Question Answering and Chatbots

5.1th Practical exercise – QA-system

Aleksandr Perevalov

aleksandr.perevalov@hs-anhalt.de

October 4, 2021

Hochschule Anhalt
Anhalt University of Applied Sciences

Plan for today

Plan for today

- Review the task for the Exercise 5.1;

Plan for today

- Review the task for the Exercise 5.1;

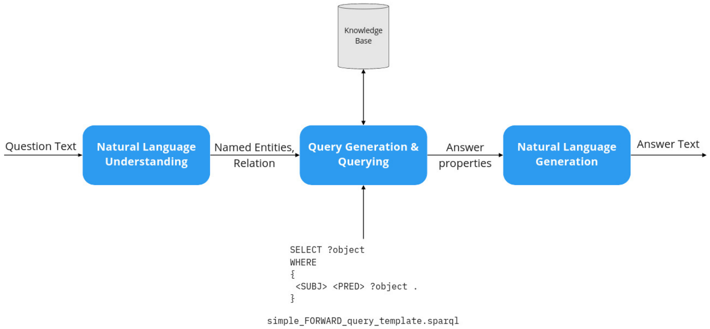
Plan for today

- Review the task for the Exercise 5.1;
- Demo Session;

Plan for today

- Review the task for the Exercise 5.1;
- Demo Session;
- Introduction to the Exercise 5.2 (Qanary Framework).

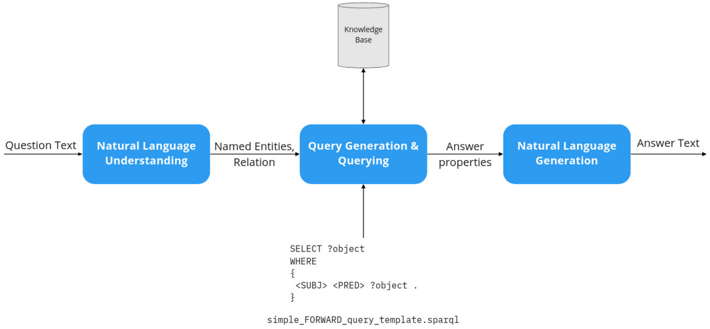
Exercise 5.1 – the Task


Knowledge Base
Question Text
Natural Language Understanding
Named Entities, Relation
Query Generation & Querying
Answer properties
Natural Language Generation
Answer Text

```
SELECT ?object
WHERE
{
 <SUBJ> <PRED> ?object .
}
```

simple_FORWARD_query_template.sparql

Exercise 5.1 – the Task

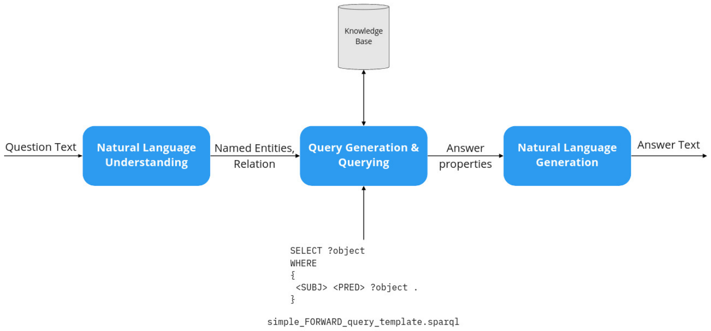

Knowledge Base
Question Text
Natural Language Understanding
Named Entities, Relation
Query Generation & Querying
Answer properties
Natural Language Generation
Answer Text

```
SELECT ?object
WHERE
{
 <SUBJ> <PRED> ?object .
}
```

simple_FORWARD_query_template.sparql

- Re-train relation prediction model based on the new data (see repository);

Exercise 5.1 – the Task

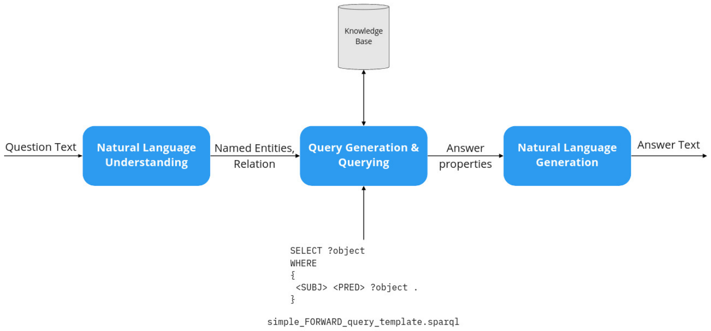

Knowledge Base
Question Text
Natural Language Understanding
Named Entities, Relation
Query Generation & Querying
Answer properties
Natural Language Generation
Answer Text

```
SELECT ?object
WHERE
{
 <SUBJ> <PRED> ?object .
}
```

simple_FORWARD_query_template.sparql

- Re-train relation prediction model based on the new data (see repository);
- Implement a NLG component based on templates. You have to create at least one template for every relation from the new data;

