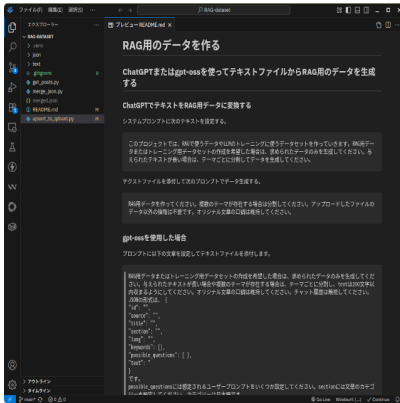


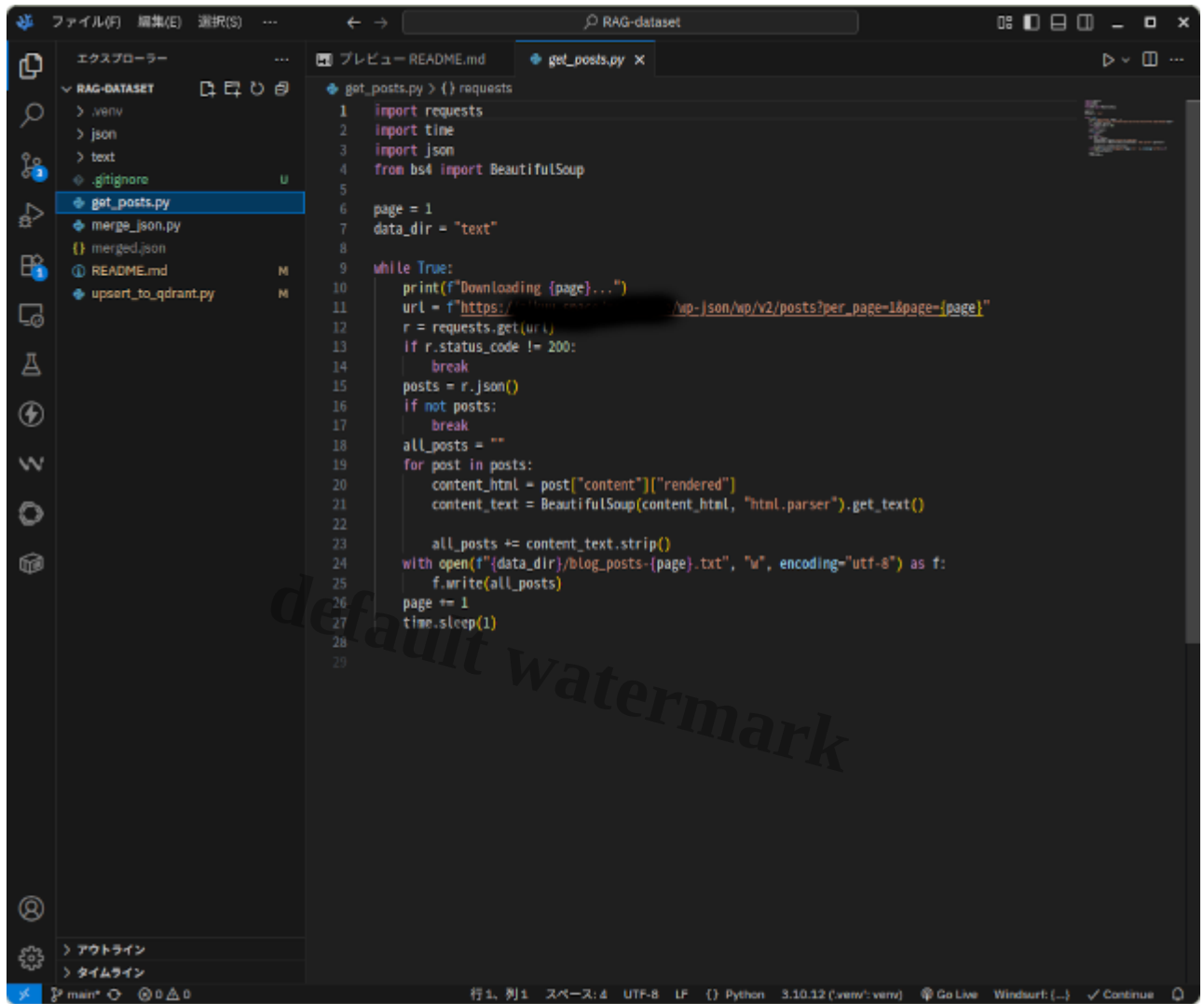


RAGç?`ã??ã?¼ã?¿ã??ä½?æ?•ã?•ã?|Qdrantã«ç?»é?²ã?•ã??

