Entity Typing and Linking Using SPARQL Patterns and DBpedia

Lara Haidar-Ahmad[1], Ludovic Font[1], Amal Zouaq[1,2], and Michel Gagnon[1(✉)]

[1] Department of Computer and Software Engineering, Ecole Polytechnique de Montreal, Montreal, Canada
{lara.haidar-ahmad,ludovic.font, michel.gagnon}@polymtl.ca, azouaq@uottawa.ca
[2] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada

Abstract. The automatic extraction of entities and their types from text, coupled with entity linking to LOD datasets, are fundamental challenges for the evolution of the Semantic Web. In this paper, we describe an approach to automatically process natural language definitions to (a) extract entity types and (b) align those types to the DOLCE+DUL ontology. We propose SPARQL patterns based on recurring dependency representations between entities and their candidate types. For the alignment subtask, we essentially rely on a pipeline of strategies that exploit the DBpedia knowledge base and we discuss some limitations of DBpedia in this context.

1 Introduction

The growth of the Semantic Web depends on the ability to handle automatically the extraction of structured information from texts and the alignment of this information to linked datasets. The first OKE Challenge competition [1] targeted these two issues and is a welcome initiative to advance the state of the art of open information extraction for the Semantic Web. In this paper, we present our service for entity typing and linking using SPARQL patterns and DBpedia[1]. This service is the winner of the OKE challenge 2016 Task 2.

Besides a participation to the OKE challenge, one aim of this research is to provide a task-based evaluation of the DBpedia knowledge base. Hence our linking strategies exploit both the DBpedia ontology and the DBpedia knowledge base to extract *rdfs:subClassOf* relationships between natural language types and DBpedia types.

This paper is structured as follows: Sect. 2 presents some related work. Sections 3 and 4 describe the two subtasks of our service: type recognition and extraction from text, and type alignment using the ontology Dolce+DUL. In Sect. 5, we present the evaluation of our system. We discuss our results in Sect. 6.

[1] http://westlab.polymtl.ca/OkeTask2/rest/annotate/post.

H. Sack et al. (Eds.): SemWebEval 2016, CCIS 641, pp. 61–75, 2016.
DOI: 10.1007/978-3-319-46565-4_5

2 Related Work

Several tasks are related to the challenge of entity typing and alignment, among which we can cite named entity recognition [2], relation extraction [3–5], ontology learning [6] and entity linking [7–9]. Due to space constraints, this state of the art will be limited to the participants of the previous OKE challenge [1].

2.1 Type Extraction

The automatic extraction of *taxonomical* and *instance-of* relations from text has been a long-term challenge. Overall, state-of-the-art approaches that target the extraction of relations from text are mainly pattern-based approaches. In the first edition of the 2015 OKE challenge, there were three participating systems for the task of type extraction from natural language definitions: CETUS [10], OAK@Sheffield [11] and FRED [12]. CETUS relies on grammar rules based on parts of speech (POS) to extract an entity type from text. OAK uses machine learning to learn to recognize the sentences' portions that express the entity type, and then uses a POS pattern grammar for type annotation. FRED uses the system Boxer [13] and Discourse Representation Theory, and thus relies on a complex architecture for ontology extraction that is not limited to type extraction. Compared to previous pattern-based approaches in the OKE competition [10, 11], our system differs by the nature of the patterns, which exploit a dependency grammar representation. One particular novelty is the use of SPARQL to model and search for patterns occurrences. Overall, we believe that our approach represents a middle ground between patterns based on a superficial representation of sentences (usually parts of speech) and approaches such as FRED [12] which depend on complex first-order logic and frame semantics.

2.2 Type Alignment

In the context of the Semantic Web, the challenge of entity typing is coupled with the difficulty of finding an alignment with linked datasets. Among the three systems of the OKE challenge 2015 mentioned previously, the authors of CETUS [10] developed an alignment between Yago and Dolce + DnS Ultralite; FRED [12] uses an already existing API that exploits Dolce, WordNet and VerbNet; OAK [11] relies on the existence of *dul* types in DBpedia, using a method similar to our method 2 (see Sect. 4.1). In our approach, we chose to use the existing mappings DBpedia - Dolce + DnS Ultralite [14] and Yago wordnet - Dolce + DnS Ultralite [15].

Our main contribution in this subtask is the exploitation of several strategies that consider either the DBpedia ontology (T-box) or the DBpedia knowledge base (A-box) to find a DBpedia type. We exploit both the knowledge about the entity and the type given as input. When there is not any direct type information linked to the DBpedia ontology or Yago, we revert to type inference methods. Among the strategies described in Sect. 4.1, method 6 is based on our previous work [16] to infer types using predicates' domain and range, while method 2 is similar to the one used by OAK [11].

However, we also introduce a novel approach based on DBpedia categories and propose a pipeline of strategies that aggregates several methods.

