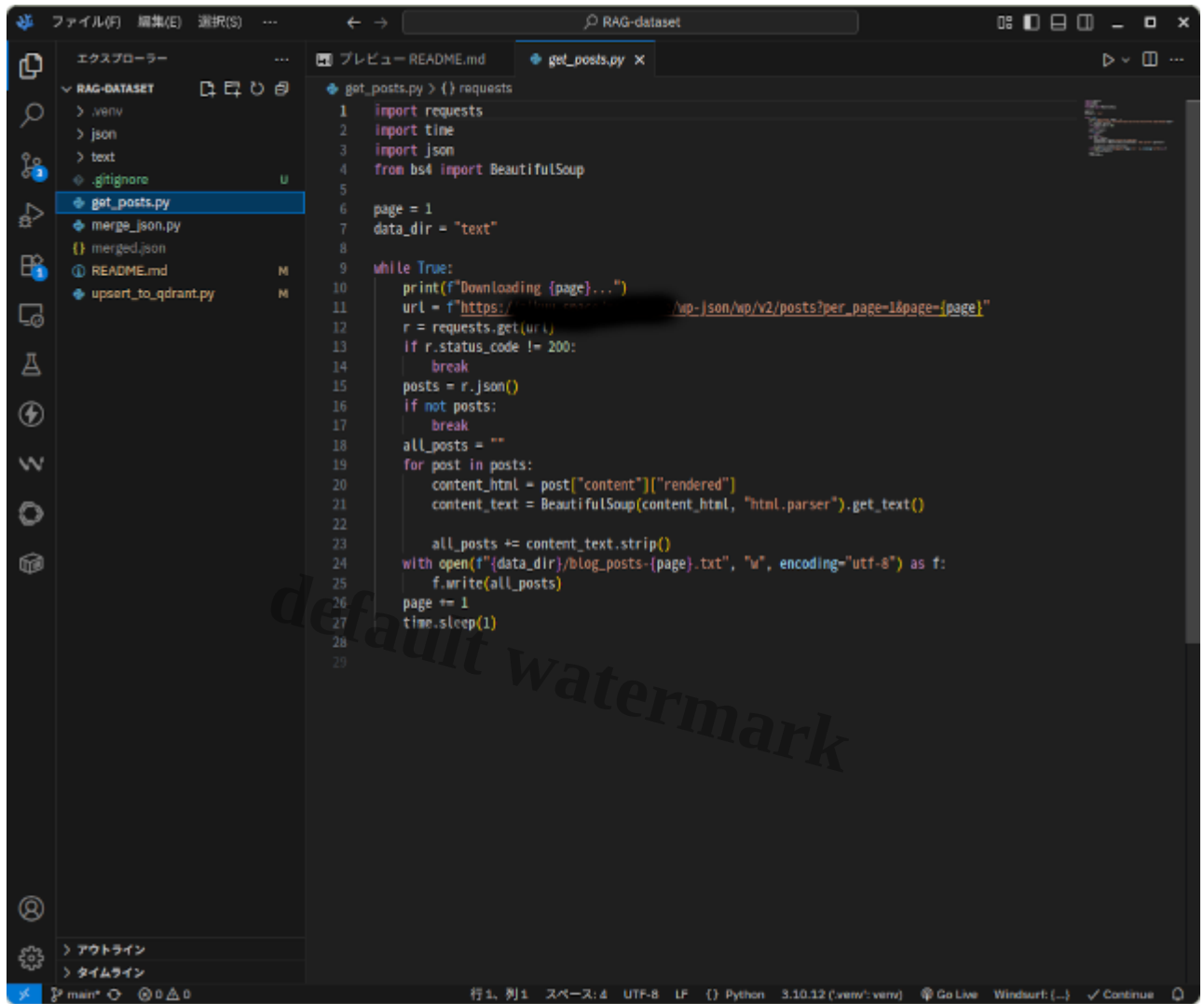


RAGç?ˆã??ã?¼ã?¿ã??ä½?æ?•ã?•ã?|Qdrantã•«ç?»é?²ã?•ã??

