CETUS – A Baseline Approach to Type Extraction

Michael Röder[1], Ricardo Usbeck[1], and Axel-Cyrille Ngonga Ngomo[1]

University of Leipzig, Germany
{roeder,usbeck,ngonga}@informatik.uni-leipzig.de

Abstract. The concurrent growth of the Document Web and the Data Web demands accurate information extraction tools to bridge the gap between the two. In particular, the extraction of knowledge on real-world entities is indispensable to populate knowledge bases on the Web of Data. Here, we focus on the recognition of types for entities to populate knowledge bases and enable subsequent knowledge extraction steps. We present CETUS, a baseline approach to entity type extraction. CETUS is based on a three-step pipeline comprising i) offline, knowledge-driven type pattern extraction from natural-language corpora based on grammar-rules, ii) an analysis of input text to extract types and iii) the mapping of the extracted type evidence to a subset of the DOLCE+DnS Ultra Lite ontology classes. We implement and compare two approaches for the third step using the YAGO ontology as well as the FOX entity recognition tool.

1 Introduction

Both the Document and the Data Web grow continuously. This is a mixed blessing, as the two forms of the Web grow concurrently and most commonly contain different forms of information. Modern information systems must thus bridge this gap to allow a holistic access to the Web. One way to bridge the gap between the two forms of the Web is the extraction of structured data from the growing amount of unstructured information on the Document Web. While extracting structured data from unstructured data allows the development of powerful information system, it also requires high-quality knowledge extraction tool chains to lead to useful results. However, standard document processing pipelines miss the opportunity to gain insights from semantic entities novel to the underlying knowledge base (KB). That is, most known tool chains recognize entities based on linguistic models and link them to a KB or null if they are emerging entities. Assigning a type to these entities is a well known task [7] and has been in the focus of several recent challenges, e.g., the TAC KBP Entity Linking challenge 2014[1], the Micropost workshop series[2] and the OKE challenge 2015[3].

[1] `http://nlp.cs.rpi.edu/kbp/2014/`
[2] `http://www.scc.lancs.ac.uk/microposts2015/`
[3] `http://2015.eswc-conferences.org/important-dates/call-OKEC`

In this article, we present CETUS, a pattern based entity type extraction tool for identifying the type of a given entity inside a given text and linking this type to a KB, i.e., to the DOLCE+DnS Ultra Lite ontology classes[4]. CETUS is a fast and easy to implement baseline approach to path a way to novel research insights. CETUS' pipeline is divided into three subsequent parts: i) an a-priori pattern extraction, ii) a grammar-based analysis of the input document and iii) mapping the type evidence to the DOLCE+DnS Ultra Lite classes. CETUS implements two approaches for the third step using the YAGO ontology as well as the FOX entity recognition tool. We will explain these parts in detail in the Sections 3, 4, 5 and 6 respectively, before we are summarizing the results of the OKE Challenge in Section 7 and conclude in Section 8. The source code of CETUS can be found at `https://github.com/AKSW/Cetus`.

2 Related Work

Next to the above mentioned challenges about entity linking, several tools have been introduced with the ability to type entities, e.g., FOX [9]. However, most of these systems differ in several major aspects compared to CETUS. First, most of the existing tools comprise a complex work flow and are using techniques ranging from supervised and semi-supervised to unsupervised learning methods [7]. Thus, these tools can not serve as a baseline with a simple approach. Second, CETUS marks the part of a given document that contains the type evidence, i.e., a string indicating the chosen type. Third, in contrast to the most other tools, CETUS uses the DOLCE+DnS Ultra Lite ontology classes for typing and is, thus, able to take part the OKE Challenge 2015.

Our approach is mainly based on patterns inspired by Hearst Patterns [3]. Those patterns match text parts describing hyponym relations between two nouns. There have been several other tools that are using patterns to identify the parts of a document containing the type of an entity, e.g., Snow et al. [8]. However, these tools differ in terms of complexity. While some of them are using a predefined set of patterns or rules, other approaches try to discover new patterns from a given corpus using bootstrapping. Since CETUS should serve as an easy to implement baseline for the OKE Challenge, we decided to use a straight forward a-priori iterative, incremental pattern extraction process described in Section 3.

