Building a Knowledge Graph

Lukas Blübaum, Nick Düsterhus, and Monika Werner

University of Paderborn, Padeborn 33098, Germany
{lukasbl,nduester,mwerner}@uni-paderborn.de
https://github.com/LukasBluebaum/BKG

Abstract. The goal of this proseminar was to create an encyclopetic knowledge graph from a large text corpus like a wikipedia dump. In order to learn and better understand the basics of Semantic Web technologies. For this we were given a wikipedia dump and the core ontology of DBpedia. To facilitate this task we were introduced to several tools including FOX [7] and DBpedia Spotlight [1]. The whole project is written in Java and as license we used the GNU General Public License v3.0. The documented source code, including instructions on how to run the code, can be found on the given Github repository.

Keywords: Knowledge Graphs · Semantic Web · Coreference Resolution · Named Entity Recognition · Entity Disambiguation · Relation Extraction · FOX · Spotlight

1 Introduction

Websites like Wikipedia contain a large volume of information. Unfortunately this information is mainly unstructured, for example in the form of natural language text. So the detection of complex informationen can be quite difficult. One approach to structure this knowledge is the extraction of triples from the natural language text. These triples consist of a subject, predicate and object and can then be displayed as a knowledge graph. The creation of such a knowledge graph is the main goal of this project.

To carry out this projekt several steps had to be taken. At first we had to process the given wikipedia dump and the core ontology. For the wikipedia dump this meant cleaning the text and for the core ontology we filtered some not required information. The next steps included Named Entity Recognition, Entity Disambiguation and Relation Extraction. As frameworks for the Named Entity Recognition and Entity Disambiguation we used FOX and Spotlight. For the Relation extraction we implemented to approaches. The first approach takes the sentences and sends them to FOX in "re" mode to directly extract the triples. The second one, our own approach, uses Spotlight and the Stanford Core NLP [2]. Finally we evaluated the results in relation to runtime and correctness. Here we measured how many triples from DBpedia we extracted per article, and manually checked the correctness of triples that are not contained in DBpedia. Due to time constraints we chose the category of U.S. Presidents for the evaluation.

1.1 Architecture

Just a few points to the architecture of our project. We used the WikiCleaner and OntologyParser classes to process the given DBpedia ontology and wikipedia dump. Then the RelationExtraction class starts two threads, one for the FOX approach and one for our own approach These two threads then try to extract the triples with the help of the NLPParser class and the webservices classes that query the online FOX demo and Spotlight demo respectively. As the last step the extracted triples will be written to a RDF graph.

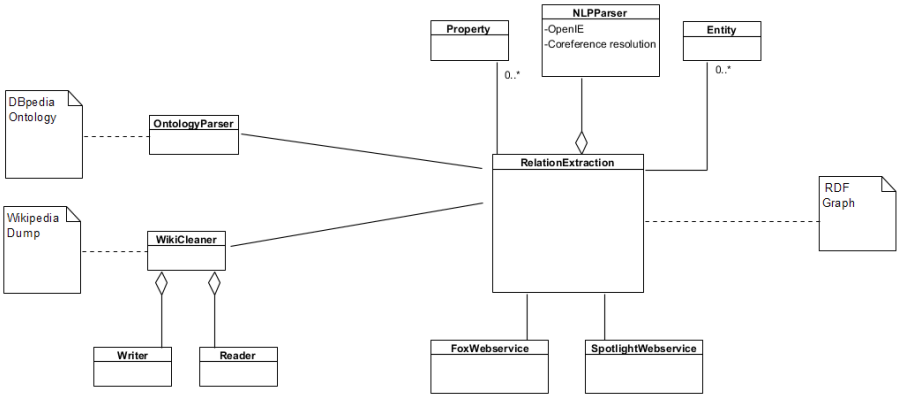


Fig. 1. A quick overview of the architecture of our project.

2 Preprocessing

The preprocessing consists of the cleaning of the wikipedia dump and the coreference resolution.

2.1 Extraction and cleaning of the text from a wikipedia dump

As the foundation we were provided with a wikipedia dump. This wikipedia dump already came without markup, infoboxes or chapter subdivisions. The fact that there are no infoboxes already presented some problems, but more to that during the evaluation and discussion. But there were still some elements that were undesired for the later steps in this process. Such elements include unnecessary URLs, text in parenthesis, whitespace, nullchars and special symbols. We removed the texts passages in parenthesis because the Stanford CoreNLP had some problems with these during the binary relation extraction for our own approach later on. The elements were removed under application of some regular expressions.