3 Entity Type Extraction

Entity type extraction consists in finding the natural language type of an entity, given its textual definition. Our approach relies on pattern extraction using a dependency-based syntactic analysis. The extraction of an entity type is processed in two steps: sentence representation in RDF and pattern occurrence identification using SPARQL queries.

3.1 Sentence Graph Representation

First, we extract grammatical dependencies from the definitions using the Stanford parser [17] and build an RDF graph representing each sentence. Before the parsing step, we identify the input DBpedia entity in the sentence and aggregate multi-words entities with an underscore between the words. For instance, in the sentence *All's Well That Ends Well is a play by William Shakespeare*, we identify *All's Well That Ends Well (*the input DBpedia resource) as one single entity and simply modify the sentence to obtain *All's_Well_That_Ends_Well is a play by William Shakespeare*.

We then construct an RDF graph representing the dependency structure of the definition. Thus we specify the label and part of speech of each word in addition to its grammatical relations with the other words. This RDF graph allows us to look for pattern occurrences using SPARQL requests in the following step. Figure 1 presents the RDF graph of the definition *Skara Cathedral is a church in the Swedish city of Skara*.

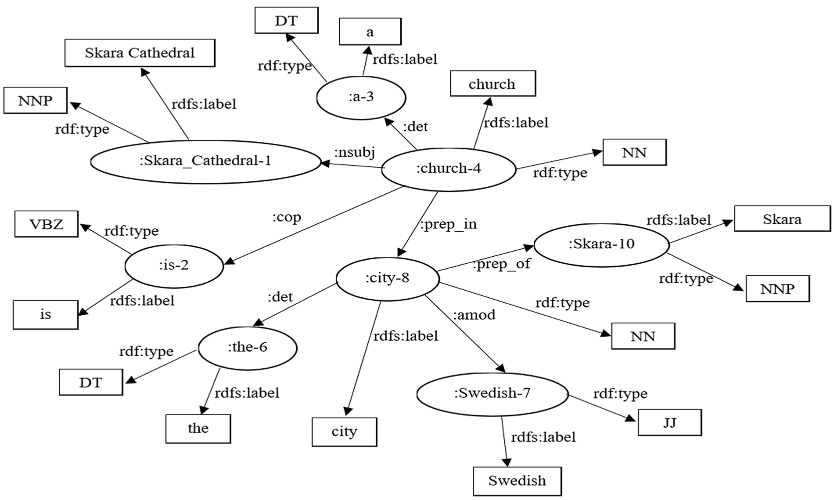

Fig. 1. The RDF Representation of the definition of *Skara Cathedral*.

3.2 Pattern Identification

As for the detection of patterns, based on the train dataset[2] distributed in the OKE challenge, we manually identified several recurring syntactic and grammatical structures between the entities and their respective types. Table 1 presents the most common patterns that we identified in the dataset.

Table 1. Most frequent patterns describing an entity/type relationship.

	Frequent patterns
(1)	$\text{Entity} \xleftarrow{:nsubj} \text{X} \xrightarrow{:cop} \text{Z}$; $\text{X} \xrightarrow{:prep\_of} \text{Type}$ Where X = { "name", "nickname", "alias", "one", "species", "form" } *Sant'Elmo is the name of both a hill and a fortress in Naples, located near the Certosa di San Martino.* ***Entity** = Sant'Elmo ; **Type** = Hill*
(2)	$\text{Entity} \xleftarrow{:nsubj} \text{Type} \xrightarrow{:cop} \text{Z}$ *El Oso, released in 1998, is an album by the New York City band Soul Coughing.* ***Entity** = El Oso ; **Type** = Album*
(3)	$\text{Entity} \xleftarrow{:nsubjpass} \text{V} \xrightarrow{:auxpass} \text{Z}$; $\text{V} \xrightarrow{:prep\_as} \text{Type}$ *Bromius in ancient Greece was used as an epithet of Dionysus/Bacchus.* ***Entity** = Bromius ; **Type** = Epithet*
(4)	$\text{Entity} \xleftarrow{x} \text{Type}$ Where x = { :amod, :appos, :nn} *The AES11 standard published by the Audio Engineering Society provides a systematic approach to the synchronization of digital audio signals.* ***Entity** = AES11 ; **Type** = Standard*

We created a pipeline of SPARQL requests with a specific processing order as shown in Table 1. In fact, Pattern 1 has a higher priority than Pattern 2. For example, in the sentence *Sant'Elmo is the name of both a hill and a fortress in Naples, located near the Certosa di San Martino,* if the second pattern was processed before the first one, we

[2] https://github.com/anuzzolese/oke-challenge-2016/blob/master/GoldStandard_sampleData/task2/dataset_task_2.ttl.

would wrongly extract the type *name*. Similarly, Pattern 4 is the pattern with the lowest priority, which is executed only after all the other patterns are tested.