3 Pattern Extraction

The patterns used for identifying the type of an entity inside a document, are generated semi-automatically in an iterative manner. First, CETUS identifies phrases containing entities and their types in a given document corpus (here we use the DBpedia 2014 abstracts) and extracts them. After sorting these

[4] See `http://stlab.istc.cnr.it/stlab/WikipediaOntology/`. Throughout this paper, we use the prefix `dul` for types of this ontology.

phrases according to the string in between the entity and its type, we analyze them and create the patterns in an incremental process. The progress of our pattern extraction is measured by the amount of phrases that are covered by our patterns. In the following, these steps are described in more detail.

3.1 Sentence Part Extraction

For extracting the phrases containing entities and their types, we used the abstracts of the English DBpedia 2014 abstracts dump file. Every abstract describes the entity it belongs to and, thus, contains the label of the entity and its type. We assume, abstracts are written properly and thus contain both information.

First, CETUS preprocesses each abstract individually. Our approach removes the text written in brackets, e.g., pronunciations. Afterwards, we use the Stanford CoreNLP [6] library for part-of-speech tagging and lemmatization as well as the Stanford Deterministic Coreference Resolution System [4] to replace pronouns with their coreferenced words, e.g., *He studied physics* with *Albert Einstein studied physics*. The last step of the preprocessing is the splitting of the abstracts into single sentences.

Second, sentences containing the entity label and at least one label of one of its types (`rdf:type`) are processed further. CETUS extracts the part of the sentence between the entity label and the type label and stores additionally the words, their lemmas and part-of-speech tags of the extracted phrase.

After analysing all abstracts, CETUS counts the different phrases. Table 1 shows examples of extracted phrases and their counts how often they have been found inside the English DBpedia. The words inside these parts are encoded as `<word>_<lemma>_<pos-tag>`.

Delving into the extracted phrases reveals insights into the structure of entity type descriptions in DBpedia abstracts. It can be seen that the formulation "`<entity>` *is a* `<type>`" occurs most often. The second most common formulation uses a type preceding the entity and is listed as the second example in table 1. The third example is a variant of the first one containing the determiner "an" instead of "a". The fourth example shows that some abstracts contain more complex formulations like "`<entity>` *is a* `<type>` *of* `<type>`" while the last example contains an additional adjective that was not a part of the types label, i.e., "flowering".

3.2 Grammar Construction

The aim of creating a grammar is to generate a parser that is able to identify the part of a sentence describing an entities type given the position of the entity inside the sentence. For generating a parser based on our grammar, we are using the ANTLR4 library[5].

Our grammar is based on the following assumptions:

[5] `http://www.antlr.org/`

Extracted phrase	Count
`<entity> is_be_VB a_a_DT <type>`	242 806
`<type> <entity>`	107 082
`<entity> is_be_VB an_an_DT <type>`	12 981
`<entity> is_be_VB a_a_DT species_species_NN of_of_IN <type>`	12 554
`<entity> is_be_VB a_a_DT species_species_NN of_of_IN flowering_flower_JJ <type>`	4 069

Table 1. Examples of sentence parts found between an entity and its type.

1. A sentence contains an entity and a type. Otherwise the sentence is not part of our grammar language.
2. A type should contain at least one noun, but can contain additional words that are specifying the meaning of the noun, e.g., adjectives. If a noun could not be found, a single adjective can be used as type as well.

The first assumption simplifies the task of defining a grammar since we can focus on the sentences that are important for our task and ignore all others. The second assumption contains the definition of a type surface form. It might seem to be contradictory w.r.t. the last example of Table 1 but for the extraction it is important that we extract all words that *could* be part of the types surface form. Following this assumptions, we can define a type inside the grammar with the rule in Listing 1.1.[6]

```
type : (ADJ|VERB|ADVERB|CD)* FOREIGN? NOUN+ (ADJ NOUN)*
     | ADJ;
```

Listing 1.1. The grammar rule defining a type surface form.

