Assignment 3 - Applying Knowledge Graphs

The scope of this assignment is to evaluate and improve a LLM-based QA system which answers questions by transforming them into Cypher queries and executing them against a Neo4j knowledge graph (movie graph).

The assignment consists of 2 tasks, where the first task asks to build a small dataset of natural language questions that should be answerable by the Neo4j movie graph, feed them to the LLM-Based QA system and evaluate their generated Cypher queries, while the second task asks to improve the system by improving the Cypher generation prompt (zero-shot learning) and using question-query examples (few-shot learning).

In the next two chapters we will look at the steps and design decisions made in both tasks to achieve the desired results.

Task 1

For the implementation of this task, i created a new evaluation dataset of natural language questions in order to test the system's responses. The dataset is an Excel file (attached to the deliverable) and consists of 20 different questions which try to reveal the vulnerabilities of the system that tries to generate the appropriate Cypher query to answer the corresponding question each time.

The questions are illustrated in the table below.

Question
What is the release date for Snow Falling on Cedars?
What is the slogan of speed racer?
When is Al Pacino's birthday?
What is the actor of each role in Top Gun?
Which actors acted in Top Gun?
In how many movies has Keanu Reeves played in?
In how many movies Tom Cruise has not been in?

Question
What rating did James Thompson give to The Da Vinci Code?
Which movies have been directed by more than one director?
Which films have more than 3 actors?
Which directors have directed the most films?
Which persons are producers and also directors?
What is the max rating of each movie?
In the movies that Robin Williams has played in, who are the directors?
How many films released during the 21st century?
Which person is followed by more than 1 person but less than 3?
In which films are the writer and the director the same person?
Which critic has reviewed more than one film which made public in the 1990s?
Which movies have not been relased in 90s?
What are the two oldest movies that Tom Cruise hasn't been in?

After running tha LLM-Based QA system on these questions, i got the following results:

1. What is the release date for Snow Falling on Cedars?

Generated Cypher Query

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
WHERE m.title = 'Snow Falling on Cedars'
RETURN m.released
```

Cypher Query Result

```
[{'m.released': 1999}, {'m.released': 1999}, {'m.released': 1999}, {'m.released': 1999}]
```

Comment

Here the generated cypher query returns the correct release date of the 'Snow Falling on Cedars', but it is appeared as many times as the number of actors who played in it. There is no need to use the relation 'ACTED_IN' as the information from it is not necessary to answer the question. The correct query should return the release date once.

2. What is the slogan of speed racer?

Generated Cypher Query

```
MATCH (:Person)-[:ACTED_IN]->(m:Movie {title: "Speed Racer"})
RETURN m.tagline
```

Cypher Query Result

```
[{'m.tagline': 'Speed has no limits'}, {'m.tagline': 'Speed has no limits'}, {'m.tagline': 'Speed has no limits'}, {'m.tagline': 'Speed has no limits'}, {'m.tagline': 'Speed has no limits'}, {'m.tagline': 'Speed has no limits'}, {'m.tagline': 'Speed has no limits'}]
```

Comment

Although in the question the property 'tagline' is expressed by the word 'slogan' and also the title of the movie does not have capital letter for the first letter of each word (as it is in the KG), the system understands them correctly, but as in the previous question, the generated Cypher query returns many times the 'tagline' of the movie, which is not correct. Again, there is no need to use the 'ACTED_IN' relation as the information from it is not needed to answer the question. The correct query should return the 'tagline' once.

3. When is Al Pacino's birthday?

Generated Cypher Query

```
MATCH (p:Person {name: "Al Pacino"})
RETURN p.born
```

Cypher Query Result

```
[{'p.born': 1940}]
```

Comment

Here the generated Cypher query correctly returns the year in which Al Pacino was born.

4. What is the actor of each role in Top Gun?

Generated Cypher Query

```
MATCH (p:Person)-[r:ACTED_IN]->(m:Movie)
WHERE m.title = 'Top Gun'
RETURN p.name, r.roles
```

Cypher Query Result

