Project Documentation

Overview

This project consists of a Knowledge Graph Agent that utilizes various tools and libraries to process data and interact with a Neo4j database. The core functionality is to embed data, retrieve information, and interact with users via a web interface.

Files and Their Roles

1. **local_interface.py**
2. **neo4jFAQ.py**
3. **agent.py**
4. **data_processing.py**

1. local_interface.py

This file sets up the web interface using Streamlit. It provides functions to load Lottie animations, set up custom CSS, and handle user interactions such as file uploads.

Key components:

- `load_lottie_url(url: str)`: Loads Lottie animations from a given URL.
- `main()`: Sets up the Streamlit interface, initializes the database, and handles file uploads.

2. neo4jFAQ.py

This script contains the implementation for embedding retrieval from a Neo4j database using OpenAI embeddings. It defines a class `GraphEmbeddingRetriever` to interact with the Neo4j database.

Key components:

- `GraphEmbeddingRetriever(BaseModel)`: Pydantic model to define and manage the Neo4j database connection and embedding retrieval.

3. agent.py

This file sets up the agent that interacts with the database and handles embedding retrieval using OpenAI models. It initializes necessary connections and provides functions for database operations.

Key components:

- `EmbeddingRetriever(BaseModel)`: Class to handle database connections and embedding retrieval.
- `Secure API Key Fetching`: Ensures the OpenAI API key is securely fetched from environment variables.

4. data_processing.py

This script handles database operations, including setting up connections and performing queries. It uses the `psycopg2` library to interact with a PostgreSQL database.

Key components:

- `get_db_config()`: Fetches database configuration from environment variables.
- `Database Connection Pool`: Initializes a threaded connection pool for database interactions.

Setup Instructions

Prerequisites

- Python 3.8 or higher
- PostgreSQL database
- Neo4j database
- OpenAI API key
- Required Python packages (listed in `requirements.txt`)

Installation

1. **Clone the Repository**

   ```
   git clone <repository_url>
   cd <repository_name>
   ```

2. **Install Required Packages**

   ```
   pip install -r requirements.txt
   ```

3. **Set Up Environment Variables**

   Create a `.env` file in the project root and add the following variables:

   ```
   OPENAI_API_KEY=<your_openai_api_key>
   DB_USER=<your_database_user>
   DB_PASSWORD=<your_database_password>
   DB_HOST=<your_database_host>
   DB_PORT=<your_database_port>
   DB_NAME=<your_database_name>
   ```

Running the Project

1. **Start the Streamlit Interface**

   ```
   streamlit run local_interface.py
   ```

2. **Interact with the Interface**

   - Upload your data files (Excel or CSV).
   - The interface will process the files and interact with the backend to retrieve information from the Neo4j database.

Detailed Function Documentation

local_interface.py

```
def load_lottie_url(url: str):
    """
    Load Lottie animation from a URL.

    Args:
        url (str): The URL of the Lottie animation.

    Returns:
        dict: The JSON representation of the Lottie animation if successful, None
            otherwise.
    """
```

Listing 1: load_lottie_url function

```
def main():
    """
    Set up and run the Streamlit web interface.

    This function sets the page configuration, adds custom CSS, initializes the
        database,
    and handles file uploads for user interaction.
    """
```

Listing 2: main function

neo4jFAQ.py

```
class GraphEmbeddingRetriever(BaseModel):
    """
    Class to handle embedding retrieval from a Neo4j database.

    Attributes:
        neo4j_uri (str): URI for the Neo4j database.
        neo4j_username (str): Username for the Neo4j database.
        neo4j_password (str): Password for the Neo4j database.
        openai_api_key (str): OpenAI API key for embedding model.
        graph (Any): Neo4jGraph instance.
        llm (Any): Language model instance.
        embedding_model (Any): Embedding model instance.
    """
```

Listing 3: GraphEmbeddingRetriever class

agent.py

```
class EmbeddingRetriever(BaseModel):
    """
    Class to handle database connections and embedding retrieval using OpenAI models.

    Attributes:
        db_connection (Any): Database connection for retrieving embeddings.
        embeddings (Any): OpenAI embeddings model.
    """
```

Listing 4: EmbeddingRetriever class

data_processing.py

```
def get_db_config():
    """
    Fetch database configuration from environment variables.

    Returns:
        dict: Database configuration dictionary.
    """
```

Listing 5: get_db_config function

Notes

- Ensure the PostgreSQL and Neo4j databases are properly set up and accessible.
- Handle environment variables securely to avoid exposing sensitive information.