Each pattern is modeled using a single SPARQL request. The following is an example of the SPARQL implementation of Pattern 2:

```
SELECT ?typeLabel
WHERE {
      ?type :nsubj ?entity.
      ?type :cop ?cop.
?entity rdfs:label ?entityLabel.
?type rdfs:label ?typeLabel.
FILTER(REGEX(?entityLabel,
'^THE_LABEL_OF_THE_DBPEDIA_ENTITY', 'i'))
}
```

As this SPARQL request shows, we search for the type of an entity, where the entity's label is a perfect match with the input DBpedia entity's label. We first execute all the SPARQL patterns in this "full match" mode. In case all requests fail to return a type, we then look for occurrences of the same patterns using a partial match of the entity's label.

Once we find a candidate type, we create an OWL class representing this type. We remove all the accents and special characters and extract the lemma of the types in plural form. Overall, we adopted the singular as a convention for our entity types. For instance, in *Alvorninha is one of the sixteen civil parishes that make up the municipality of Caldas da Rainha, Portugal*, we extract the type *oke:Parish* from the string *parishes*. Finally, we create a *rdf:type* relation between the entity and the returned type.

4 Type Alignment

In this paper, we refer to the namespaces http://dbpedia.org/ontology/ and http://dbpedia.org/page/ as *dbo* and *dbr* respectively. The ontologies http://www.ontologydesignpatterns.org/ont/dul/DUL.owl and http://ontologydesign-patterns.org/ont/wikipedia/d0.owl are represented by the prefixes *dul* and *d0* respectively. Besides, we use "Dolce" as a shortcut for "Dolce + DnS Ultralite".

Once the *natural language type* of a given DBpedia entity is identified, for instance [*dbr:Brian_Banner*, *oke:Villain*], where the first element represents the entity and the second element the natural language type, the second part of the OKE challenge task 2 is to align the identified type to a set of given types in the Dolce ontology[3]. The objective is to link the natural language type to a super-type in the ontology using an *rdfs:subClassOf* link. For instance, *dul:Person* would be a possible super-class for *oke:Villain*.

[3] https://github.com/anuzzolese/oke-challenge-2016#task-2.

Our alignment strategy relies only on the DBpedia knowledge base and its links to external knowledge bases, when applicable, and exploits available mappings between DBpedia and Dolce, and Yago and Dolce. In fact, besides the OKE challenge in itself, one objective of this research is to determine whether the DBpedia knowledge base, one of the main hubs on the Linked Open Data cloud, is a suitable resource for the entity linking task. Thus our goal is to find a link, either directly or indirectly, between the *oke* type (e.g. *oke:Village*) returned by the first subtask and the DBpedia ontology and/or Yago and/or Dolce.

4.1 DBpedia Type Identification

Our global alignment strategy first queries the DBpedia ontology (http://dbpedia.org/ontology/[*Input Type*]). If the type is not found as a DBpedia class, we query DBpedia resources (http://dbpedia.org/resources/[*Input Type*]) and either find direct types or infer candidate types using several strategies. Our queries result in three possible outputs:

1. There is a *dbo* resource for the input type. In our dataset, this case occurred in 83 out of 198 cases (42 %).
2. There is only a *dbr* resource for this type. In this case, we attempt to find a predicate *rdf:type* between the natural language type and some type that can be aligned with Dolce + DnS Ultralite, i.e. a type in the DBpedia ontology, Yago-Wordnet or DUL. In our dataset, this case occurred in 68 % of the cases.
3. There is neither a *dbo* nor a *dbr* resource for this type (e.g. *oke:Villain*). In this case, we cannot infer any type and we rely solely on the entity page (*dbr:Brian_Banner*) to identify a potential type when possible. In our training data set, this case never occurred.

Next, we assign a score to our candidate types based on the number of instances available for these types. Finally, we return the Dolce + DnS Ultralite type that is equivalent or is a super-type of the chosen DBpedia type. The following sections describe the various implemented strategies for type alignment.

Method 1: Alignment Based on the DBpedia Ontology: The first method checks if there is full match between the natural language type and a class in the DBpedia ontology (e.g. for the input "*oke:Villain*", we look for the URI *dbo:Villain*). If such a class exists, we simply align this type with Dolce.

Method 2: Alignment Based on the Type of Instances: In this step, the idea is to exploit the "informal" types available in the *dbr* namespace using the predicate *dbo:type*. For instance, even though *dbr:Village* is not defined as a class, we can find the triple *dbr:Bogoria,_Poland dbo:type dbr:Village*. Thus, given that *dbr:Bogoria,_Poland* is also of type *dbo:Place*, our general hypothesis is that we can consider *dbr:Village* to be a subclass of *dbo:Place.* To choose among all the candidates, we consider all the instances (using *dbo:type*) of *dbr:Village*, and assign a score to each of their types (available through *rdf:type*) depending on the number of times in which they appear in relation with the instances of *dbr:Village*.