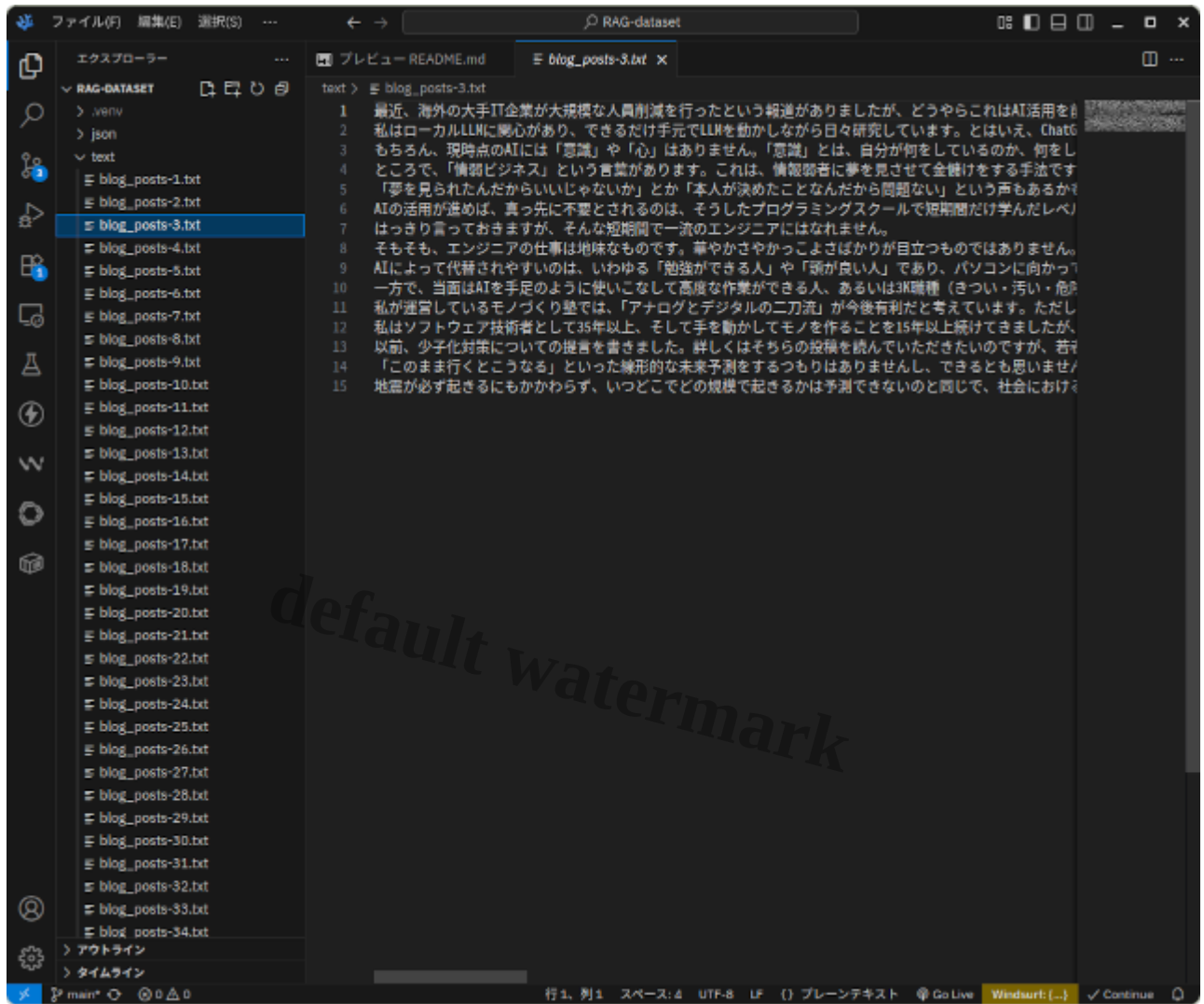
Description

[illegible]