```
[{'p.name': 'Val Kilmer', 'r.roles': ['Iceman']}, {'p.name': 'Meg Ryan', 'r.roles': ['Carole']}, {'p.name': 'Tom Skerritt', 'r.roles': ['Viper']}, {'p.name': 'Kelly McGillis', 'r.roles': ['Charlie']}, {'p.name': 'Tom Cruise', 'r.roles': ['Maverick']}, {'p.name': 'Anthony Edwards', 'r.roles': ['Goose']}]
```

Comment

Here the generated Cypher query is absolutely correct. The system understood that it must use the "ACTED_IN" relation and its properties to answer the question.

5. Which actors acted in Top Gun?

Generated Cypher Query

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
WHERE m.title = 'Top Gun'
RETURN p.name
```

Cypher Query Result

```
[{'p.name': 'Val Kilmer'}, {'p.name': 'Meg Ryan'}, {'p.name': 'Tom Skerritt'}, {'p.name': 'Kelly McGillis'}, {'p.name': 'Tom Cruise'}, {'p.name': 'Anthony Edwards'}]
```

Comment

Here the generated Cypher query does not have any problem and correctly returns the actors of the movie.

6. In how many movies has Keanu Reeves played in?

Generated Cypher Query

```
MATCH (:Person {name: "Keanu Reeves"})-[:ACTED_IN]->(m:Movie)
RETURN COUNT(m) as moviesPlayedIn;
```

Cypher Query Result

```
[{'moviesPlayedIn': 7}]
```

Comment

Here the generated Cypher query does not have any problem and correctly returns the number of movies where Keanu Reeves acted in based on the KG.

7. In how many movies Tom Cruise has not been in?

Generated Cypher Query

```
MATCH (p:Person {name: "Tom Cruise"})
MATCH (m:Movie)
WHERE NOT (p)-[:ACTED_IN]->(m)
RETURN COUNT(m)
```

Cypher Query Result

```
[{'COUNT(m)': 35}]
```

Comment

Here the generated Cypher query is absolutely correct. The system was able to understand the negation in the question and express it correctly in the Cypher query.

8. What rating did James Thompson give to The Da Vinci Code?

Generated Cypher Query

```
MATCH (p:Person {name: "James Thompson"})-[:REVIEWED]->(m:Movie {title: "The Da Vinci Code"})
RETURN p.name, m.title, m.rating
```

Cypher Query Result

```
[]
```

Comment

Here the generated Cypher query is wrong. It seems that the system struggles to understand what the query should do and how it should look when it needs to use properties of a relationship to answer a question, unlike question 4. In the correct query the 'REVIEWED' relationship should have a variable and in the return statement the 'rating' property should have this variable and not the varibale m of the Movie type.

9. Which movies have been directed by more than one director?

Generated Cypher Query

```
MATCH (m:Movie)
WITH m, size((:Person)-[:DIRECTED]->(m)) AS numDirectors
WHERE numDirectors > 1
RETURN m.title
```

Cypher Query Result

A syntax error raised

Comment

Here the Cypher query is wrong. The problem is that although it wants to correctly calculate all 'DIRECTED' relationships for each movie, the syntax that does so is incorrect. To count the relations with the 'size' function, this syntax "size([(p)-[:RELATIONSHIP]->(m) | p])" should be followed using the list comprehension systax.

10. Which films have more than 3 actors?

Generated Cypher Query

```
MATCH (m:Movie)
WHERE size((:Person)-[:ACTED_IN]->(m)) > 3
RETURN m.title
```

Cypher Query Result

A syntax error raised

Comment

Here the syntax of Cypher query has a problem which is exactly the same as in question 9.

