Final Report - Project Group Data Extraction, Search, Analysis and Benchmarking

Kevin Haack(7094226)

Paderborn University
khaack@mail.uni-paderborn.de

1 Task definition

The higher-level task of our subgroup, is to create a platform for knowledge extraction. This platform should be a webbased application based on different knowledge extraction tools. It should cooperate with various DICE projects and should unify the access to them. We should be able to convert information datasets to a RDF knowledge graphs. This grpahs should be used from other subgroups and other tools of the DICE group.

Over both semesters, it was my job to lead the subgroup, assign tasks and provide assistance for all subgroups. Like most projects, this one is divided into three phases: define and document the requirements, the implementation and the integration. For me, the first semester began with the requirement engineering and the design of the entire software architecture. Before we could start the development process, all group members had to be brought to a stand first. For the first weeks, André and I made sure that all the members of the group mastered the necessary technologies. At the same time, it was my responsibility with André to create the basic architecture. After that and in the second semester my main task was limited to creating the UI and provide general support to all and especially for the weaker group members.

2 Technical description

In order to be as flexible as possible in the future, we opted for a microservice architecture. We used the Spring Boot framework with the Netflix Eureka-Registry-Server. With these microservices we also wanted to create a unified access to external system. And during the project, we also introduced the SASK commons as a subproject, for classes that are used across the microservices.

2.1 Repository

As first persistent component we use the open source project Hadoop. Hadoop is software for storing data in a cluster of distributed systems. In our case, we only use one server, but with this solution we offer an easy-to-scale system for the future. The Hadoop API did not support Java9 at the time, so we had to implement the access the Hadoop interfaces by ourselves. This was the main task of André, I supported him to configure hadoop at initial in a docker container. We implemented the access to the haddop system in the *repo-ms*, currently this facade supports all file and folder operations and offers these to the other microservices.

2.2 Database

The second persistent component, next to Hadoop, is a Fuseki server. Apache Jena Fuseki is a SPARQL server, in our case its setup in a docker container. It uses a TDB and provides a REST interface for updating and query the database. This was the main task of Suganya and Sepide in the first semester. They implemented the *database-ms* as a facade, to provide a simplified access to the database and ensure that other systems are informed about changes centrally. I support the two in programming the microservice.

2.3 Workflow

To give the user the flexibility to decide how to handle his task. We give him the opportunity to create workflows on the UI. Workflows consist of operators and links. An operator can be a file, an extractor or a target graph. And the user is able to create a dataflow with links.

2.4 Executer

The executer is the central unit for working with workflows. In the first semester André and I implemented the first version of the *executer-ms*. We were able to execute the passed workflows from the UI. In this first restricted version we execute simple workflows with one file, one extractor and one target graph. In the second semester we implemented the support for more complex workflows, with multiple files or extractors. The executer creates automatically threads for every operator in the workflow, so that a parallel execution is possible.

2.5 User interface

The *webclient* microservice provides simple HTML files and on the client side we rely on popular technologies such as JQuery and Bootstrap. We implemented JQuery plugins, to keep the source code as clear as possible. In the first semester I already created the almost finished UI. The user was able to create workflows, upload files, manage the content of the hadoop system and could see the status of each microservice. Later, we only expanded the UI. We integrated the chatbot, were able to visualize a selected target graph and extended the creation of the workflows.

2.6 Logging

To implement a centralized logging solution we setup the ELK stack. Consist of Elasticsearch, Logstash, and Kibana. All three are delivered in preconfigured docker containers. Elasticsearch is a search engine. Logstash is a server-side pipeline that collects the log data from the sources and sends it to Elasticsearch. Kibana visualize the data from Elasticsearch. Each microservices writes its log into a special directory, from which Logstash retrieves the logs. This was the main task of Suganya and Sepide in the second semester and I supported them in the configuration of the docker containers and the implementation of the changes in the microservices.

2.7 Continuous deployment

Responsibilities changed shortly before the end of the project group. We developed the ability to install each microservices with one command in a Docker image. In the CI these should then be pushed and later pulled on the staging machine. We implement build profiles and more flexible configuration files. Before we could finish the task, the responsibilities changed again. But our work served as preparation for the other group.

3 Learned skills

During the project I have solidified my knowledge in many technologies, such as Docker, Maven, Travis CI, git, Hadoop, Bootstrap, JQuery and Spring-Boot with Eureka. I've improved my ability to lead a team and to teach other developer in basic development techniques. From a scientific point of view, I learned a lot about about named entity recognition and relation extraction.

4 Self-evaluation

Throughout the project, it has always been my job to help the weaker group members. Since the hardest parts of the development were completed after the first semester, I put in the second semester even more time in this support. Already in the first week it turned out that it was a good decision. The most of the weaker group members have learned something about software development and independent work. In my estimation, some group members were able to improve their way of working. I also think that working as a team leader has worked well in almost all cases. I was able to help all the group members who were looking for help and were motivated. By telephone, I was always available for the group members. So I was able to provide support even outside the meetings. In this way others in the group could work without interruption.
All in all, I think that under my leadership and with my help, some members have greatly improved their skills. And even the project had some good improvements in the last semester.