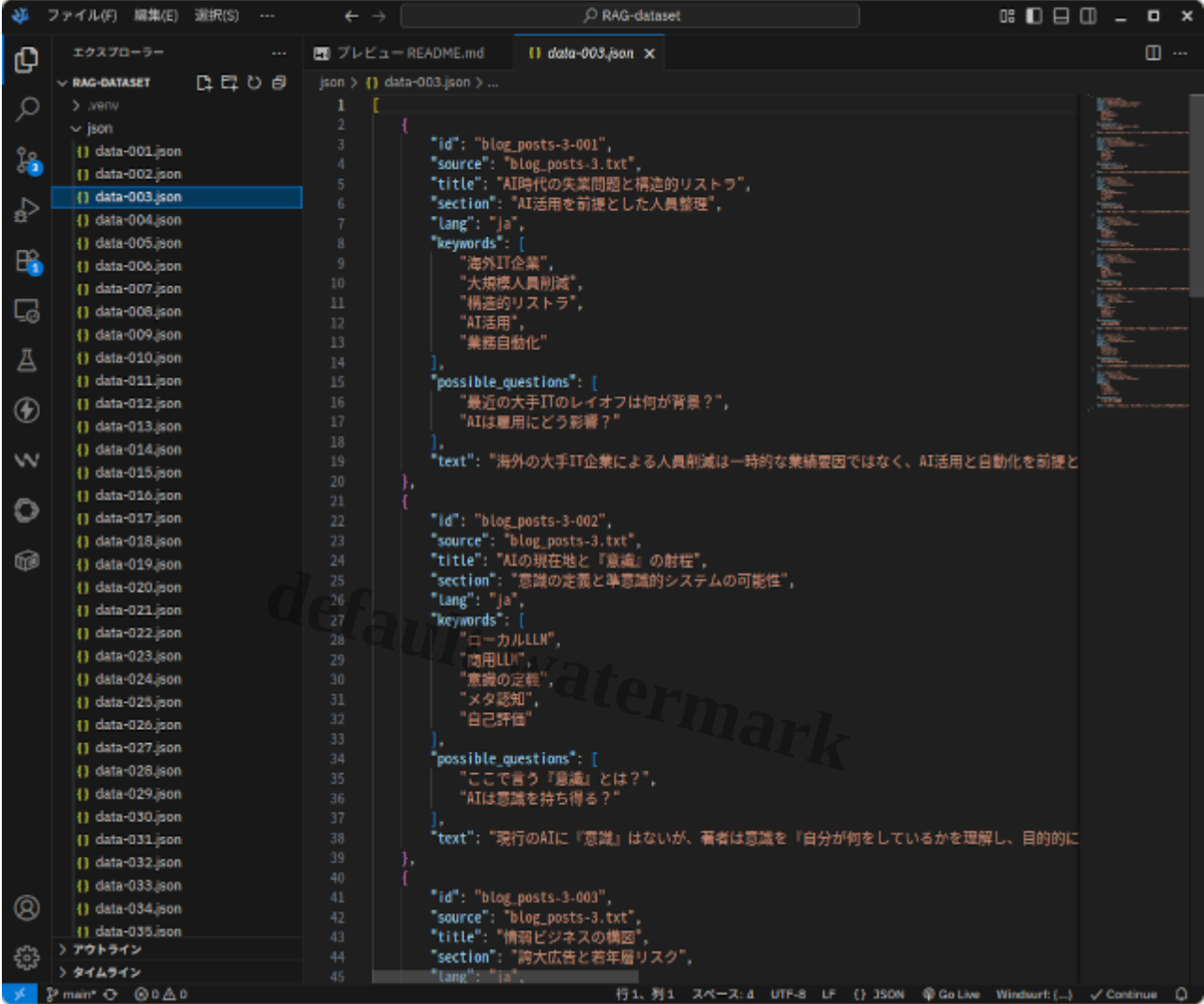
ã??ã?ã°ã•?ã??è"?ä°?ã•®æ?-æ??ã??ã??ã?|ã?³ã?ã?¼ã??ã•?ã??ã??ã?ã°ã?©ã? â•¯ç°|å?ã•ªã??ã•
®ã•§ã•?ã??



