Graph DB QA chain

This notebook shows how to use LLMs to provide a natural language interface to a graph database you can query with the Cypher query language.

You will need to have a running Neo4j instance. One option is to create a free Neo4j database instance in their Aura cloud service. You can also run the database locally using the Neo4j Desktop application, or running a docker container. You can run a local docker container by running the executing the following script:

```
docker run \
    --name neo4j \
    -p 7474:7474 -p 7687:7687 \
    -d \
    -e NEO4J_AUTH=neo4j/pleaseletmein \
    -e NEO4J_PLUGINS=\[\"apoc\"\]  \
    neo4j:latest
```

If you are using the docker container, you need to wait a couple of second for the database to start.

```
from langchain.chat_models import ChatOpenAI
from langchain.chains import GraphCypherQAChain
from langchain.graphs import Neo4jGraph
```

```
graph = Neo4jGraph(
    url="bolt://localhost:7687", username="neo4j", password="pleaseletmein"
)
```

Seeding the database

Assuming your database is empty, you can populate it using Cypher query language. The following Cypher statement is idempotent, which means the database information will be the same if you run it one or multiple times.

```
graph.query(
    """
MERGE (m:Movie {name:"Top Gun"})
WITH m
UNWIND ["Tom Cruise", "Val Kilmer", "Anthony Edwards", "Meg Ryan"] AS actor
MERGE (a:Actor {name:actor})
MERGE (a)-[:ACTED_IN]->(m)
"""
)
```

```
    []
```

Refresh graph schema information

If the schema of database changes, you can refresh the schema information needed to generate Cypher statements.

```
graph.refresh_schema()
```

```
print(graph.get_schema)
```

```
            Node properties are the following:
            [{'properties': [{'property': 'name', 'type': 'STRING'}], 'labels': 'Movie'}, {'properties':
[{'property': 'name', 'type': 'STRING'}], 'labels': 'Actor'}]
            Relationship properties are the following:
            []
            The relationships are the following:
            ['(:Actor)-[:ACTED_IN]->(:Movie)']
```

Querying the graph

We can now use the graph cypher QA chain to ask question of the graph

```
chain = GraphCypherQAChain.from_llm(
    ChatOpenAI(temperature=0), graph=graph, verbose=True
)
```

```
chain.run("Who played in Top Gun?")
```

```
    > Entering new GraphCypherQAChain chain...
    Generated Cypher:
    MATCH (a:Actor)-[:ACTED_IN]->(m:Movie {name: 'Top Gun'})
    RETURN a.name
    Full Context:
    [{'a.name': 'Val Kilmer'}, {'a.name': 'Anthony Edwards'}, {'a.name': 'Meg Ryan'}, {'a.name': 'Tom
Cruise'}]

    > Finished chain.
```

```
    'Val Kilmer, Anthony Edwards, Meg Ryan, and Tom Cruise played in Top Gun.'
```

Limit the number of results

You can limit the number of results from the Cypher QA Chain using the `top_k` parameter. The default is 10.

```
chain = GraphCypherQAChain.from_llm(
    ChatOpenAI(temperature=0), graph=graph, verbose=True, top_k=2
)
```

```
chain.run("Who played in Top Gun?")
```

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
MATCH (a:Actor)-[:ACTED_IN]->(m:Movie {name: 'Top Gun'})
RETURN a.name
Full Context:
[{'a.name': 'Val Kilmer'}, {'a.name': 'Anthony Edwards'}]

> Finished chain.
```

```
'Val Kilmer and Anthony Edwards played in Top Gun.'
```

Return intermediate results

You can return intermediate steps from the Cypher QA Chain using the `return_intermediate_steps` parameter

```
chain = GraphCypherQAChain.from_llm(
    ChatOpenAI(temperature=0), graph=graph, verbose=True, return_intermediate_steps=True
)
```

```
result = chain("Who played in Top Gun?")
print(f"Intermediate steps: {result['intermediate_steps']}")
```

```
print(f"Final answer: {result['result']}")
```

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
MATCH (a:Actor)-[:ACTED_IN]->(m:Movie {name: 'Top Gun'})
RETURN a.name
Full Context:
[{'a.name': 'Val Kilmer'}, {'a.name': 'Anthony Edwards'}, {'a.name': 'Meg Ryan'}, {'a.name': 'Tom Cruise'}]

> Finished chain.
Intermediate steps: [{'query': "MATCH (a:Actor)-[:ACTED_IN]->(m:Movie {name: 'Top Gun'})\nRETURN a.name"}, {'context': [{'a.name': 'Val Kilmer'}, {'a.name': 'Anthony Edwards'}, {'a.name': 'Meg Ryan'}, {'a.name': 'Tom Cruise'}]}]
Final answer: Val Kilmer, Anthony Edwards, Meg Ryan, and Tom Cruise played in Top Gun.
```

Return direct results

You can return direct results from the Cypher QA Chain using the `return_direct` parameter

```
chain = GraphCypherQAChain.from_llm(
    ChatOpenAI(temperature=0), graph=graph, verbose=True, return_direct=True
)
```

```
chain.run("Who played in Top Gun?")
```

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
MATCH (a:Actor)-[:ACTED_IN]->(m:Movie {name: 'Top Gun'})
RETURN a.name

> Finished chain.
```

```
[{'a.name': 'Val Kilmer'},
 {'a.name': 'Anthony Edwards'},
 {'a.name': 'Meg Ryan'},
 {'a.name': 'Tom Cruise'}]
```