11. Which directors have directed the most films?

Generated Cypher Query

```
MATCH (d:Person)-[:DIRECTED]->(m:Movie)
RETURN d.name, COUNT(m) AS numMovies
ORDER BY numMovies DESC
LIMIT 1
```

Cypher Query Result

```
[{'d.name': 'Lana Wachowski', 'numMovies': 5}]
```

Comment

Here the generated Cypher query does not return all the corrected answers to the question, as based on the KG there are two directors (Lana Wachowski, Lilly Wachowski) who have directed the most movies, which are 5. The query should always include all the answers corresponding to a question.

12. Which persons are producers and also directors?

Generated Cypher Query

```
MATCH (p:Person)-[:PRODUCED]->(:Movie)
MATCH (p)-[:DIRECTED]->(:Movie)
RETURN p.name
```

Cypher Query Result

```
[{'p.name': 'Cameron Crowe'}, {'p.name': 'Nora Ephron'}, {'p.name': 'Nora Ephron'}, {'p.name': 'Rob Reiner'}, {'p.name': 'Rob Reiner'}, {'p.name': 'Rob Reiner'}, {'p.name': 'Lana Wachowski'}, {'p.name': 'Lana Wachowski'}, {'p.name': 'Lana Wachowski'}, {'p.name': 'Lana Wachowski'}]
```

Comment

Here the logic of the Cypher query is right but it displays multiple times each person who is producer and also director. When the same value does not need to be displayed multiple times, the query should return the distinct/unique values.

13. What is the max rating of each movie?

Generated Cypher Query

```
MATCH (m:Movie)
RETURN m.title, max((m)<-[:REVIEWED]-(:Person).rating) as max_rating
```

Cypher Query Result

A syntax error raised

Comment

Here the generated Cypher query is completely wrong. It is trying to compute the max rating of each movie but with using wrong syntax. Also, it does not understand that the 'rating'is property of the 'REVIEWED' relationship.

14. In the movies that Robin Williams has played in, who are the directors?

Generated Cypher Query

```
MATCH (p:Person {name: "Robin Williams"})-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(d:Person)
RETURN d.name as Director, m.title as MovieTitle
```

Cypher Query Result

```
[{'Director': 'Vincent Ward', 'MovieTitle': 'What Dreams May Come'}, {'Director': 'Chris Columbus', 'MovieTitle': 'Bicentennial Man'},
{'Director': 'Mike Nichols', 'MovieTitle': 'The Birdcage'}]
```

Comment

Here the generated Cypher query is correct and returns the actual directors.

15. How many films released during the 21st century?

Generated Cypher Query

```
MATCH (m:Movie)
WHERE m.released >= 2000 AND m.released < 2100
RETURN COUNT(m) as num_films;
```

Cypher Query Result

```
[{'num_films': 15}]
```

Comment

Here the generated Cypher query is correct. The system understood the time range and expressed it in the query.

16. Which person has more than 1 follower but less than 3?

Generated Cypher Query

```
MATCH (p:Person)-[:FOLLOWS]->(followed)
WITH p, COUNT(followed) AS numFollowers
WHERE numFollowers > 1 AND numFollowers < 3
RETURN p.name
```

Cypher Query Result

```
[]
```

Comment

Here the generated Cypher query is wrong. It should calculate the number of p's for each follower, not the other way around as it does. One possible reason for this is that the system cannot fully distinguish what a person following someone means and what a follower of someone also means. The correct answer based on the KG is the person 'Jessica Thompson'.

17. In which films are the writer and the director the same person?

Generated Cypher Query

```
MATCH (p:Person)-[:WROTE]->(m:Movie)<-[:DIRECTED]-(p) RETURN m.title
```

Cypher Query Result

```
[{'m.title': 'Jerry Maguire'}, {'m.title': 'Speed Racer'}, {'m.title': 'Speed Racer'}, {'m.title': "Something's Gotta Give"}]
```

Comment

Here the generated Cypher query is correct and the only thing that needs to be entered is the return of the discrete values, as there is no need to display the values multiple times. This mistake appeared also in the questions 1,2 and 12.