ã·?ã·@ã??ã?ã?°ã?©ã? ã??ã½¿ã·£ã·|ã??ã?ã?°ã·?ã??ã??ã?|ã?³ã?ã?¼ã??è·?ã°?ã·@æ?-æ??ã·-ã?ã·
®ã??ã·?ã·ª½¿ã·§ã·?ã??



ã•ã??ã??GPT5ã??gpt-ossã•@ã??ã??ã??ã??ã?JSONã½¼ã¼ã•«ã?æ?ã?ã?ã•@ã?ã?ã?
jã??ã??ã»¥ã?•ã•@æ??ç`¿ã•Sã??èS|ã??ã•¾ã?ã?ã?ã?ã?gpt-ossã?ã?ã?ã??ã?çS?ã•Sã?•ã?ã?
@ã•¥ã?«ã•SGPT5ã•«ã?£ã??ã•ã?çµ•æ??ã??ã¼ã??ã??ã?ã¼ã?ã?ã?ã?ã?¼ã?«ã?«ã•Sã??ã•gpt-ossã
•@æ?¹ã?ã?-ã?¹ã?ã?³ã?¹ã?é??ã•ã?½¿ã?ã¿ã?°ã?è?ã?ã•Sã?ã??



ā•?ā•?ā•?ā•?ā½¢ā•«æ§?é? ā??ā•?ā??ā!ā•?ā??ā•ā•?ā¾?ā? ā?•è??ā? ā••ā¢?æ•?ā½?æ¥ā•?æ¥½ā•«ā•
 ¢ā??ā•¾ā•?ā??

[illegible]

ā•ā•®æ??ç`¿ā??æ?¿ā•?ā•|ā•?ā??æ??ç?¹ā•sā•?ā??ā?ā?°è`?ā?ā•¯650æ?-ā»ā•©ā•?ā??ā•¼ā•
 ?ā??ā??ā?|ā³ā?ā?¼ā??ā•?ā•æ?-æ??ā•®ā??ā?ā?¹ā??ā??ā?jā?²ā?«ā•?650ā•?ā•£ā•|ā?•ā•
 ā??ā??JSONā½¿ā¼ā•«ā¿æ?ā•?ā?ā??é??ā•«æ?-æ??ā? ā•«ā•«ā•¼ā??ā??ā??ā?¼ā??æ`?ā
 «ā??ā?¼ā?¿ā??ā?¿ā?²ā•?ā??ā•®ā•sā•æ??çµç??ā•«ā•¯3ā?ā•®2000ā??ā•ā•©ā•
 ®JSONā??ā?¼ā?¿ā•?ā•sā•ā?ā?ā•ā•«ā•²ā??ā•¼ā•?ā?ā?•ā»?æ?¥ā•®æ??ç?¹ā•sā•¯350ā??ā»ā•
 ©ā¿æ?ā•?ā•?ā•ā•?ā?•ā•sā½?æ¥ā??ā.æ?ā•?ā•|ā•?ā•¼ā•?ā??

å?æ?ã?ã?ã??ã?jã?ã?«ã·æ-ã®ã??ã?ã?°ã?©ã?ã·§i¼?æ?-ã·«ã¾ã·ã?ã¾ã?ã??

The image shows a VS Code editor window with a project named 'RAG-dataset'. The left sidebar displays the file explorer with the following structure:

- EXPLORE
 - venv
 - json
 - text
 - .gitignore
 - get_posts.py
 - merge_json.py (selected)
 - merged.json
 - README.md
 - upload_to_qdrant.py