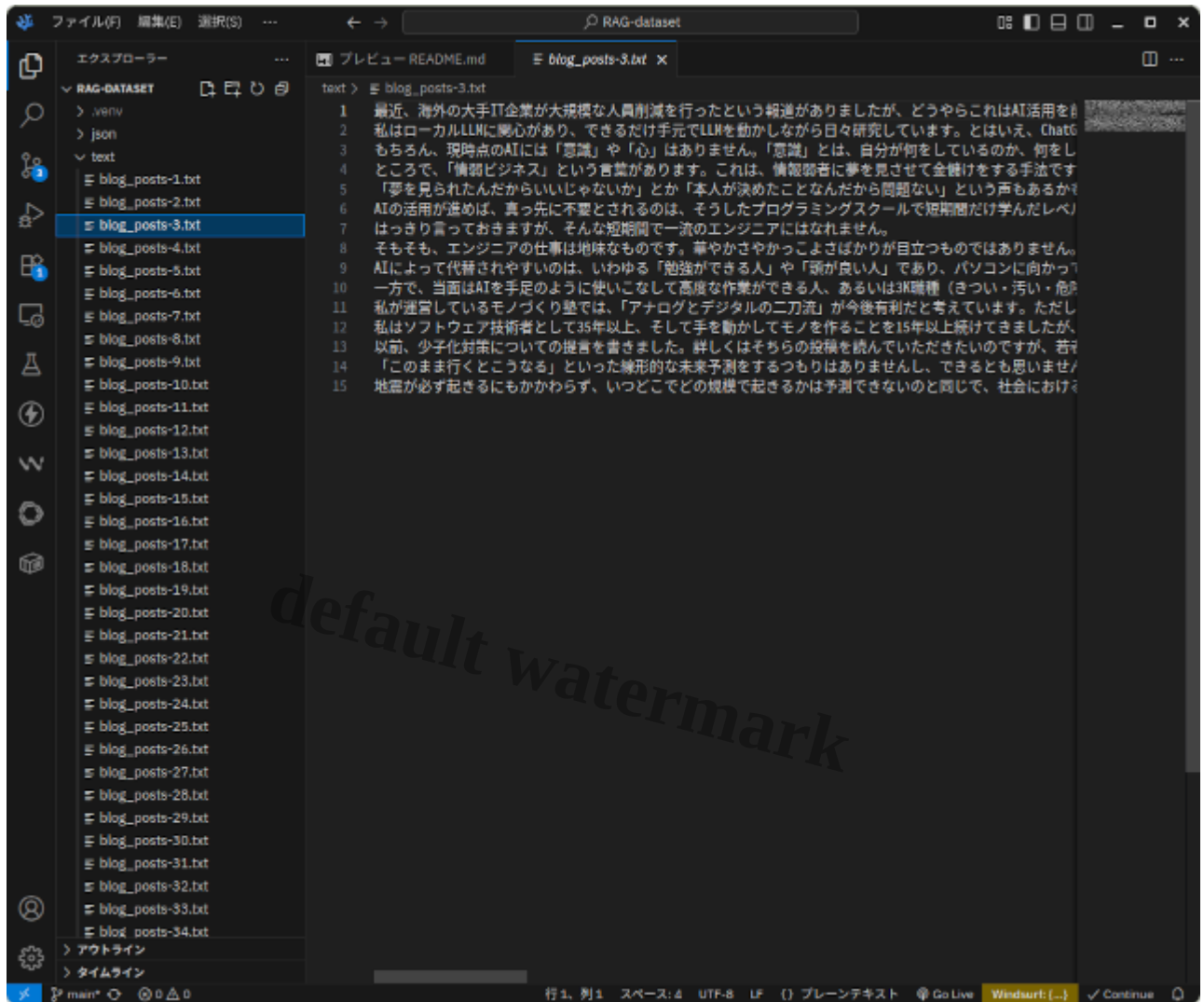
Description

à»?æ?¥â•̂-â?â•̂®â??â?â?•â•̂®è°â°â??â??â?|â?³â?â?¼â?â?â•̂-â•̂|â?•GPT5â??gpt-ossâ•̂Sâ½?æ?•â•̂-â?•?RAGç?•â??Qdrant â??â?•â?•¿â?¼â??â?¼â?¿â??â?¼â?¹â«ç?»é?â•̂?â??â??â?-â?•â?©â? â??æ?•,â•̂-â•̂¾â?â?â?â??â?â?â?©â? â•̂-â•̂â•̂-â•̂|Pythonâ•̂Sè°è¿•â??â?•â??â??â??æ?•£â•̂?â°â?µâ?³â??â?«â?³â?¼â??â?â?â?â•̂-â•̂?â??è|?â•̂-â?â??â?â•̂®â•̂Sâ?é?£â•̂?â?â??â?-â?•â?©â? â•̂-â•̂â•̂?â•̂-æ?•â?â?¾â?â??