18. Which critic has reviewed more than one film which made public in the 1990s?

Generated Cypher Query

```
MATCH (p:Person)-[:REVIEWED]->(m:Movie)
WHERE m.released >= 1990 AND m.released <= 1999
WITH p, COUNT(m) AS numReviews
WHERE numReviews > 1
RETURN p.name as Critic, numReviews as NumReviews
```

Cypher Query Result

```
[{'Critic': 'Jessica Thompson', 'NumReviews': 2}]
```

Comment

Here the generated Cypher query is absolutely correct. The system was able to understand that a critic is a reviewer, the phrase 'made public' refers to the release date of movies and the time range.

19. Which movies have not been released in 90s?

Generated Cypher Query

```
MATCH (m:Movie)
WHERE NOT m.released >= 1990 AND m.released < 2000
RETURN m.title
```

Cypher Query Result

```
[{'m.title': "One Flew Over the Cuckoo's Nest"}, {'m.title': 'Top Gun'}, {'m.title': 'Stand By Me'}]
```

Comment

Here the generated Cypher query is incorrect. Although, the system was able to understand the negation in the question, it had to put this code "m.released >= 1990 AND m.released < 2000" in parentheses in order to be considered as one in the NOT operator.

20. What are the two oldest movies that Tom Cruise hasn't been in?

Generated Cypher Query

```
MATCH (m:Movie)
WHERE NOT (m)<-[:ACTED_IN]-(:Person {name: "Tom Cruise"})
RETURN m.title, m.released
ORDER BY m.released ASC
LIMIT 2
```

Cypher Query Result

```
[{'m.title': "One Flew Over the Cuckoo's Nest", 'm.released': 1975}, {'m.title': 'Stand By Me', 'm.released': 1986}]
```

Comment

Here the generated Cypher query is absolutely correct. The system was able to understand the negation in the question and express it correctly in the Cypher query. It also understood that it must first found the movies in which Tom Cruise has been in and then subtracted them from the movies being considered.

Based on the system's results on the questions, the most common mistakes LLM makes when translating natural language questions into Cypher queries are:

1. The system sometimes creates queries that include relationships between nodes that do not need to be used to answer certain questions.
2. The system sometimes creates queries that return duplicate results because they do not use the 'DISTINCT' keyword where it is needed.

3. Sometimes the Cypher query needs to access properties of a relationship to answer certain questions, but the system cannot understand it.
4. It seems that the system cannot understand that to use the properties of a relationship, it must first define a variable for it before it can access its properties.
5. The system sometimes fails to account for cases where multiple values should be returned due to ties. In these cases, the query needs to ensure that it captures all relevant results rather than just one.
6. The system fails to use a proper syntax to calculate max/min values.
7. The Cypher queries sometimes do not include the necessary parentheses and can lead to issues with operator precedence, resulting in incorrect queries.
8. The system sometimes cannot explicity understand the meaning of a relationship, in our dataset the relationship 'FOLLOWS', which can lead to wrong results.
9. Sometimes the queries do not use the correct syntax, especially when they need to count the instances of a relationship with the function 'size'.

Task 2

Based on the most common mistakes LLM makes when trying to convert natural language queries of the dataset into Cypher queries, I tried to improve the system by improving the Cypher generation prompt using one-shot and few-shot learning.

My goal was to find and include the most appropriate zero and few shot content in the prompt, so that the system minimizes the above errors and produces better Cypher queries. After doing a lot of testing to see which instructions and examples fit best, I came up with the following:

1. The system sometimes creates queries that include relationships between nodes that do not need to be used to answer certain questions.

This error occurred in questions 1 and 2, and to solve it I decided to put in a simple guideline and an example. In the example the question asks something that doesn't need to use a relationship to be answered.

```
- Use a relationship only when it is needed to answer a question.

Example 1:
Question: What are the names of all persons?
Cypher Query: MATCH (p:Person) RETURN p.name
```

2. The system sometimes creates queries that return duplicate results because they do not use the 'DISTINCT' keyword where it is needed.