A surface form of a type can contain a number of adjectives, verbs or adverbs as well as a foreign word, e.g., the latin word "sub". Additionally, a type has one or more nouns.

As mentioned above, the construction of the grammar is designed to be an iterative, incremental, self-improving process. We start with the simple *is-a* pattern that matches the most common phrase "`<entity>` *is a* `<type>`". The definition of this pattern is shown in Listing 1.2

With this simple grammar, we try to match all phrases extracted beforehand and create a list containing all those phrases that have not been matched so far. Using this list, we extend our grammar to match other phrases. In our example, we extend the simple *is-a* pattern towards matching different temporal forms of the verb "be" and different determiners, e.g., "a" and "an", see Listing 1.3.

[6] Abbreviations in Listing 1.1: ADJ = adjective, CD = cardinal number.

```
is_a_pattern : ENTITY is_be_VB a_a_DT type;
```

Listing 1.2. First simple version of the *is-a* pattern. `ENTITY` is a marking for the entities position.

```
is_a_pattern : ENTITY FORM_OF_BE DETERMINER type;

FORM_OF_BE : ~[ \t\r\n]+ '_be_VB' ~[ \t\r\n]?;
DETERMINER : ~[ \t\r\n]+ '_' ~[ \t\r\n]+ '_DT';
```

Listing 1.3. Extended version of the *is-a* pattern.

With this iterative, incremental process, we further extended the grammar until we covered more than 90% of the extracted phrases.[7]

4 Type Extraction

The pattern-based type extraction can be separated into two steps. The first step extracts type evidence strings from the text, while the second step creates a local type hierarchy based on the extracted string. In the following, we describe both steps in more detail.

4.1 Type String Extraction

To identify the type evidence string for a certain entity, CETUS extracts the string containing the type of a given entity from a given text using the grammar from above. Let us assume the following running example: CETUS processes the document as input with "Albert Einstein" marked as entity.

*In 1921, **Albert Einstein** got the Nobel Prize in Physics. He was a German-born theoretical physicist.*

First, the Stanford Deterministic Coreference Resolution System is applied to replace the pronoun of the second sentence by "Albert Einstein".

*In 1921, **Albert Einstein** got the Nobel Prize in Physics. **Albert Einstein** was a German-born theoretical physicist.*

After that, the text is split into sentences and the surface form of the entity is replaced by a placeholder.

*In 1921, **ENTITY** got the Nobel Prize in Physics.*
***ENTITY** was a German-born theoretical physicist.*

[7] The complete grammar can be found in the projects source code repository.

A parser based on the grammar from Section 3.2 is applied to every sentence. While the first sentence is identified as not contained in the language of the grammar, the second sentence is identified to be in the language. Moreover, the parser identifies "German-born theoretical physicist" as evidence type string.

4.2 Local Type Hierarchy

Based on the extracted evidence type string, CETUS creates a local type hierarchy and links the given entity to the hierarchy. The type hierarchy comprises classes that are generated automatically from the extracted string based on the second assumption of Section 3.2. Each class is generated by concatenating the words found in the extracted string using camel case. After a class has been created, the first word is removed and the next class is created. Every following class is a super class of the classes generated before. Finally, the entity is connected to all generated classes.

For our example, three classes would be generated and linked to the entity as shown in Figure 1 and Listing 1.4[8].

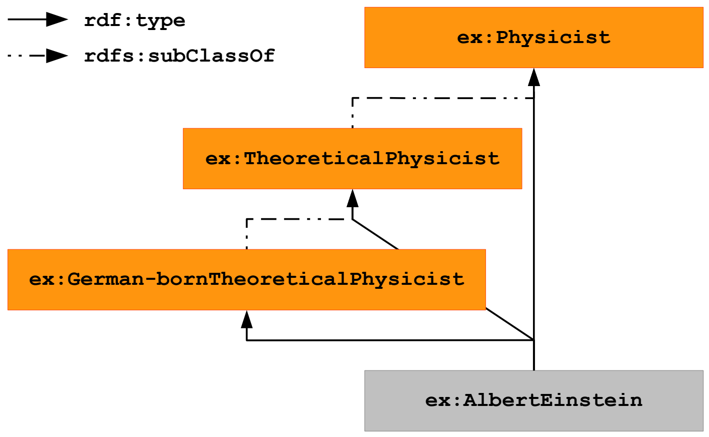


Fig. 1. Schema of the generated local hierarchy of the example.