Method 3: Alignment Based on the Entity Type: In this strategy, we exploit the information available in the DBpedia entity page itself. In fact, if the given natural language type does not have any DBpedia page, or if that page does not contain any information that could allow us to infer a valid type, we search for a direct *rdf:type* relation in the **entity** description. For instance, for the input [*dbr:Adalgis*, *oke:King*], our assumption is that the triple: *dbr:Adalgis rdf:type dul:NaturalPerson* implies *oke: King rdfs:subClassOf dul:NaturalPerson*. All the (rdf) types of the entity represent our candidate types with an initial score of 1.

Method 4: Alignment Based on Direct Types: Here we query the DBpedia resource corresponding to the natural language type (e.g. *dbr:Club*) and find the triples of the form *dbr:Club rdf:type [Type]* and return *[Type]*. Like in Method 3, all the candidates returned by this method have an initial score of 1.

Method 5: Alignment Based on Categories: This strategy exploits Wikipedia categories represented by the http://dbpedia.org/page/Category: namespace (*dbc*). Categories are indicated in most pages using the predicate *dct:subject*. The idea here is to look at all the categories in which a given type is included (for instance *dbc:Administrative_divisions*, *dbc:Villages*, *etc.* for *dbr:Village*), and then find the type(s) of all the elements in each of these categories. In this example, the category *dbc:Villages* contains several villages (such as *dbr:Mallekan*) of type *dbo:Place*. *dbo:Place* is therefore a candidate type for *dbr:Village*. Like in previous methods, this approach returns many candidates. Each type is given a score equal to the number of triples in which it appears.

Method 6: Alignment Based on Predicates' Domain and Range: This method infers a type for an entity by examining the *rdfs:domain* and *rdfs:range* of predicates that are used in the description of the DBpedia page associated with the natural language type. For instance, the two triples:

$$\begin{array}{lll} dbr{:}Marko\_Vovchok & dbo:birthplace & dbr:Village \\ dbo:birthPlace & rdfs:range & dbo:Place \end{array}$$

allow us to infer *dbr:Village rdf:type dbo:Place* using the information available in the range of the predicate. In this approach, we only take into account the *dbo* predicates, as the *dbp* (http://dbpedia.org/property) predicates typically do not have any domain or range specified. Like in method 2, we give each inferred type a score equal to the number of triples in which the type is used.

4.2 Dolce+DUL Alignment

Following all our type identification methods, we obtain a set of candidate types with a score. Next, we rely on the alignment between the DBpedia ontology and Dolce + DnS Ultralite to replace each *dbo* type with their *dul/d0* counterpart. The same is done to replace *yago* types with *dul/d0* types. However, as the set of types used by the OKE challenge does not include all *dul* types, we modify this alignment in the following way: if a *dul* type is not included in the set of OKE challenge types, we replace it by its closest ancestor that is included in the set. For instance, *dul:SocialPerson* is not an element of the OKE challenge set, but its super class, *dul:Person*, is available. Therefore, if our alignment returns *dbo:Band rdfs:subClassOf dul:SocialPerson*, our final output is *dbo:Band rdfs:subClassOf dul:Person*. In our experiments on the OKE dataset, this strategy did not work well only with the *dul* types *dul:Concept* and *dul:Agent*, which do not have any parent in the OKE challenge set.

Next, the obtained set of *dul* candidates is very often a set of classes that have some taxonomical link among themselves. Given that our objective is to find the most precise candidate, the score of each candidate is modified by adding the score of its ancestors among this set, thus effectively favoring classes that are deeper in the taxonomy. Finally, the chosen candidate is the *dul* type with the highest score.

Here is a full example of our process for method 2 (instances) with the input (*dbr:Calvarrasa_de_Abajo,* "*oke:Village*"). First, we retrieve all the URIs that appear in a triple of the form *[subject] dbo:type dbr:Village*. Then, we retrieve all the types (*rdf:type*) of URIs of the form: *[subject] rdf:type [dbo_type]*. Each of these types' score increases by 1 every time it appears. In this example, the final list contains 26 types, with the best (score-wise) being: *dbo:Place* (480), *dbo:Location* (480), *dbo:PopulatedPlace* (480), *dbo:Settlement* (480), *dbo:Village* (460), *yago:location* (436) and *yago:object* (436). After the DOLCE alignment, this list becomes *d0:Location* (1905), *dul:PhysicalObject* (437), *dul:Object* (436) and *dul:Region* (436). During this step, if several types are aligned to the same *dul/d0* type, their scores are combined.