Exercise 5.1 – the Task

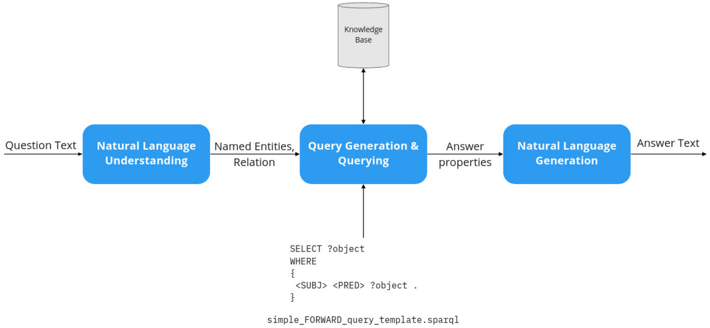

Knowledge Base
Question Text
Natural Language Understanding
Named Entities, Relation
Query Generation & Querying
Answer properties
Natural Language Generation
Answer Text

```
SELECT ?object
WHERE
{
 <SUBJ> <PRED> ?object .
}
```

simple_FORWARD_query_template.sparql

- Re-train relation prediction model based on the new data (see repository);
- Implement a NLG component based on templates. You have to create at least one template for every relation from the new data;
- Combine all the components together as it's shown in the Figure above;

Exercise 5.1 – the Task


Knowledge Base
Question Text
Natural Language Understanding
Named Entities, Relation
Query Generation & Querying
Answer properties
Natural Language Generation
Answer Text

```
SELECT ?object
WHERE
{
 <SUBJ> <PRED> ?object .
}
```

simple_FORWARD_query_template.sparql

- Re-train relation prediction model based on the new data (see repository);
- Implement a NLG component based on templates. You have to create at least one template for every relation from the new data;
- Combine all the components together as it's shown in the Figure above;
- Connect your Frontend and Backend such that your system can work in Question-Answer mode and prepare some questions for the Demo session.

Exercise 5.1 – typical remarks

Exercise 5.1 – typical remarks

- Group Python files into several modules (folders);

Exercise 5.1 – typical remarks

- Group Python files into several modules (folders);
- Group other files to the related folders: all datasets to data/, all models to bin/ etc.;

Exercise 5.1 – typical remarks

- Group Python files into several modules (folders);
- Group other files to the related folders: all datasets to data/, all models to bin/ etc.;
- Files and methods naming: usually in Python methods and files are `named_like_this`, `notLikeThis` and `Not_This`. Not very important issue;

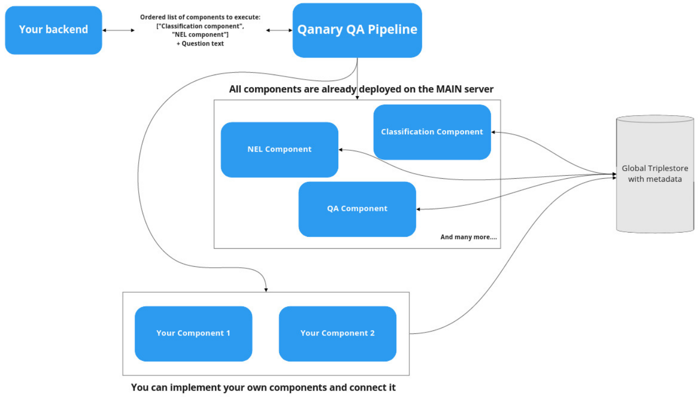
Exercise 5.1 – typical remarks

- Group Python files into several modules (folders);
- Group other files to the related folders: all datasets to data/, all models to bin/ etc.;
- Files and methods naming: usually in Python methods and files are `named_like_this`, `notLikeThis` and `Not_This`. Not very important issue;
- Save the models into binary files. Load models when the application is starting.

Any questions?

Let's start the demo!

Qanary Framework


Your backend
Ordered list of components to execute: ["Classification component", "NEL component"] + Question text
Qanary QA Pipeline
All components are already deployed on the MAIN server
NEL Component
Classification Component
QA Component
And many more....
Global Triplestore with metadata
Your Component 1
Your Component 2
You can implement your own components and connect it

+ You can reuse any component (even from other people);
+ No need to reinvent the wheel;
– Requires more effort to invest at the beginning.

Plan for the Exercise 5.2: Qanary Integration

- Split your QA system into separate components;
- Define configuration parameters in app.conf file for each component;
- Implement your components. The example of implementation is located within classifier.py in the example;
- Define the data that you would like to save to the triplestore w.r.t your components and adjust the SPARQL insert query;
- Change your backend as it shown in the example (controllers.py) such that it will run Qanary Pipeline and retrieve the final answer;
- Start all components. Check its status here;
- If all components are working correctly start your QA system and test it.

? Try to reuse components – we will do it later.

1. SPARQL;
2. Work with Natural Language (NER);
3. Question classification;
4. Back-end and Front-end;
5. **Simple QA system and Qanary Framework**;
6. Tests for QA system;
7. Deploying QA system;
8. ...