The result is a simple text containing only the letters from the english alphabet, numbers, periods, commas and of course simple spaces. In order to speed up the cleaning process, we used a combination of Reader,Worker and Writer threads. To handle the concurreny we used BlockingQueues from the standard Java libraries. One which the Reader thread uses to pass the read lines to the Worker threads and another one which the Worker threads use to pass the cleaned lines to the Writer thread.

2.2 Coreference Resolution

Another step that needs to be taken before the information can be properly extracted is the coreference resolution, so that Spotlight and FOX can also recognize named entities in sentences were previously only a coreference appeared. Therefore we have to replace the pronouns with their representative mention. To actually find this representative mention we used the coref annotator [6] of the Stanford CoreNLP.

Example 1. John met Judy in 1960. He married her during his college year.

In this sentence we obviously want to replace the he, her and his with their corresponding representative mention. The result would look like this:

Example 2. John met Judy in 1960. John married Judy during John's college year.

This example already shows one of the challenges that arise during the coreference resolution. Possesive pronouns have to be detected and handled accordingly. So in this case *his* should not be replaced with *John* but rather with *John's*. To make this distinction we use the dependency parser [5] of the Stanford CoreNLP. Here we take the pronoun in question and then determine if it appears in connection with a possesion modifier relation. If it does, we adjust the representative mention the pronoun will be replaced with accordingly.

3 Named Entity Recognition

For this step we use the DBpedia Spotlight web service. We send the 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. As information we keep the uri, surface form, offset and the types. Since we send more than one sentence at a time to Spotlight we need to keep the offset to map the entities back to their corresponding sentence where they appeared. The surface form we need later on for our own relation extraction where we have to locate the entities inside binary relations. In addition to that we also keep the types because during our own relation extraction we compare these to the range and domain of some properties.

4 Entity Disambiguation

The Entity Disambiguation itself is already done by Spotlight and FOX. Spotlight for example has two approaches for this. After they have located a candidate they take the information next to the surface form as information and try to determine the best match based on that. They also try to weigh words on their ability to actually disambiguate between candidates [3]. FOX and AGDISTIS [4] [8] were already discussed in the lecture.

5 Relation Extraction

5.1 Using FOX

5.2 Own Approach

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 3. 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 4. 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.

6 Example

In the following we will demonstrate the whole process at an example.

Example 5. Obama was born on August 4, 1961, at Kapiolani Medical Center for Women and Children in Honolulu, Hawaii. He graduated from Harvard University.

After the coreference resolution the sentence will look like this:

Example 6. Obama was born on August 4, 1961, at Kapiolani Medical Center for Women and Children in Honolulu, Hawaii. Obama graduated from Harvard University.

Next we use OpenIE to extract binary relations from the sentences, some of them are listed in the following table.

Table 1. Shortened version of the binary relations extracted by OpenIE. (only some of the important ones we can actually use)

Subject	**Predicate**	**Object**
Obama	be bear on	August 4 1961
Obama	graduate from	Harvard University
Obama	be bear at	Kapiolani Medical Center for Women in Honolulu

Then Spotlight will find the following Named Entities.

Example 7. **Obama** was born on August 4, 1961, at **Kapiolani** Medical Center for Women and Children in **Honolulu**, **Hawaii**. **Obama** graduated from **Harvard University** .

Now as already discussed we will try to map the subject, predicate and object to entities, properties and entities or literals. Here we distinguish between entity to literal relation and entity to entity relation.

Table 2. Mappings entity to literal relation. For the first binary relation of Table 1.

Part of binary relation	**Keywords**	**Mapping**
Subject	Obama	dbr:Barack_Obama
Predicate	bear	dbo:birthDate
Object	August 4 1961	1961-08-04