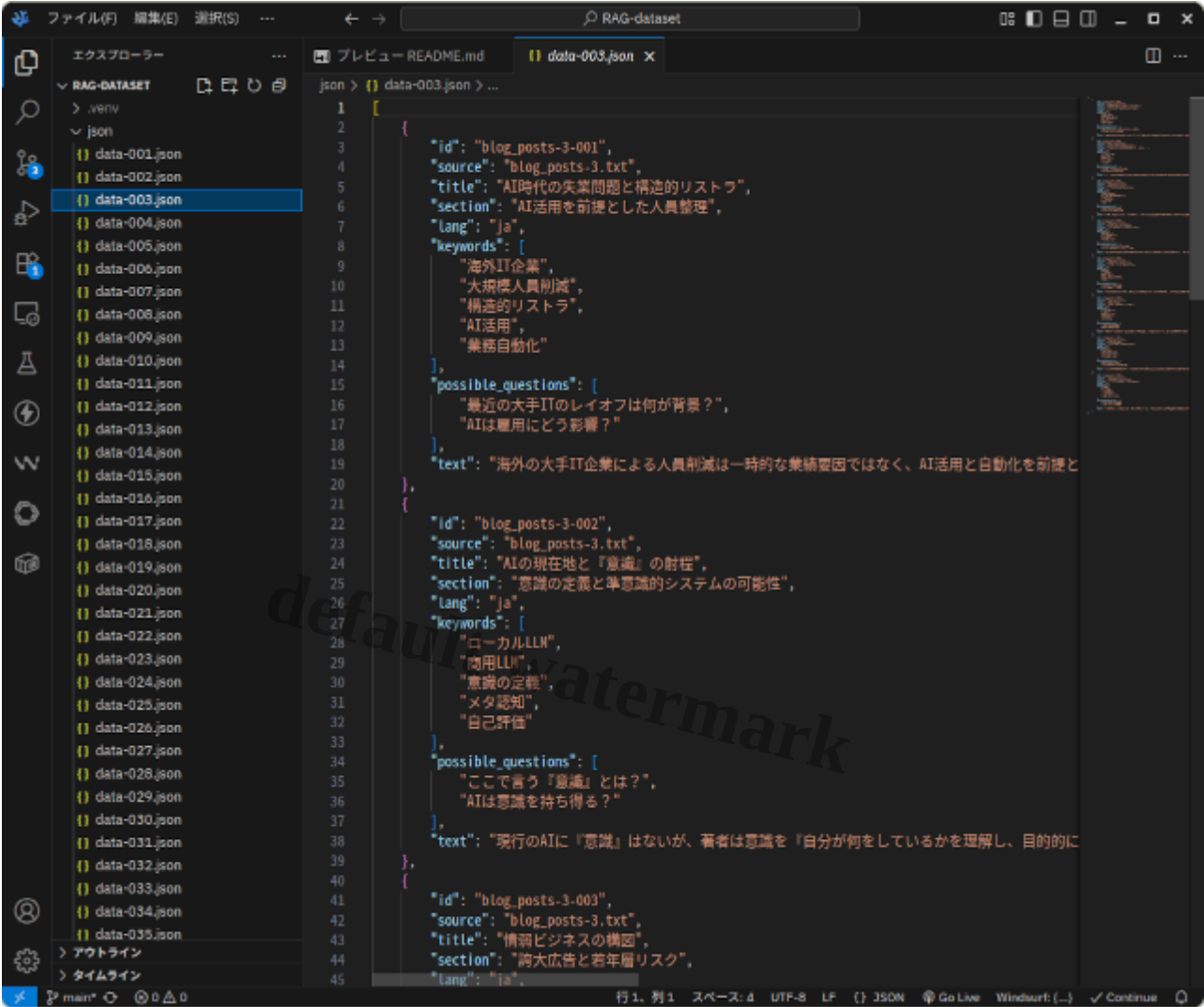
ã??ã?ã°ã•?ã??è"?ä°?ã•®æ?-æ??ã??ã??ã?|ã?³ã?ã?¼ã??ã•?ã??ã??ã?ã°ã?©ã? â•¯ç°|â?ã•ã??ã•
®ã•§ã•?ã??



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?ã?ã?ã?ã?ã?ã?ã?ã?ã?ã?
@ã•¥ã?ã•«ã•Sã?
•@æ?¹ã?ã?-ã?¹ã?ã?³ã?¹ã?é??ã•ã?½¿ã?ã¿ã?°ã?è?ã?ã?ã?ã?ã?ã?ã?ã?ã?ã?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??

