# Contribution Title

Lukas Blübaum, Nick Düsterhus, and Monika Werner

University of Paderborn, Padeborn 33098, Germany

**Abstract.** The goal was to create a software that is able to extract information from a Wikipedia dump and represent the information in a knowledge graph. In order to achieve this, our software uses the given systems and frameworks in each step of the process to make the resulting graph as accurate and complete as possible.

**Keywords:** Knowledge Graphs · Semantic Web.

## 1 The Task

The task given to us was to create a programm that can create a knowledge graph, agraph that represents entities as nodes and relations between those entities as edges, from a large text corpus, in this case a Wikipedia dump, in order to learn the basics of Semantic Web technologies. We should learn how to extract, aggregate and present information in scalable ways. To facilitate this task we were introduced to several tools including FOX and DBpedia Spotlight which are capable of handling parts of the task on their own. To show how well our programm works, its results were compared to the results from DBpedia.

## 2 The Process

There are several steps that need to be taken in order to create a knowledge graph from a given text which will be demonstrated on the following small example text:

*Example 1.* Obama was born on August 4, 1961, at Kapiolani Medical Center for Women and Children in Honolulu, Hawaii. He graduated from Havard University.

### 2.1 Preprocessing

**Extraction and Cleaning of the Text from a Wikipedia Dump** The Wikipedia dumps that the software works on come as plaintexts without markup, infoboxes or chapter subdivison but they still contain elements that are undesired for later steps in the process. Such elements include unnecessary URLs, paranthesis, whitespace, nullchar and certain symbols. In order to get rid of these unwanted elements, we created a class called WikiCleaner that reads through the dump, recognizes the unwanted elements, removes them and writes the resulting text into an output file for usage in later steps. The result is a simple text

containing only the letters from the English alphabet, numbers, periods and of course simple spaces.

In order to speed up the cleaning process, the WikiCleaner uses Reader and Writer threads for reading the input file and writing the output file respectivly.

This step was the easiest part of the entire process and uses only standard Java libraries. There were no difficulties during the implemention of this step.

**Coreference Resolution** Another step that needs to be taken before the information can be properly extracted is replacing all the pronouns with the corresponding noun it refers to. This is done so that the process can find all he information it needs in the current sentence and doesn't have to look back on previous parts of the text in order to understand the pronouns.

For this task we use the Stanford CoreNLP in order to split the text into tokens (roughly "words", see x) an check each token whether it is a pronoun or not. If it is a pronoun, we replace it with the last mentioned noun that matches its grammatical gender.

After both steps of te preprocessing are done, our text example looks as follows:

*Example 2.* Obama was born on August 4 , 1961 , at Kapiolani Medical Center for Women and Children in Honolulu , Hawaii . Obama graduated from Havard University.

This is still a relative easy part after unerstanding and learning how to use Stanford NLP properly.

### 2.2 Named Entity Recognition

In this part of the process we scan through the cleaned text to identify named entities like people, locations etc. which will be the nodes of our graph later on.

For this step we use the DBpedia Spotlight web service. We send the cleaned text to DBpedia Spotlight and receive a JSON response.The response is then parsed and the found entities and their information is stored in objects of our Entity class.

From our example text, the word that are recognized as named entitis are those that are underlined in the following:

*Example 3.* Obama was born on August 4, 1961, at Kapiolani Medical Center for Women and Children in Honolulu, Hawaii. Obama graduated from Havard University .

As the process of entity recognition is mainly done by DBpedia Spotlight our main task in this step is simply to send the data and deal with the response. Therefore, implementing this step was a bit more demanding then the imlementation of the first but still not particuary difficult.

### 2.3 Entity Disambiguation

After finding the named entities we need to recognize which entities are actually refered to by the named entities found in the Wikipedia article. There are many people, places etc. sharing the same or similar names so we need to find out which entity was ment by the article in order to build the correct relations between them later on.

This step is partially already done with DBpedia Spotlight, however, in addition to it we also use FOX. Again, we simply sent our data and parse and save the response for later steps.

As this is completely done by the given frameworks, we did not have to add much.

### 2.4 Relation Extraction

After recognizing the entities correctly we need to find the relations between them mentioned in the given Wikipedia dump to finish our knowledge graph. The relation become the edges of the graph.

The type of relation we are most interested in are binary relations, so relations between only two entities an not more. Therefore, we need to find those first. The relations of this type that can be found in our example include:

*Example 4.* First sentence: Obama - be bear on - August 4 1961. (among others)
Second sentence: Obama - graduate from - Havard Law school (among others)

Afterwards, we add the relation to our graph. In order to do so, we first look at the subject of the binary relation. In both sentences from our example we recognize "Obama" as the subject with its type including "Person". Then we look at the object of the relation. In the first sentence, we detect that the object contains numbers, which can indicate that the object contains either a date or a number literal. We check whether the object is a date and if it is not, we extract it as a single literal of a fitting number type if it is a number or keep it as a string if the number just happens to be part of a name and therefore not convertable to any other type of literal. In our case, we detect "August" is a month and therefore convert the whole object to the proper dateformat 1961-08-04. In the second sentence, we detect "Hardvard University" as another entity of type EducationalInstitution. Then, in order to be able to map the found relation to the correct property, we look at the predicate of the sentences and check for keywords. In our first sentence, we find "bear" in a relation with "Person" as the domain and xsd:date as range so we search for a property which fits this keyword, domain and range and get dbo:birthDate, while in our second sentence we find the keyword "graduate" with "Person" as domain and " EducationalInstitution" as range and get dbo:almaMater as the fitting property. With these informations, we turned the first and second sentence of our example into the following triples:

*Example 5.* First sentence: dbo:Barack_Obama dbo:birhtDate "1961-08-04".
Second sentence: dbo:Barack_Obama dbo:almaMater dbo:Havard_University.

These triples can now be properly added to our knowledge graph. After this has been done with all found triples our knowledge graph from the given text is as far completed as the programm is capable to do so.

For this step again we use FOX and Spotlight. We use FOX in "re" mode for entity recognition and relation extraction, however FOX does not recognize many triples and misses too many relations. Therefore we implemented our own method using Spotlight and OpenIE. We use DBdia Spotlight first for named entity recognition so that we can run our method concurrent to FOX.

## 3 Correctness and Runtime

## References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017