5 Entity Type Linking using YAGO

The linking of the generated classes to a KB can be done in two different ways. Our first approach, CETUS$_{YAGO}$, uses the labels of the automatically generated classes to find a matching class inside another, well-known KB. CETUS uses the

[8] The `rdfs` prefix stands for `http://www.w3.org/2000/01/rdf-schema#` while the prefix `ex` could stay for every user defined vocabulary, e.g., `http://example.com/`.

```
ex:AlbertEinstein
    a ex:German-bornTheoreticalPhysicist,
        ex:TheoreticalPhysicist, ex:Physicist .

ex:German-bornTheoreticalPhysicist
    a rdfs:Class ;
    rdfs:subClassOf ex:TheoreticalPhysicist ;
    rdfs:label "German-born theoretical physicist" .

ex:German-TheoreticalPhysicist
    a rdfs:Class ;
    rdfs:subClassOf ex:Physicist ;
    rdfs:label "theoretical physicist" .

ex:German-Physicist
    a rdfs:Class ;
    rdfs:label "physicist" .
```

Listing 1.4. The local hierarchy that is generated from the extracted string expressed using turtle.

YAGO ontology [5] which comprises a large class hierarchy and, thus, increases the chance to match one of these classes. YAGO itself contains more than 10 mio. entities and exceeds 350.000 classes.

First, we created an index containing the surface forms of the YAGO classes with a mapping to the class URIs. Second, for every class that has been generated during the extraction step described in Section 4, CETUS retrieves all YAGO classes with a label equal to the label of the generated class. All retrieved classes are linked to the local generated class using a `owl:equivalentClass` predicate.

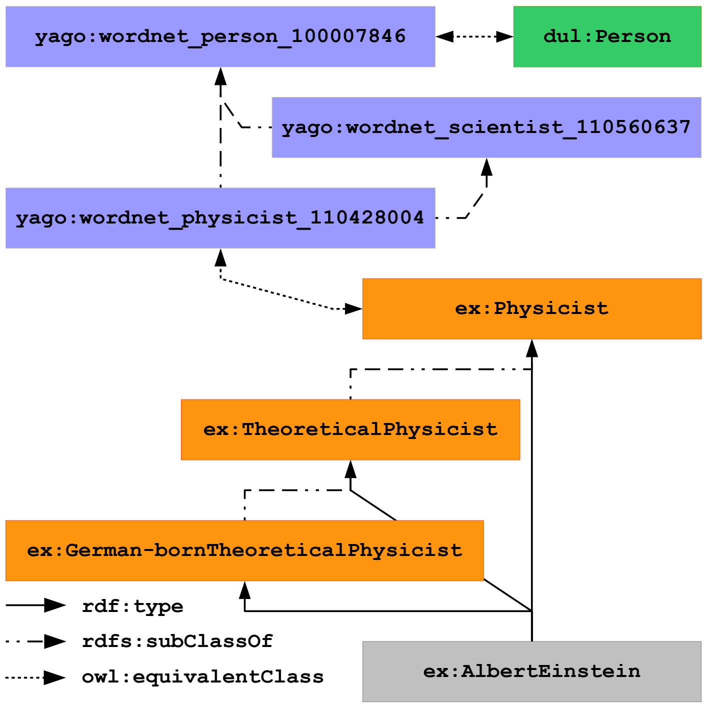
After that, we are using a predefined mapping from the YAGO ontology to the DOLCE+DnS Ultra Lite ontology[9] to iterate through the class hierarchy from the linked classes to the root of the DOLCE ontology. The lowest DOLCE classes on these paths to the root are used as super type for the local generated classes and, thus, are used as types for the entity. The result for our running example can be seen in Figure 2.[10]

6 Entity Type Linking using FOX

A second approach for a type extraction baseline is the usage of one of the various, existing entity typing tools. For our second version CETUS$_{FOX}$, we are

[9] This mapping can be found inside the git repository of the project at `https://github.com/AKSW/Cetus/blob/master/DOLCE_YAGO_links.nt`.