ã•?ã•?ã•¾ã•§ã•®ä½¿?æ¥ã•¯æ?¬æ¥ã•ã??ã•°èªã??ã??ã•?ã•?ã•?ã•·æ?ã•£ã•|ã•?ã•¾ã•?ã•?ã•?ã•?
¾æ???ç?¹ã•§ã•¯ã¤?æ?ã•?ã??ã•?ã??ã•?¼ã?¿ã??ã??ã? ã«çº°ªã•?ã??ã•?ã??ã«ã??ã?¥ã«ã??ã?-
ã?ã?©ã? ã§ãå?|ç?ã•?ã•|ã•?ã•¾ã•?ã??ã•?ã??ã•?çµæ§?ã¤§ã¤?ã•§ã•?ã•?ã•?ã??ã•?ç??æ?•A|ã•
®ã??ã??ã??ã??ã•ã•ã•£ã•?ã??ã•?ã•?ã•®ã¤?æ?ã½¿?æ¥ã«10ã?•ã•ã??ã•?ã•®æ??é??ã•?ã•?ã•?ã•?ã•
¯ã?ã•§ã•?ã??ã•?ã??ã•§ã??ç?¿½¿ã•®ç?ã??ã??ã•?é?æ?ã•?ã??ã•|ã•?ã•¾ã•?ã•?