Finally, we check if our candidates include *dul* types that are not available in the OKE challenge set, and replace them by their equivalent (if available) or closest ancestor type that is available in the OKE set. Here, *dul:Region* is replaced by *d0:Characteristic*. We end up with *d0:Location* (1905), *dul:PhysicalObject* (873), *dul:Object* (436) and *d0:Characteristic* (436). The type with the highest score is *d0:Location* (1905), therefore we return *oke:Village rdfs:subClassOf d0:Location.*

5 Evaluation

5.1 Type Extraction Evaluation

Our first evaluation calculates the precision and recall of the natural language type identification subtask. We consider a type as a true positive only when its lemmatized oke type is a perfect match with at least one of the lemmatized oke types of the OKE gold standard (See footnote 2). Using this evaluation method on the 2016 train dataset, our precision and recall for the type extraction subtask is 87 % as shown in Table 2a, and 80 % on the evaluation dataset as shown in Table 2b. Table 4 presents the official evaluation on the 2016 train dataset using Gerbil[4] [18]. We can notice some decrease in performance using Gerbil (Tables 2a and 4), some of which can be explained by the existence of OKE types in plural form in the gold standard (e.g. *oke:Awards* versus

[4] http://gerbil.aksw.org/gerbil.

oke:Award). In fact, contrary to Table 4, the results in Table 2a take into account the lemmatization of both the natural language types and the gold standard types.

Table 2a. Statistics for the type extraction evaluation on the train dataset

Pattern	(1)	(2)	(3)	(4)	Total
Found types	10	156	2	4	172
Total occurrences	14	173	2	9	198
Precision/recall	71 %	90 %	100 %	44 %	87 %

Table 2b. Statistics for the type extraction evaluation on the evaluation dataset

Pattern	(1)	(2)	(3)	(4)	Total
Found types	1	39	0	0	39
Total occurrences	1	47	0	2	50
Precision/recall	100 %	82.98 %	–	0 %	80 %

We performed an analysis of the unsuccessful sentences and identified few potential sources of errors. A good proportion of errors arise from grammatical ambiguities and incorrect syntactic analyses of sentences. This is the case for sentences like *Brad Sihvon was a Canadian film and television actor*, for which we find the type *oke:Film* instead of *oke:Actor*, due to an error in the parsing process. Similar errors can also occur, in some rare cases, for long sentences like *Bradycardia, also known as bradyarrhythmia, is a slow heart rate, namely, a resting heart rate of under 60 beats per minute (BPM) in adults* for which we extract the type *oke:Heart* instead of *oke:Rate*. In some cases, the errors are debatable. We list as examples the sentence *Gimli Glider is the nickname of an Air Canada aircraft that was involved in an unusual aviation incident*, for which we extract the type *oke:Aircraft* instead of *oke:Nickname*, or *Caatinga is a type of desert vegetation, and an ecoregion characterized by this vegetation in interior northeastern Brazil*, for which we find the type *oke:Vegetation* instead of *oke:TypeOfDesertVegetation*.

We also evaluated the precision and recall of our patterns separately. Given that the precision and recall are the same, Tables 2a and b show the results for each pattern based on the 2016 OKE train and evaluation datasets.

5.2 Type Alignment Evaluation

To assess the efficiency of each type alignment method, we compared the obtained types to those present in the gold standard. Table 3 shows the number of returned types, the number of correct types, as well as the precision, recall and F-measure for each method on the OKE challenge train dataset. Taken individually, most methods achieve limited or poor performance. However, we also implemented a pipeline strategy to combine these methods, thus increasing the recall of our approach. The pipeline is based on the most successful to the least successful strategies (in terms of precision) based on the results of individual methods. In this pipeline, a strategy is executed only if the previous one was unsuccessful in returning a type.

Table 3. Comparison of each method for type alignment on the OKE challenge train dataset

Method	Returned types	Correct types	Precision	Recall	F-measure
1: ontology	83	59	71 %	30 %	42 %
2: instances	57	38	67 %	19 %	30 %
3: entity	140	67	48 %	34 %	40 %
4: direct type	17	6	35 %	3 %	6 %
5: category	130	22	15 %	10 %	12 %
6: predicates	89	11	12 %	6 %	8 %
Pipeline 1–6	172	96	56 %	48 %	52 %

5.3 Overall Results

The overall results of the task of entity typing and alignment on the OKE training dataset using the evaluation framework Gerbil are shown in Table 4. These results rely on the pipeline strategy for the type alignment subtask.

We can notice a slight decrease in performance compared to our local evaluation on the 2016 train dataset.

Table 4. Overall precision, recall and F-measure computed using Gerbil on the 2016 train dataset

Task	Micro precision	Micro recall	Micro F-measure	Macro precision	Macro recall	Macro F-measure
Type extraction	82.32 %	75.81 %	78.93 %	82.32 %	78.96 %	80.05 %
Type alignment	49.63 %	45.47 %	47.45 %	49.62 %	45.42 %	46.47 %
Total (average)	65.97 %	60.64 %	63.19 %	65.97 %	62.19 %	63.26 %

The system was also tested on the 2016 test dataset; the result for this dataset are presented in Table 5.