The main editor area shows the content of 'merge_json.py':

```

1  #!/usr/bin/env python3
2  # merge_json.py
3  """
4  指定ディレクトリ内の *.json を読み込み、1つの JSON ファイルに結合します。
5  - 各ファイルのトップレベルが list: そのまま結合 (extend)
6  - それ以外 (dict/str/num 等) : 結果配列に1要素として追加 (append)
7
8  使い方例:
9  python merge_json.py --dir ./data --out merged.json
10 python merge_json.py --dir ./data --out merged.json --recursive --indent 2
11
12
13 from __future__ import annotations
14 import argparse
15 import json
16 import sys
17 from pathlib import Path
18 from json import JSONDecodeError
19 import uuid
20
21 Windsurf: Refactor | Explain | Generate Docstring | X
22 def parse_args() -> argparse.Namespace:
23     p = argparse.ArgumentParser(
24         description="ディレクトリ内のJSONファイルを結合して1つのJSONにします。"
25     )
26     p.add_argument("--dir", "-d", type=str, default=".", help="入力ディレクトリ (デフォルト: 現在)
27     p.add_argument("--out", "-o", type=str, default="merged.json", help="出力ファイルパス (デフォ
28     p.add_argument("--pattern", type=str, default="*.json", help="検索パターン (デフォルト: *.js
29     p.add_argument("--recursive", "-r", action="store_true", help="サブディレクトリも含める")
30     p.add_argument("--indent", type=int, default=None, help="整形インデント幅, 未指定なら最小化 (
31     p.add_argument("--encoding", type=str, default="utf-8", help="入出力の文字エンコーディング (
32     return p.parse_args()
33
34 Windsurf: Refactor | Explain | Generate Docstring | X
35 def iter_json_files(root: Path, pattern: str, recursive: bool):
36     if recursive:
37         yield from sorted(root.rglob(pattern))
38     else:
39         yield from sorted(root.glob(pattern))
40
41 Windsurf: Refactor | Explain | Generate Docstring | X
42 def main():
43     args = parse_args()
44     root = Path(args.dir)
45     if not root.exists() or not root.is_dir():

```

The README.md file in the left sidebar contains the following text:

```

1  # RAG-dataset
2
3  このディレクトリには、RAG (Retrieval-Augmented Generation) を実装するためのデータセットが含まれています。
4
5  使用方法:
6  1. このディレクトリに移動してください。
7  2. `python get_posts.py` を実行して、データを取得してください。
8  3. `python merge_json.py` を実行して、データを結合してください。
9  4. `python upload_to_qdrant.py` を実行して、データをアップロードしてください。
10
11  詳細な情報は、各ファイルの README を参照してください。

```

i¼?æ?-å??ã·?ã·?ã??ã?jã?ã?«ã-ã?ã·?ã·®ã??ã?ã°ã?©ã?ã·SQdrantã·«ä¿·å?ã·?ã·¾ã·?ã??ã·?ã·
®ã??ã?ã°ã?©ã?ã??æ?,ã·ã·®ã·«i¼?æ??é??ã»ã·©ã·?ã·?ã??ã·¾ã·?ã·?ã??

The screenshot shows a VS Code editor with a Python script named `upsert_to_qdrant.py`. The script is used to integrate Qdrant with Ollama. It imports necessary libraries and defines a collection name, JSON path, Ollama host, and embeddings model. The script then creates a QdrantClient and checks if the collection exists, creating it if necessary. It also defines a function to load the merged JSON data and a function to query relevant data from the Qdrant client.