[10] Throughout this paper, we use the prefix `yago` for `http://yago-knowledge.org/resource/`.



yago:wordnet_person_100007846
dul:Person
yago:wordnet_scientist_110560637
yago:wordnet_physicist_110428004
ex:Physicist
ex:TheoreticalPhysicist
ex:German-bornTheoreticalPhysicist
rdf:type
rdfs:subClassOf
owl:equivalentClass
ex:AlbertEinstein

Fig. 2. Resulting type hierarchy that is created based on the YAGO ontology.

using FOX[11], a named entity recognition and typing tool based on ensemble learning over 8 different tools.

CETUS$_{FOX}$ sends the given document to the FOX web-service for retrieving annotations. If the entity inside the document is found and typed by FOX, the type is used to choose one of the DOLCE+DnS Ultra Lite classes, see Table 2. The chosen class is used as super class for the automatically created classes. Unfortunately, FOX identifies only persons, locations and organizations in its current version.

With respect to our running example, the FOX tool marks ”Albert Einstein” as a person. Thus, the created classes would be defined as subclasses of `dul:Person` as shown in Figure 3.

7 Evaluation

CETUS and two other tools—FRED [1] and OAK [2]—participated in the second task of the OKE Challenge 2015. The dataset used for the evaluation contains 99 documents. For evaluating the different systems, a local modified version of

[11] `http://aksw.org/Projects/FOX.html`

FOX class	DOLCE+DnS Ultra Lite class
`scmsann:PERSON`	`dul:Person`
`scmsann:LOCATION`	`dul:Place`
`scmsann:ORGANIZATION`	`dul:Organization`

Table 2. Mapping from FOX classes to DOLCE+DnS Ultra Lite classes.

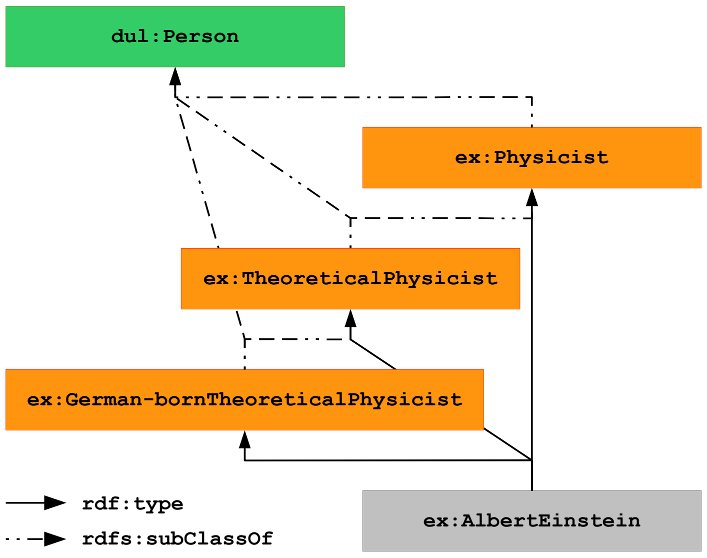


Fig. 3. Resulting type hierarchy that is created based on the results of FOX.

GERBIL [10] has been used. Since the official results contained only the results of CETUS$_{YAGO}$[12] we set up an instance of GERBIL and repeated the evaluation for both versions of CETUS. The results can be seen in table 3. The tables show that both versions of CETUS outperform the other participants regarding the F1 score.