Table 5. Overall precision, recall and F-measure computed using Gerbil on the 2016 test dataset

Task	Micro precision	Micro recall	Micro F-measure	Macro precision	Macro recall	Macro F-measure
Type extraction	81.63 %	73.39 %	77.29 %	80.81 %	76.60 %	77.95 %
Type alignment	46.46 %	42.51 %	44.40 %	46.46 %	42.51 %	43.53 %
Total (average)	64.05 %	57.95 %	60.85 %	63.64 %	59.55 %	60.74 %

For comparison purposes, we also report the results of two competing systems on the evaluation dataset in Table 6. These systems are Mannheim [19], a participant to the OKE 2016 challenge and CETUS [10], the baseline system and the winner of the

OKE 2015 challenge. Mannheim uses taxonomical relation ("isa") extraction based on Hearst-like patterns in text to find the entity type, then chooses one of these isa relations and exploits a mapping between OntoWordnet, Wordnet and DOLCE to infer the super class in the OKE types set. CETUS uses pattern extraction to identify potential types, then creates a hierarchy between these types. Finally, it proposes two approaches to align the type with DOLCE: the first one is based on a mapping with Yago, and the second on an entity recognition tool (FOX).

Table 6. Overall precision, recall and F-measure for the two participating systems and the baseline CETUS on the test dataset

System	Task	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
CETUS	Extr.	68.75 %	88.00 %	77.19 %	72.00 %	88.00 %	77.33 %
	Align.	22.17 %	24.47 %	23.26 %	22.17 %	24.47 %	19.89 %
	Total	45.46 %	56.24 %	50.23 %	47.08 %	56.24 %	48.61 %
Mannheim	Extr.	77.27 %	68.00 %	72.34 %	63.00 %	68.00 %	64.67 %
	Align.	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
	Total	38.64 %	34.00 %	36.17 %	31.50 %	34.00 %	32.33 %
WestLab	Extr.	86.00 %	86.00 %	86.00 %	86.00 %	86.00 %	86.00 %
	Align.	8.00 %	6.67 %	7.27 %	8.00 %	6.67 %	7.00 %
	Total	47.00 %	46.33 %	46.64 %	47.00 %	46.33 %	46.5 %

Our system WestLab obtains satisfying results in terms of precision for the type extraction task; we obtain a Micro and Macro values of 86 %, whereas the baseline CETUS obtains 68.75 % (micro) and 72 % (macro), and Mannheim obtains 77.27 % (micro) and 63 % (macro).

As for the recall, we obtain 86 % (micro and macro), which is better than Mannheim's recall of 68 % (micro and macro), but lower than CETUS which obtains 88 % (micro and macro).

Our Micro and Macro F-Measures are both 86 %. These results are higher than CETUS', which are 77.19 % and 77.33 % respectively, and Mannheim's, that obtains 72.34 % and 64.67 % respectively, for the type extraction.

Thus, we can conclude that we outperform other systems when taking into account precision but we note that our recall is lower than the one obtained by the baseline CETUS. These results also show that our patterns do not always detect and extract the type of the entity, which is an indicator that the patterns set must be extended in our future work. However, our patterns rarely extract types that are false positives, which shows that they are well defined and accurate.

Concerning the type alignment, there have been some issues with the test dataset distributed by the OKE challenge organizers at the time of the evaluation, which have been corrected later. This explains the very low performance shown in Table 6 for the type alignment subtask. Given this modification, we are able to provide results only for our system and the CETUS baseline on the corrected evaluation dataset. At the time of this publication, we don't have the updated results for the Mannheim system. Table 7

Table 7. Overall precision, recall and F-measure for the two participating systems and the baseline CETUS on the corrected test dataset

System	Task	Micro			Macro		
		Precision	Recall	F1	Precision	Recall	F1
CETUS	Extr.	68.75 %	88.00 %	77.19 %	72.00 %	88.00 %	77.33 %
	Align.	22.17 %	24.47 %	23.26 %	22.17 %	24.47 %	19.89 %
	Total	45.46 %	56.24 %	50.23 %	47.08 %	56.24 %	48.61 %
WestLab	Extr.	81.63 %	73.39 %	77.29 %	80.81 %	76.60 %	77.95 %
	Align.	46.46 %	42.51 %	44.40 %	46.46 %	42.51 %	43.53 %
	Total	64.05 %	57.95 %	60.85 %	63.64 %	59.55 %	60.74 %

provides our results on the updated dataset, compared with the baseline. Overall, we can notice a huge improvement on the corrected dataset. In fact, the WestLab system obtains an F-Measure of 44.4 % (micro) and 43.53 % (macro) for the type alignment subtask, whereas CETUS obtains 23.26 % and 19.89 %. These results constitute a considerable improvement for the type alignment task, even though they are still under the threshold of 50 %.