ā•ā•®æ??ç`¿ā??æ?¿ā•?ā•|ā•?ā??æ??ç?¹ā•šā••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?ā?«ā•?650ā•?ā•£ā•|ā?•ā•
 ā??ā??JSONā½¿ā¼ā•«ā?æ?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•šā••æ??çµç??ā«ā•¯3ā?ā•®2000ā??ā•ā•©ā•
 ®JSONā??ā?¼ā?¿ā•?ā•šā••ā??ā•?ā•ā«ā•²ā??ā•¼ā•?ā•?ā?•ā»?æ?¥ā•®æ??ç?¹ā•šā•¯350ā??ā»ā•
 ©ā?æ?ā•?ā•?ā•ā•?ā?•ā•šā½?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
- ▼ RAG-DATASET
 - 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: *.json
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 ## 使い方
4
5 1. このディレクトリに *.json ファイルを配置する
6 2. このディレクトリで merge_json.py を実行する
7 3. 出力ファイルは merged.json となる
8
9 例:
10 python merge_json.py --dir ./data --out merged.json
11 python merge_json.py --dir ./data --out merged.json --recursive --indent 2
```

i¼?æ?-å??ã?ã?ã??ã?|ã?ã?«ã-ã?ã?ã?®ã??ã?ã?°ã?©ã? ã·S Qdrantã«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 interact with a Qdrant vector database. It imports necessary libraries like `requests`, `QdrantClient`, `VectorParams`, `Distance`, `PointStruct`, `uuid4`, `Path`, `Any`, `Dict`, `List`, and `json`. The script defines a `collection_name` of `"demo"`, a `json_path` of `"./merged.json"`, an `ollama_host` of `"localhost:11434"`, and an `embeddings_model` of `"mxbai-embed-large"`. It creates a `QdrantClient` and checks if the collection exists, creating it if necessary. It then defines a function `load_merged_json()` to load the JSON data and a function `query_relevant_data()` to query the Qdrant client.

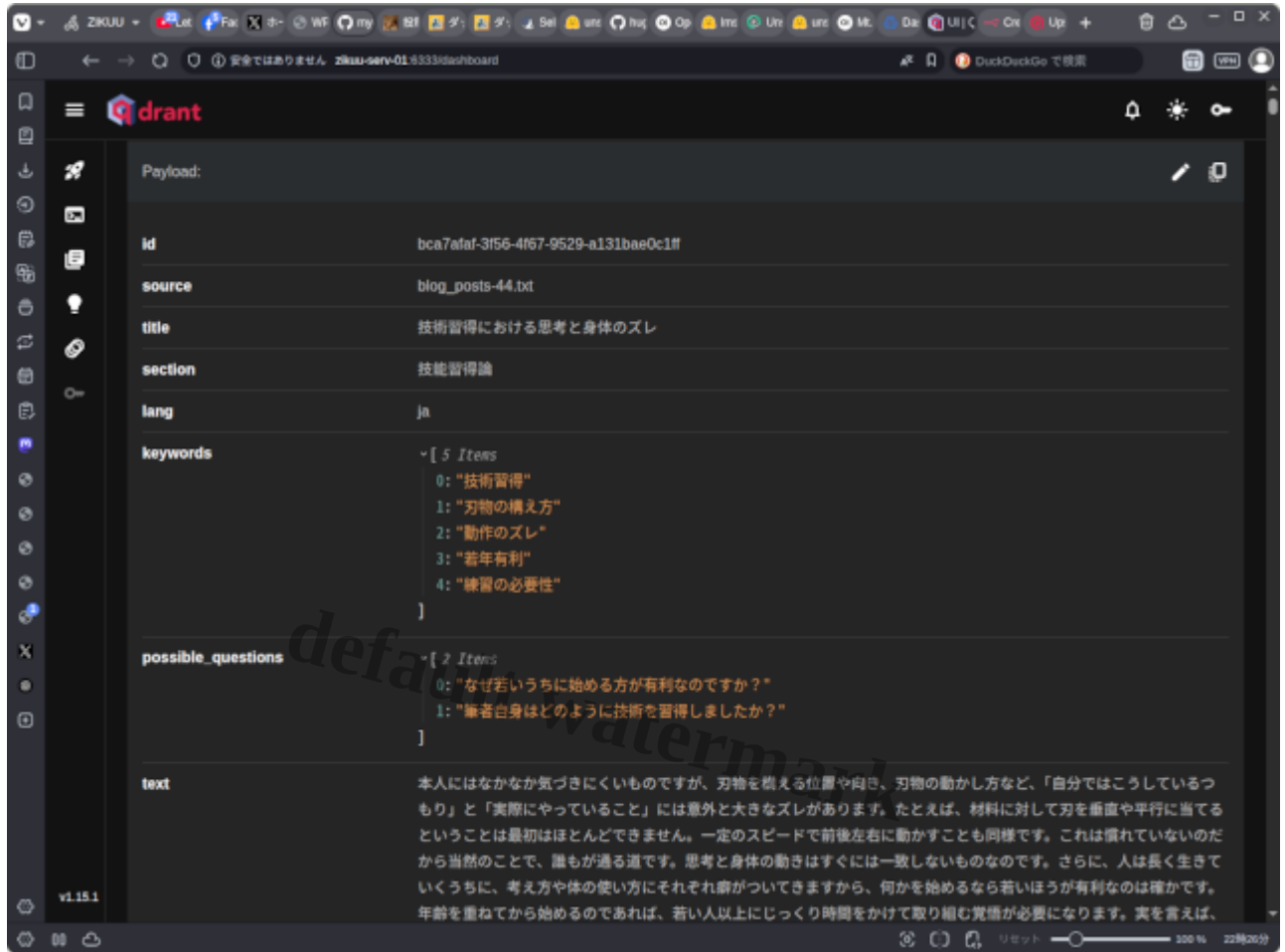
```

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()