This error occurred in questions 1,2,12 and 17, and to solve it I decided to add an instruction that if the question does not ask for duplicate values, then only the unique ones should be shown.

```
- Use the 'DISTINCT' keyword to keep only unique values. Keep duplicates only when the question
explicitly asks for them.
```

3. Sometimes the Cypher query needs to access properties of a relationship to answer certain questions, but the system cannot understand it.
4. It seems that the system cannot understand that to use the properties of a relationship, it must first define a variable for it before it can access its properties.

To resolve these errors, I used the same pattern. The errors occurred in questions 8 and 13 and to solve them I decided to include an instruction and an example in promt so that the system would fully understand how to behave in such cases.

```
- Assign a variable to a relationship before accessing its properties (e.g. '[p:Person]-
[r:REVIEWED]->[m:Movie] ... RETURN r.rating, r.summary', '[p:Person]-[r:ACTED_IN]->[m:Movie] ...
RETURN r.roles').

Example 2:
Question: What are the roles in Birdcage?
Cypher Query: MATCH (p:Person)-[r:ACTED_IN]->(m:Movie) WHERE m.title = "The Birdcage" RETURN
r.roles
```

5. The system sometimes fails to account for cases where multiple values should be returned due to ties. In these cases, the query needs to ensure that it captures all relevant results rather than just one. 6. The system fails to use a proper syntax to calculate max/min values.

These errors are somehow the same based on the reponses on the questions and to resolve them i used them same pattern. Occured in questions 11 and 13, and to solve them i provided an instruction and also two examples as follows.

```
- Return all entities when there are multiple with the same highest or lowest value. Avoid using
LIMIT 1 in such cases.
```

```
Example 3:
Question: Which actors have been in the most movies?
Cypher Query: MATCH (p:Person)-[:ACTED_IN]->(m:Movie) WITH p, COUNT(m) as total_movies WITH
MAX(total_movies) as max_movies MATCH (p:Person)-[:ACTED_IN]->(m:Movie) WITH p, COUNT(m) as
total_movies, max_movies WHERE total_movies = max_movies RETURN p.name

Example 4:
Questions: Who is the oldest actor?
Cypher Query: MATCH (p:Person)-[:ACTED_IN]->(m:Movie) WITH MAX(p.born) as max_born MATCH
(p:Person)-[:ACTED_IN]->(m:Movie) WITH p, max_born WHERE p.born = max_born RETURN p.name
```

I have provided two examples to help the system fully understand the cases in which we need to do a grouping before finding the max/min and the cases in which this is not necessary. Both, examples also return all instances with the 'max' value and not only one.

7. The Cypher queries sometimes do not include the necessary parentheses and can lead to issues with operator precedence, resulting in incorrect queries.

This error appeared in question 19 and to solve it, I added the following guideline and example to promt.

```
- Always use parentheses to group conditions and ensure the precedence of operations and
operators, especially with logical operators (AND/OR).

Example 5:
Question: Which persons were born between 1950 and 2000?
Cypher Query: MATCH (p:Person) WHERE (p.born >= 1950 AND p.born <= 2000) RETURN p.name
```

8. The system sometimes cannot explicity understand the meaning of a relationship, in our dataset the relationship 'FOLLOWS', which can lead to wrong results.

This error occurred only in question 16, and in order to teach the system how to behave in such cases, i tried to explain how the concepts of 'following' and 'follower' relate to the 'FOLLOWS' relationship.

```
- For finding the followers of a person use the incoming FOLLOWS relationships, otherwise use the
outgoing FOLLOWS relationships.
```

9. Sometimes the queries do not use the correct syntax, especially when they need to count the instances of a relationship with the function 'size'.