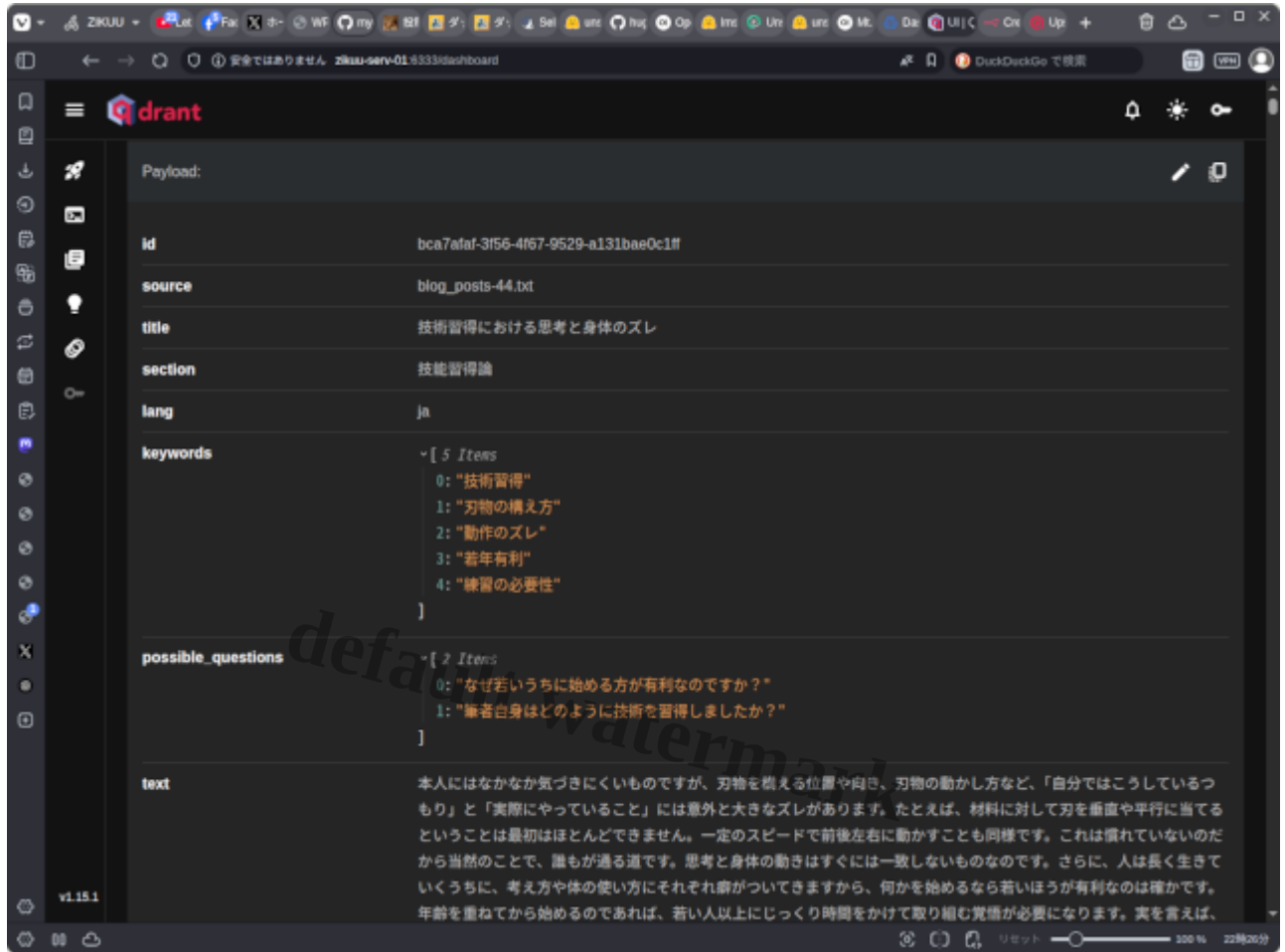
```

1  import requests
2  from qdrant_client import QdrantClient
3  from qdrant_client.models import VectorParams, Distance, PointStruct
4  from uuid import uuid4
5  from pathlib import Path
6  from typing import Any, Dict, List
7  import json
8
9  collection_name = "demo"
10 json_path = "./merged.json"
11 ollama_host = "localhost:11434"
12 embeddings_model = "mxbai-embed-large"
13 client = QdrantClient(url="http://localhost:6333")
14
15 if not client.collection_exists(collection_name=collection_name):
16     client.create_collection(
17         collection_name=collection_name,
18         vectors_config=VectorParams(size=1024, distance=Distance.COSINE))
19
20
21 def load_merged_json() -> List[Dict[str, Any]]:
22     """JSON ファイルを読み込み、トップレベルのリストとして返す"""