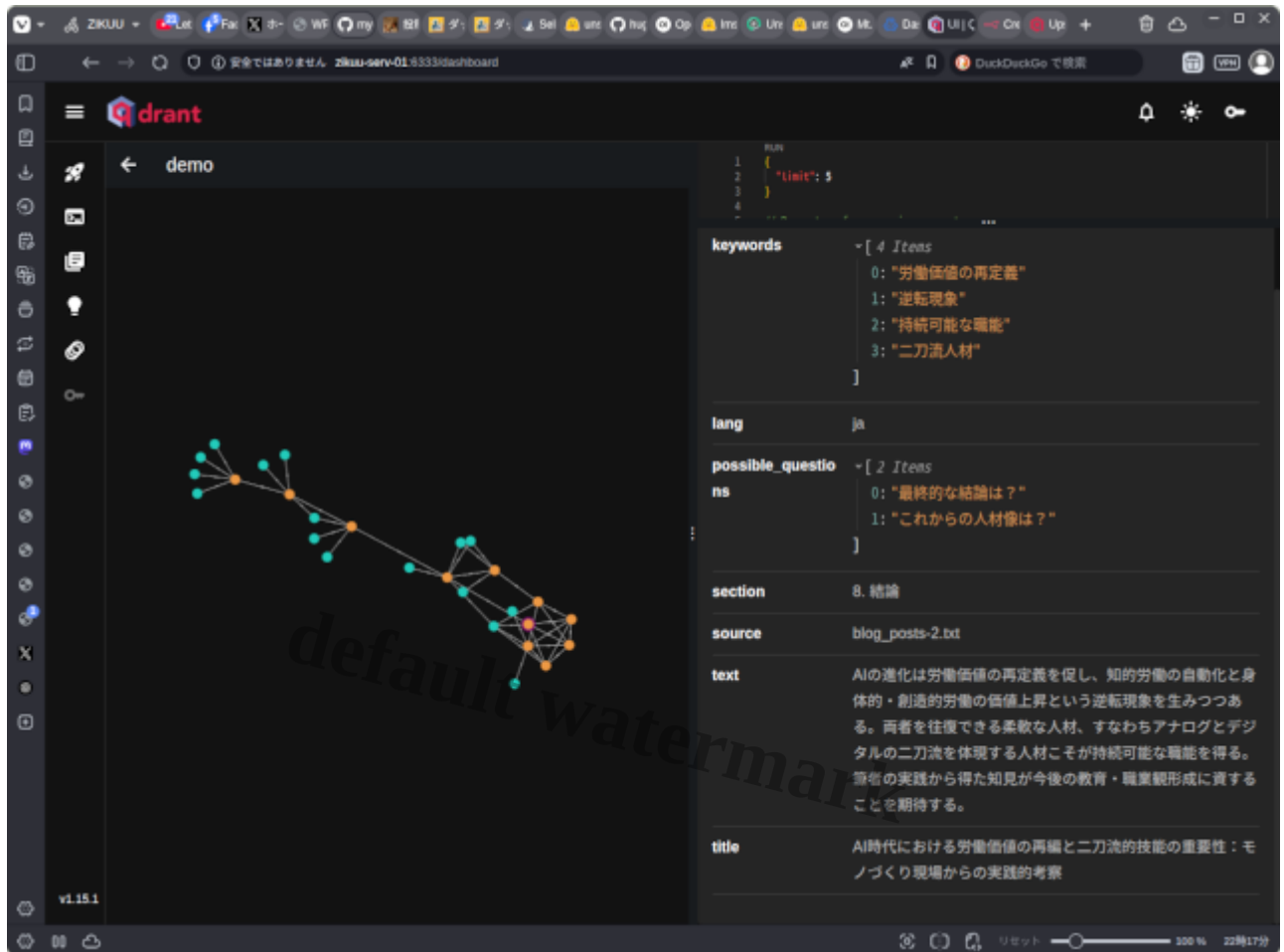
```

à?â?ã-ä?å?è¼¼ä·¿ä?¢ä?ä?«ä«Ollamaä®mxbai-embed-largeä??ä½¿ä·?ä¾ä·?ä?ä?GPUé·
æ·è¼¼ä·Ré· ä·?ä?µä?¼ä·ä·¼ä·Sä??ä·ä·ä·?ä?Ollamaä??ä?©ç?ä·ä·ä·®ä·Sä?·350ä»¶ä·
®ä??ä?¼ä?¿ä??Qdrantä«ç?»é²ä·ä?ä·ä·®ä«5ä??ä»ä·©ä·ä·ä??ä¾ä·ä·ä??GPUæ·è¼¼æ©?ä·
ªä??2000ä»¶ä·Sä??1ä??ç?°!ä·Sçu?ä·ä??ä?|ç?ä· ä·æ·ä·ä??ä¾ä·ä??

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·jã·?ã·¾ã·?ã??Qdrantã«ã¬ã??ã?¼ã?¿ã®ä½·ç½®é?£¿¿ã??ã?°ã?©ã??èj´ç²°ã·
 ?ã??æ©?è?½ã·?ã·?ã??ã·¾ã·?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?¹æ??ă??ç£°èª•ă•ă??
4. å?•é¿?ă•ă?ă??ă•ă??ă?¼ă?¿ă??è!ç?¿_ă•ă•!ă?•¼ă?•¼ă?¿ç¹ªă??è¿?ă•?

ã•ã•?ã•?æ??ã•?ã•§ã•?ã•??

ä½?è£?ã°ã?ã??ã°ã?•ã°ã??ã??ã°ã°ã°,é?£ã°ã°ã°¼ã°ã°ã°ã°¼ã°?èªå??å??ã
ã°ã°æ?¹æ³?ã°?è??ã°ã°!ã°?è£ ä°ã°ã°ã°ã°ã°ã°ã°ã°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