This error occurred in questions 9 and 10 and refers to the incorrect syntactic use of the size function when counting the instances of a relationship. In these cases Cypher's question should use it using list comprehension and to teach the system this, I have provided an example of using this function so that if the system needs to use it, it will use it with the correct syntax.

```
Example 6:
Question: Which persons follows only one person?
Cypher Query: MATCH (p:Person) WHERE size([(p)-[:FOLLOWS]->(n:Person) | n]) = 1 RETURN p.name
```

In the notebook you can see the system answer for each question and the improved and accurate version of each Cypher query based on the enhanced Cypher generation prompt.

Here are the final Cypher queries for each question that had a problematic query based on the initial Cypher generation prompt.

1. What is the release date for Snow Falling on Cedars?

Improved

```
MATCH (m:Movie {title: "Snow Falling on Cedars"})
RETURN m.released
```

It does not use the 'ACTED_IN' relationship and returns only the unique values.

2. What is the slogan of speed racer?

Improved

```
MATCH (m:Movie {title: "Speed Racer"})
RETURN m.tagline
```

It does not use the 'ACTED_IN' relationship and returns only the unique values.

8. What rating did James Thompson give to The Da Vinci Code?

Improved

```
MATCH (p:Person {name: "James Thompson"})-[r:REVIEWED]->(m:Movie {title: "The Da Vinci Code"})
RETURN r.rating
```

The system understood the it must use the REVIEWED relationship and its property to answer the question.

9. Which movies have been directed by more than one director?

Improved

```
MATCH (p:Person)-[:DIRECTED]->(m:Movie)
WITH m, COUNT(p) as total_directors
WHERE total_directors > 1
RETURN m.title
```

The system realised that the way it was using the size function was not the best way. It decided to use another way of translating the question, which is absolutely correct.

10. Which films have more than 3 actors?

Improved

```
MATCH (m:Movie)<-[:ACTED_IN]-(p:Person)
WITH m, COUNT(p) as total_actors
WHERE total_actors > 3
RETURN m.title
```

The system realised that the way it was using the size function was not the best way. It decided to use another way of translating the question, which is absolutely correct.

11. Which directors have directed the most films?

Improved

```
MATCH (p:Person)-[:DIRECTED]->(m:Movie)
WITH p, COUNT(m) as total_movies
WITH MAX(total_movies) as max_movies
MATCH (p:Person)-[:DIRECTED]->(m:Movie)
```

```
WITH p, COUNT(m) as total_movies, max_movies
WHERE total_movies = max_movies
RETURN p.name
```

The system realized that there are additional directors who have directed the most films and decided to properly create a new query that includes them all.

12. Which persons are producers and also directors?

Improved

```
MATCH (p:Person)-[:PRODUCED]->(:Movie)
WITH p
MATCH (p)-[:DIRECTED]->(:Movie)
RETURN DISTINCT p.name
```

The system correctly understood that has to display only the unique values.

13. What is the max rating of each movie?

Improved

```
MATCH (m:Movie)<-[r:REVIEWED]-(p:Person)
WITH m, MAX(r.rating) as max_rating
RETURN m.title, max_rating
```

The system correctly understood that it has to use the rating property of the 'REVIEWED' relationship to answer the question and also adopted a better way to calculate the max rating.

16. Which person has more than 1 follower but less than 3?

Improved

```
MATCH (p:Person)<-[:FOLLOWS]-(n:Person)
WITH p, COUNT(n) as followers
WHERE followers > 1 AND followers < 3
RETURN p.name
```

The system understood the correct meaning of 'FOLLOWS' and what the terms 'follower' and 'following' mean.

17. In which films are the writer and the director the same person?

Improved

```
MATCH (p:Person)-[:WROTE]->(m:Movie)
MATCH (p)-[:DIRECTED]->(m)
RETURN DISTINCT m.title
```

The system returns correctly only the unique values.

19. Which movies have not been released in 90s?

Improved

```
MATCH (m:Movie)
WHERE NOT (m.released >= 1990 AND m.released <= 1999)
RETURN m.title
```

The system correctly added parentheses to the conditions to ensure the priority of the operators.