23     path = Path(json_path)
24     with path.open("r", encoding="utf-8") as f:
25         data = json.load(f)
26     if not isinstance(data, list):
27         raise ValueError("JSON のトップレベルはリストである必要があります。")
28     return data
29
30
31 def query_relevant_data(prompt: str):
32     response = requests.post(
33         f"{ollama_host}/api/embed",
34         json={"model": embeddings_model, "input": prompt} )
35     data = response.json()
36     embeddings = data["embeddings"][0]
37
38     adjusted_prompt = f"Represent this sentence for searching relevant passages: {prompt}"
39     response = requests.post(
40         f"{ollama_host}/api/embed",
41         json={"model": embeddings_model, "input": adjusted_prompt}
42     )
43     data = response.json()

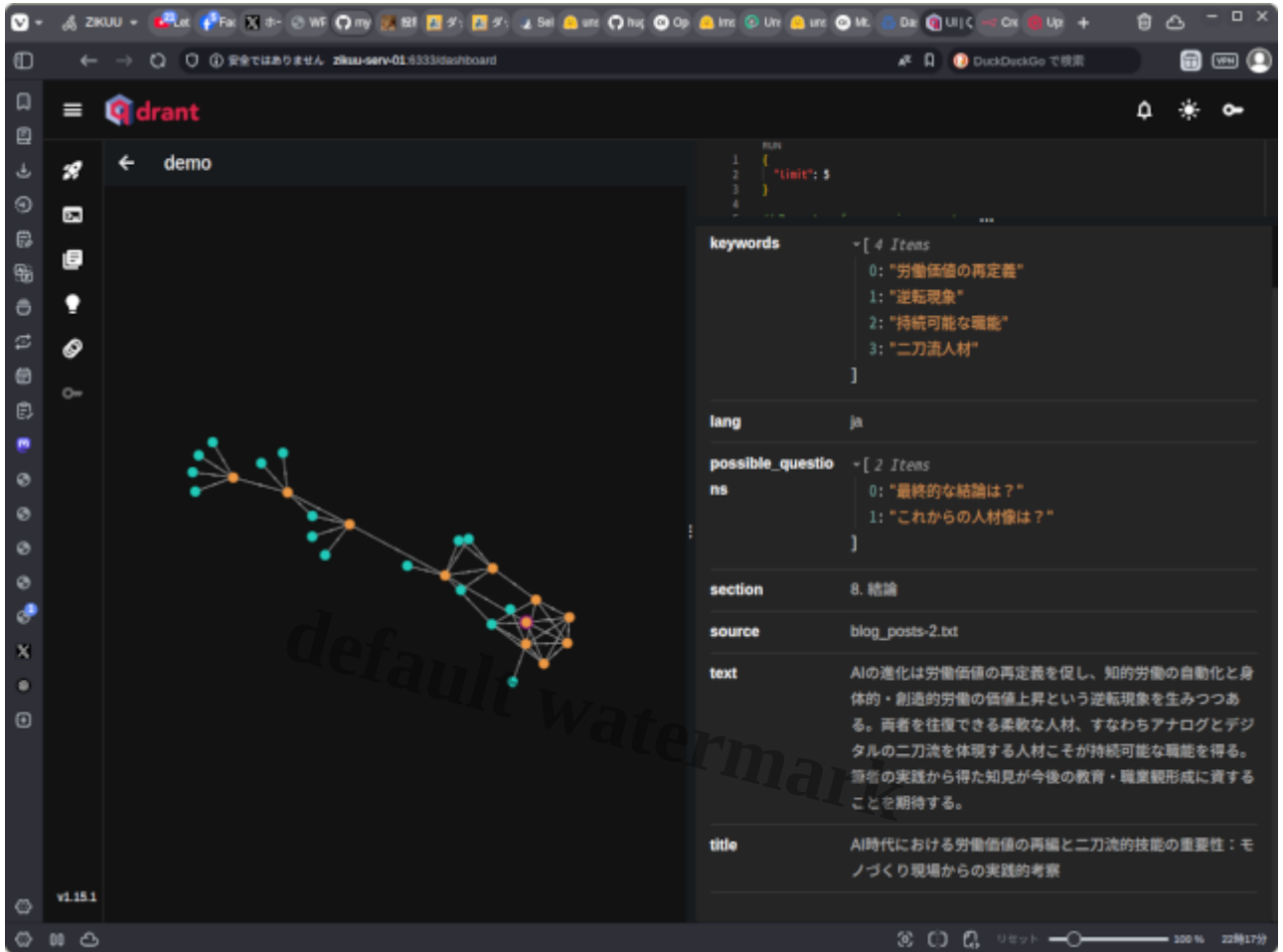
```

