Question Answering and Chatbots

4th Practical exercise – Frontend and Backend

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 4;

Plan for today

- Review the task for the Exercise 4;
- A little bit of theory;

Plan for today

- Review the task for the Exercise 4;
- A little bit of theory;
- Demo Session;

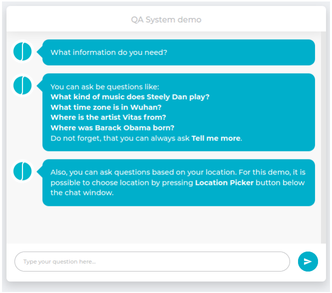
Plan for today

- Review the task for the Exercise 4;
- A little bit of theory;
- Demo Session;
- Introduction to the Exercise 5.

Exercise 4 – the Task

Implement a **Frontend (Client)** side of a QA system. Possible ways:

- Web page – implement by your own or take template;
- Messenger integration – use API.

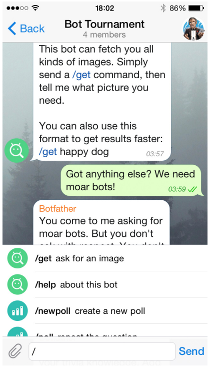


What information do you need?

You can ask be questions like:
What kind of music does Steely Dan play?
What time zone is in Wuhan?
Where is the artist Vitas from?
Where was Barack Obama born?
Do not forget, that you can always ask **Tell me more**.

Also, you can ask questions based on your location. For this demo, it is possible to choose location by pressing **Location Picker** button below the chat window.

Figure: Web interface



This bot can fetch you all kinds of images. Simply send a /get command, then tell me what picture you need.

You can also use this format to get results faster: /get happy dog 03:57

Got anything else? We need moar bots! 03:59

Botfather
You come to me asking for moar bots. But you don't

/get ask for an image
/help about this bot
/newpoll create a new poll

Figure: Messenger interface

Exercise 4 – the Task

Implement a **Backend (Server)** side of a QA system. The server has to support 2 HTTP methods:

- Type: `GET`, Name: `health`, Returns: {`'status': 'OK'`};
- Type: `POST`, Name: `question`, Params: `question_text`, Returns: `dictionary` (see GitHub).

Process the questions via the `question` method according to your variant and save it to the `output.json` file.

Sometimes questions are not compatible to your Relation Classifier model. Hence, the training data needs to be extended (reference to the next exercise).

Exercise 4 – the Task

Connect **Frontend** and **Backend** together. Show how it works.

Any questions?

Let's start the demo!

Architecture of a QA system

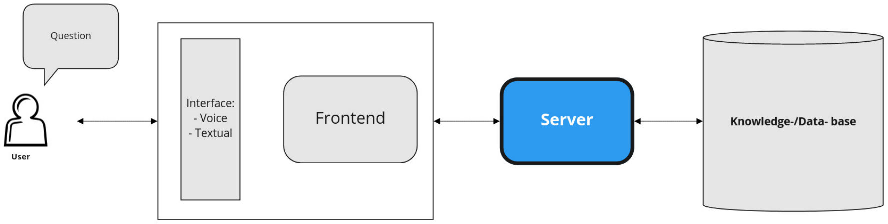
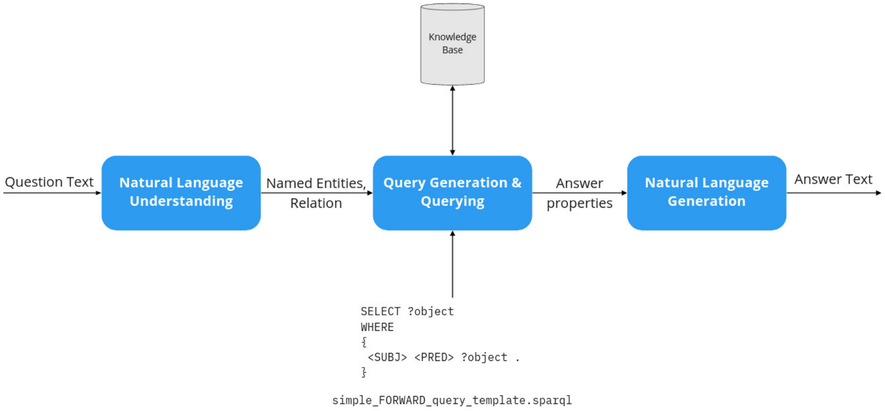


Figure: QA System Conceptual Architecture

Architecture of a QA system

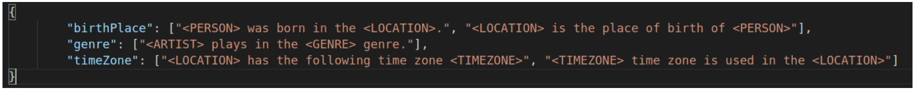

Knowledge Base
Question Text
Natural Language Understanding
Named Entities, Relation
Query Generation & Querying
Answer properties
Natural Language Generation
Answer Text

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

simple_FORWARD_query_template.sparql

Figure: QA system Server Architecture

Architecture of a QA system

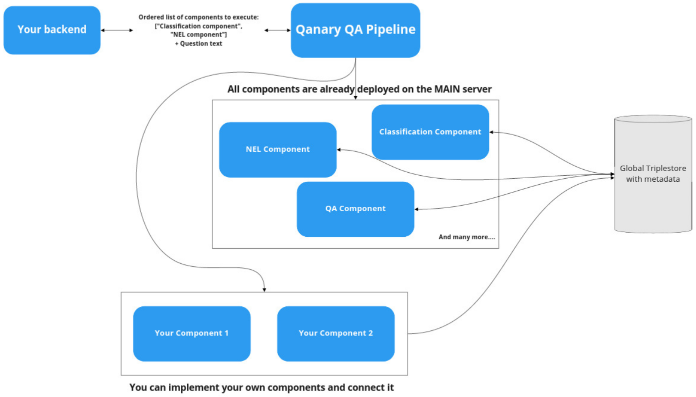


```
{
    "birthPlace": ["<PERSON> was born in the <LOCATION>.", "<LOCATION> is the place of birth of <PERSON>"],
    "genre": ["<ARTIST> plays in the <GENRE> genre."],
    "timeZone": ["<LOCATION> has the following time zone <TIMEZONE>", "<TIMEZONE> time zone is used in the <LOCATION>"]
}
```

Figure: A possible way of storing the answer templates

1. The corresponding template is chosen randomly;
2. The placeholders are replaced with the entities.

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
Classification Component
NEL 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: QA System

- Retrain relation prediction component on the new dataset;
- Implement simple NLG component;
- Combine all previously created components within your QA system;
- Connect Frontend and Backend;
- Integrate Qanary Framework in your system;

? Try to reuse components.

The final task with all examples and data will be published in 1-2 days.

The submission process will contain 2 steps:

1. Send your code and show the demo without Qanary;
2. Integrate Qanary and do previous step.

We will spend 2 weeks on this task.

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. ...