For the overall results including both type extraction and alignment, we outperform all systems and obtain F-Measures of 60.85 % (micro) and 60.74 % (macro), whereas CETUS obtains F-Measures of 50.23 % (micro) and 48.61 % (macro). We cannot compare our system with Mannheim on the corrected test dataset, except on the type recognition subtask, for the reasons mentioned in the previous paragraph.

6 Discussion

Type Extraction. One limitation of our approach for natural language type identification is the small number of implemented patterns, which does not guarantee to find an entity type. However, our proposal of SPARQL patterns, coupled with an RDF representation of definitions, represents an elegant and simple solution which facilitates the addition of new patterns. Another limitation comes from the fact that our system relies on a syntactic analysis. Thus, errors that occur in the parsing process also affect our system. However, according to our preliminary results, this approach displays a satisfactory precision and recall values compared to previous approaches in the OKE competition.

DBpedia for Type Alignment. Task alignment requires the discovery of *rdfs:subClassOf* links between natural language types and ontological classes. One of our research objectives was to assess how well a type alignment could be performed based on the structured knowledge available in the DBpedia ontology and resources. Some of our methods exploit the grey zone around the notion of subclass and instance in DBpedia. In fact, DBpedia resources (A-box) cannot be normally expected to use the *rdfs:subClassOf* predicate. However, some of the resources employ the predicate *dbo:type*. For example, *dbr:Bogoria,_Poland dbo:type dbr:Village*. Thus *dbr:Village* can be effectively considered as a *class* based on RDFS semantics. There were 57 (out of 198) similar cases in our train dataset. Based on this line of thought, if we found *dbr:Village rdf:type dbo:Place,* we inferred *dbr:Village rdfs:subClassOf dbo:Place.* These examples show that DBpedia resources (A-box) are also described using an informal or implicit schema. This further highlights the need of describing these resources in the ontology rather than in the knowledge base.

Due to the lack of directly exploitable type information in DBpedia, we relied on type inference methods (M2 - instances, M6 - predicates, M5 - categories) in few cases (27 out of 172 types are retrieved using these methods). More specifically, we employed these strategies when an input type does not have a *dbo* page or when its *dbr* page does not contain any *rdf:type* predicate. However, these methods often give poor results. Finally we did not process the disambiguation pages (e.g. *dbr:Motion*) that are sometimes returned by our methods. Altogether, our system failed to return any type in 14 % of the cases. In this case, it returns *owl:Thing*.

Examples of Problematic Cases. Most of our errors boil down to two error sources: (a) inaccurate, noisy, or plain false information and (b) unavailable information in DBpedia. In the following, we give a few examples of problematic cases in some of the alignment methods.

M2 – instances: According to the gold standard, *dbr:Court* should be a *dul:Organization*. However, in DBpedia, *dbr:Court* instances, as depicted by the *rdf:type* predicate, are inaccurate (e.g. *dbr:Mansion_in_Grabowo_Krolewskie)* or refer to broken links. Our type alignment based on these links wrongly concludes that *dbr:Court* is a *d0:Location*.

M6 – predicates: *dbr:Season* should be a *dul:Situation*. However, in DBpedia, there is a confusion between a season (time of the year) and seasonal music (such as Christmas songs) which does not have a *dbr* resource. Therefore, the resource *dbr:Season* is used erroneously instead of the non-existing *dbr:Seasonal_Music* page. This leads to triples such as *dbr:Christmas _(Kenny_Rogers_album) dbo:genre dbr:Season.* Given that the predicate method exploits *dbo:genre rdfs:range dbo:Genre*, we erroneously conclude that a *dbr:Season* is a subclass of *dbo:Genre*.

M5 – categories: *dbr:Tournament* is part of only one category, *dbc:Tournament_systems*, containing pages such as *dbr:Round-robin_tournament* or *dbr:Double-elimination_tournament*. All of these resources have a type in Yago (*artifact*) that is aligned to *dul:PhysicalObject*, which makes us conclude that a *dbr:Tournament* is a *dul:PhysicalObject*. Here, the error is double: *dbr:Tournament* should not be in the category *dbc:Tournament_systems*, and the resources should not be typed as *yago:Artifact*.

In all the above examples, the correct answer is never present in our candidates list. This observation confirms that DBpedia resources are often poorly described [16]. Despite these limitations, our pipeline, which is based on a set of methods ordered from the most trustworthy to the least one, obtains a micro precision of 49.6 % on the training dataset and 46.5 % on the test dataset, and micro recall of 45.5 % on the training dataset and 42.5 % on the test dataset, which we consider as reasonable given the complexity of the task.

Gold Standard. We had some issues when comparing our results with the gold standard. Quite often, our results could be considered as correct, but are different from the ones in the gold standard as they are based on the DBpedia ontology. For instance, we infer that a *oke:Meeting* is a subclass of *dul:Event* (*dul:Event:* "Any physical, social, or mental process, event, or state"), but the gold standard states that a *oke: Meeting* is a subclass of *dul:Activity*. Both answers could be acceptable. In the OKE train dataset, we identified 20 "borderline" cases out of 198 in the alignment subtask. In the natural language type extraction subtask, we identified some potentially questionable types in the gold standard of the form "Set_Of_X" or "Type_Of_X". For instance, in the sentence *Caatinga is a type of desert vegetation…* our position is that the type could be *oke:DesertVegetation* rather than *oke:TypeOfDesertVegetation*.