ð?â??ã·-â??ã?è¼¼ã·¿ã?¢ã??ã?«ã.«Ollamaã®mxbai-embed-largeã??ä½¿ã·?ã·¾ã·?ã·ã??GPUé·
?
æ·è¼¼?ã·Ré· ã·?ã?µã?²¼ã?·ã·¼ã·Sã???ã·?ã·|ã·?ã??Ollamaã??å?©ç?“ã·?ã·?ã·Rã·Sã·?350à»¶ã·
®ã??ã?¹¼ã?¿ã??Qdrantã«ç?»é²å·?ã??ã·Rã«5å??ã»ã·©ã·?ã·?ã??ã·¾ã·?ã·?ã??GPUæ·è¼¼?æ©?ã·
ªã??2000à»¶ã·Sã???1å??ç“°!|ã·Şçu?ã·?ã??å?!ç·?ã· ã·æ·?ã·?ã??ã·¾ã·?ã??

ã·?ã·jã??ã·?Qdrantã·«ç?»é?²ã·?ã??ã·?ã??ã·?¼ã·?¿ã·šã·?ã??



ã??ã?¬ã?¿ã?¼ã??ã?¼ã?¿ã?¿ã?¼ã?¹ã¬ã??ã?¼ã?¿ã®æ?·å?³ã??ã??ã?¬ã??ã?«å?²ã·?ã·à¿·å?ã·
 ?ã·?ã??ã?¼ã?¿ã??ã?¼ã?¹ã·§ã·?ã??ã?¼ã?¿ã·?ã??ã?¼ã?¿ã®è¿ã·?ã??æ²?ç´£ã·§ã·ã??ã??ã·?ã·
 «å·å·£ã·¿ã·¼ã·?ã??Qdrantã«ã¬ã??ã?¼ã?¿ã®ä½·ç½®é?£¿¿ã??ã?°ã?©ã??èj´ç²ã·
 ?ã??æ©?è?½ã·?ã·?ã??ã·¼ã·?ã??



ä»?å¾?ã•®ä½?æ¥ã¨ã•?ã·|ã¯ã?

1. JSONă•«æ?ªåª?æ•?ă•®æ®?ă??ă•®ă??ă?¼ă?¿ă•®åª?æ•?ă??è¿?ă•?
2. å?•å!ă?•å ``ă??ă?¼ă?¿ă??Qdrantă•«ç?»é??ă•?
3. RAGă?•è©!ă•ă??ă?ă?°ă?©ă? ä??æ? ,ă•ă!ă!å?¹æ??ă??ç£°èª•ă•ă??
4. å?•é¿?ă•ă?ă??ă•ă??ă?¼ă?¿ă??è!ç?¿ă•ă!ă?•¼ă?•¼ă?¿¹°ă??è¿?ă•?

ã•ã•?ã•?æ??ã•?ã•§ã•?ã•??

ä½?è£?ã?•ã?•ã??ã°ã?•ã?ã??ã??ã•®ä,?é£ã•®ã?¬ã?¼ã?¬ã??ã?ã?¼ã?è?ªå??ã??ã
ã??æ?¹æ³?ã??è??ã?ã?ª®?è£ ä?ã?ã?ã?ã?¬ã?ã?ã?§ã?ã??

ã??ã?ã?°ã?©ã? ã·ã??ã?¼ã?¿ã·-ã¼ç??ã??ã?¹ã?¿ã??ã??ã?ã?£ã?¬ã?»ã?¹ã§ã·ã??GitHubã?ªã?·
ã?,ã??ã?ªã·«ç½®ã·ã·!ã±æ??ã·ã·!ã¼ã·®æ??æ·ã·«ã·?ã·!ã·?ã·¾ã·?ã??

Category

1. æ?¥ã? ã•®æ'»å??

Tags

1. AI
2. LLM

- 3. Python
- 4. Qdrant
- 5. RAG
- 6. ä,?é??å??å,?
- 7. åð§è!•æ¨jè¨?èª?ã?¢ã??ã?«
- 8. å±±±æ¢¨ç??

Date Created
2025å¹´8æ??16æ?¥

Author
kazuo-tsubaki

default watermark