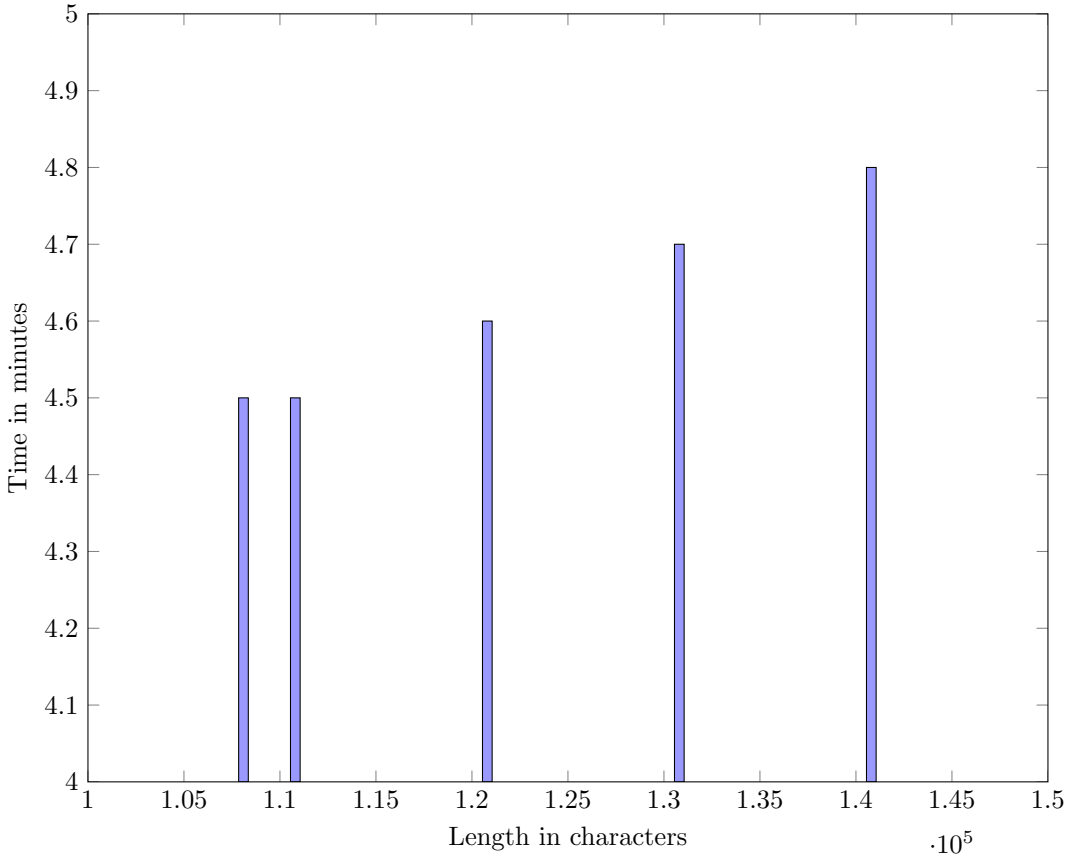
Table 3. Mappings entity to entity relation. For the third binary relation of Table 1.

Part of binary relation	**Keywords**	**Mapping**
Subject	Obama	dbr:Barack_Obama
Predicate	graduate	dbo:almaMater
Object	Harvard University	dbr:Harvard_University

So finally we get these two triples and write them to the graph. In this example we could also find the triple with the birth place of Honolulu.

7 Evaluation

7.1 Runtime



7.2 Correctness

References

1. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems (I-Semantics) (2013)
2. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60 (2014), http://www.aclweb.org/anthology/P/P14/P14-5010
3. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: Shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems. p. 3. I-Semantics '11 (2011)

4. Moussallem, D., Usbeck, R., Röder, M., Ngonga Ngomo, A.C.: Mag: A multilingual, knowledge-base agnostic and deterministic entity linking approach. In: K-CAP 2017: Knowledge Capture Conference. p. 8. ACM (2017), https://svn.aksw.org/papers/2017/KCAP_MAG/sigconf-main.pdf
5. Nivre, J., de Marneffe, M.C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C.D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., Zeman, D.: Universal dependencies v1: A multilingual treebank collection. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016) (2016)
6. Recasens, M., de Marneffe, M.C., Potts, C.: The life and death of discourse entities: Identifying singleton mentions. In: North American Association for Computational Linguistics (NAACL) (2013)
7. Speck, R., Ngonga Ngomo, A.C.: Ensemble learning for named entity recognition. In: The Semantic Web ISWC 2014, Lecture Notes in Computer Science, vol. 8796. Springer International Publishing (2014)
8. Usbeck, R., Ngonga Ngomo, A.C., Auer, S., Gerber, D., Both, A.: Agdistis - graph-based disambiguation of named entities using linked data. In: 13th International Semantic Web Conference (2014), http://svn.aksw.org/papers/2014/ISWC_AGDISTIS/public.pdf