Future Work. For the type alignment sub-task, our next step will consider the problem of the disambiguation pages. Such pages represent a non-negligible portion of the data set (26 %), and systematically constitute a source of errors. The objective is to choose the correct type among all the possible disambiguations. For instance, given the input [*dbr:Babylonia, oke:State*], the returned type *dbr:State* is a disambiguation page, linking to pages such as *dbr:Nation_state*, *dbr:State_(functional_analysis)* or *dbr: Chemical_state*.

7 Conclusion

This paper describes our approach for the extraction of entity types from text and the alignment of these types to the Dolce+DUL ontology. The patterns used to extract natural language types from textual definitions achieved high precision and recall values. As for the type alignment, the strength of our approach is based on the multiplicity of strategies which exploit both the DBpedia ontology and knowledge base and rely on DBpedia large coverage. Our experiments highlight the necessity of a better linkage between DBpedia resources and the DBpedia ontology and the need for restructuring some DBpedia resources as ontological classes.

Acknowledgement. This research has been funded by the NSERC Discovery Grant Program.

References

1. OKE 2015 Challenge Description. https://github.com/anuzzolese/oke-challenge/blob/master/participating%20systems/OKE2015_challengeDescription.pdf. Accessed 20 Mar 2016
2. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvist. Investig. **30**(1), 3–26 (2007)
3. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora (1992)
4. Min, B., Shi, S., Grishman, R., Lin, C.Y.: Ensemble semantics for large-scale unsupervised relation extraction. In: 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1027–1037. Association for Computational Linguistics (2012)

5. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL, pp. 1003–1011. ACL, Association for Computational Linguistics (2009)
6. Otero-Cerdeira, L., Rodriguez-Martinez, F.J., Gomez-Rodriguez, A.: Ontology matching: a literature review. Expert Syst. Appl. **42**(2), 949–971 (2015)
7. Bizer, C., Heath, T., Ayers, D., Raimond, Y.: Interlinking open data on the web. Media **79**(1), 31–35 (2007)
8. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: Discovering missing background knowledge in ontology matching. In: Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence, pp. 382–386. IOS Press (2006)
9. Kachroudi, M., Moussa, E.B., Zghal, S., Ben, S.: Ldoa results for OAEI 2011. In: Ontology Matching, p. 148 (2011)
10. Röder, M., Usbeck, R., Ngonga Ngomo, A.: CETUS - a baseline approach to type extraction. In: OKE Challenge 2015 Co-located with the 12th Extended Semantic Web Conference (ESWC) (2015)
11. Gao, J., Mazumdar, S.: Exploiting linked open data to uncover entity types. In: OKE Challenge 2015 Co-located with the 12th Extended Semantic Web Conference (ESWC) (2015)
12. Consoli, S., Reforgiato, D.: Using FRED for named entity resolution, linking and typing for knowledge base population. In: OKE Challenge 2015 Co-located with the 12th Extended Semantic Web Conference (ESWC) (2015)
13. Bos, J.: Wide-coverage semantic analysis with boxer. In: Bos, J., Delmonte, R. (eds.) Semantics in Text Processing, pp. 277–286. College Publications (2008)
14. Ontology Design Patterns (2014). http://ontologydesignpatterns.org/ont/dbpedia_2014_imports.owl. Accessed 20 Mar 2016
15. CETUS. DolCE_YAGO_mapping (2015). http://github.com/AKSW/Cetus/blob/master/DOLCE_YAGO_links.nt. Accessed 20 Mar 2016
16. Font, L., Zouaq, A., Gagnon, M.: Assessing the quality of domain concepts description in DBpedia. In: 11th International Conference on Signal-Image Technology and Internet-Based Systems, pp. 254–261. IEEE (2015)
17. De Marneffe, M-C., Manning, C.D.: The stanford typed dependencies representation. In: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation, ACL (2008)
18. Usbeck, R., Röder, M., Ngonga Ngomo, A.-C., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., Ferragina, P., Lemke, C., Moro, A., Navigli, R., Piccinno, F., Rizzo, G., Sack, H., Speck, R., Troncy, R., Waitelonis, Jö., Wesemann, L.: GERBIL – General Entity Annotation Benchmark Framework. In: 24th WWW Conference (2015)
19. Faralli, S., Ponzetto, S.: DWS at the 2016 open knowledge extraction challenge: a hearst-like pattern-based approach to hypernym extraction and class induction. In: OKE Challenge 2016 Co-located with the 13th Extended Semantic Web Conference (ESWC) (2016)