Table 4 shows the detailed results of the two steps of CETUS. It can be seen, that the pattern based recognition of the string containing the type of an entity performs well with a micro F1 measure of $\tilde{0}.7$. However, there is still space for improvement. A large problem for this approach are formulations that have a different grammatical structure than those inside the DBpedia abstracts. Thus, a system with a better understanding of the internal structure of the sentence, e.g., by using parse trees, could avoid these problems.

Comparing both type linking approaches, it can be seen that both have a similar precision (see table 4). But the YAGO-based approach has a higher recall leading to a slightly higher F1 score. The FOX-based type linking lacks

[12] The results of the challenge can be found at `https://github.com/anuzzolese/oke-challenge#results`.

System	Micro			Macro		
	F1	Prec.	Recall	F1	Prec.	Recall
CETUS$_{YAGO}$	**0.47**	0.45	**0.52**	**0.45**	0.42	**0.53**
CETUS$_{FOX}$	0.46	0.45	0.46	0.44	**0.42**	0.47
OAK@Sheffield	0.44	**0.52**	0.39	0.39	0.40	0.40
FRED	0.30	0.29	0.32	0.27	0.26	0.32

Table 3. Results of the OKE Challenge 2015

System	Micro			Macro		
	F1	Prec.	Recall	F1	Prec.	Recall
CETUS (Type Recognition)	0.70	0.69	0.70	0.66	0.64	0.72
CETUS$_{YAGO}$ (Type Linking)	0.25	0.20	0.34	0.23	0.20	0.34
CETUS$_{FOX}$ (Type Linking)	0.22	0.21	0.22	0.22	0.21	0.22

Table 4. Results for the different sub tasks

the identification of types different to persons, organizations and locations. The YAGO-based type linking suffers from two main problems. First, some of the extracted local types cannot be matched to YAGO types. This might be solved by using a better search strategy for finding YAGO types with a similar label, e.g., trigram similarity. The second point of failure is the mapping from YAGO to DOLCE types. For some YAGO types there are no linked DOLCE types while for others the linked DOLCE types are very high inside the hierarchy leading to a coarse typing result and, thus, to a lower precision. A further improvement of the mapping between YAGO and DOLCE types could reduce these problems.

8 Conclusion

We presented CETUS—a pattern based type extraction that can be used as baseline for other approaches. Both versions—CETUS$_{YAGO}$ and CETUS$_{FOX}$—have been explained in detail. We showed how the first one uses a label matching for determining a super type for the automatically generated classes while the second is based on one of the various, existing entity typing tools. Both versions outperformed the competing systems during the OKE Challenge 2015. However, the evaluation pointed out several possibilities for further improvement.

Acknowledgements This work has been supported by the FP7 project GeoKnow (GA No. 318159) and the BMWI Project SAKE (Project No. 01MD15006E).

References

1. S. Consoli and D. Reforgiato. Using fred for named entity resolution, linking and typing for knowledge base population. In *Proceedings of the OKE Challenge 2015 co-located with the 12th Extended Semantic Web Conference (ESWC 2015)*, 2015.

2. J. Gao and S. Mazumdar. Exploiting linked open data to uncover entity type. In *Proceedings of the OKE Challenge 2015 co-located with the 12th Extended Semantic Web Conference (ESWC 2015)*, 2015.
3. M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, 1992.
4. H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 2013.
5. F. Mahdisoltani, J. Biega, and F. Suchanek. YAGO3: A knowledge base from multilingual Wikipedias. In *CIDR*, 2014.
6. C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.
7. D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.
8. R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS 2004)*, November 2004.
9. R. Speck and A.-C. N. Ngomo. Ensemble learning for named entity recognition. In *ISWC*. 2014.
10. R. Usbeck, M. Röder, A.-C. Ngonga Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann, P. Ferragina, C. Lemke, A. Moro, R. Navigli, F. Piccinno, G. Rizzo, H. Sack, R. Speck, R. Troncy, J. Waitelonis, and L. Wesemann. GERBIL – General Entity Annotation Benchmark Framework. In *24th WWW conference*, 2015.