Towards Linked Hypernyms Dataset 2.0: Complementing DBpedia with Hypernym Discovery and Statistical Type Inference

Tomáš Kliegr, Ondřej Zamazal

Department of Information and Knowledge Engineering, Faculty of Informatics and Statistics, University of Economics
W. Churchill Sq. 4, 130 67 Prague 3, Czech Republic
tomas.kliegr@vse.cz, ondrej.zamazal@vse.cz

Abstract

This paper presents a statistical type inference algorithm for ontology alignment, which assigns DBpedia entities with a new type (class). To infer types for a specific entity, the algorithm first identifies types that co-occur with the type the entity already has, and subsequently prunes the set of candidates for the most confident one. The algorithm has one parameter for balancing specificity/reliability of the resulting type selection. The proposed algorithm is used to complement the types in the LHD dataset, which is RDF knowledge base populated by identifying hypernyms from the free text of Wikipedia articles. The majority of types assigned to entities in LHD 1.0 are DBpedia resources. Through the statistical type inference, the number of entities with a type from DBpedia Ontology is increased significantly: by 750 thousand entities for the English dataset, 200.000 for Dutch and 440.000 for German. The accuracy of the inferred types is at 0.65 for English (as compared to 0.86 for LHD 1.0 types). A byproduct of the mapping process is a set of 11.000 mappings from DBpedia resources to DBpedia Ontology classes with associated confidence values. The number of the resulting mappings is an order of magnitude larger than what can be achieved with standard ontology alignment algorithms (Falcon, LogMapLt and YAM++), which do not utilize the type co-occurrence information. The presented algorithm is not restricted to the LHD dataset, it can be used to address generic type inference problems in presence of class membership information for a large number of instances.

Keywords: type inference, ontology alignment, DBpedia, entity classification

1. Introduction

The Linked Hypernyms Dataset (LHD) provides entities described by Dutch, English and German Wikipedia articles with types taken from the DBpedia namespace. It contains 2.8 million entity-type assignments.

The LHD follows the same data modelling approach as the well-known DBpedia (Bizer et al., 2009) and YAGO (Hoffart et al., 2013) knowledge bases. While DBpedia and YAGO are populated from the structured and semistructured information in Wikipedia (article categories and infoboxes), the entries in LHD were extracted using lexico-syntactic patterns from the text of the first sentences of the respective Wikipedia articles. It thus provides a complementary source of type information. For many entities in DBpedia, LHD provides the only type. The dataset is downloadable in the n-triples format, with both the entity and the type from the DBpedia namespace. The raw (plain text) hypernyms are also downloadable.

A certain limitation of the LHD 1.0 dataset is that it is not well connected to Linked Open Data (LOD) cloud, since majority of its types are DBpedia resources. In this paper, we introduce a statistical type inference algorithm for creation of the next version of the dataset, dubbed LHD 2.0, which significantly increases coverage by DBpedia Ontology classes.

The paper is organized as follows. Section 2. describes the LHD extraction framework. Section 3. presents the proposed algorithm for statistical type inference. Section 4. introduces LHD 2.0-draft as the output of this algorithm. Section 5. presents dataset evaluation. Section 6. describes a preliminary attempt to address the mapping problem with sample of standard ontology alignment algorithms. The conclusions point at limitations of the approach in addition to presenting the dataset license and availability. Sketch of future work is also included.

2. LHD 1.0 Extraction Framework

Linked Hypernym Dataset (Kliegr, 2013) associates DBpedia resources (corresponding to Wikipedia articles) with a type which is obtained by parsing the first sentences of the respective Wikipedia article. The type is initially a plain text string, which is further disambiguated to a DBpedia resource creating a "linked hypernym". The LHD 1.0 extraction framework also makes a naive attempt to associate the linked hypernym with a DBpedia Ontology class, which increases the semantic interoperability of the dataset.

The remainder of this section briefly describes the individual steps of the LHD 1.0 extraction framework. An overview of the resulting LHD 1.0 dataset in terms of size and accuracy is given in Table 1. A more detailed description the framework as well as additional size and evaluation metrics are available in (Kliegr, 2013).

2.1. Hypernym Discovery

The extraction framework is implemented on top of GATE.[1] The core of the system is a JAPE transducer (a GATE component) which applies lexico-syntactic patterns encoded as grammar in the JAPE language on the first sentence of Wikipedia articles. The extraction grammars require that the input text is tokenized and assigned part-of-speech (POS) tags. For English, the framework relies on the ANNIE POS Tagger, available in GATE, for German and Dutch on TreeTagger.[2] Extraction grammars were hand-coded using a development set of 600 manually annotated articles per language. The evaluation of the extraction accuracy was performed on a different set of manually annotated articles as reported in Section 5.

[1] http://gate.ac.uk

[2] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

Example 1.
An example input for this phase is the first sentence of Wikipedia article on Václav Havel: *Havel was a Czech playwright, essayist, poet, dissident and politician.* The output is the word "playwright", the first hypernym in the sentence.

The output of the hypernym discovery phase is provided as a separate dataset providing plain text, not disambiguated hypernyms. The accuracy for this dataset (denoted as "plain") is reported in Table 1.

2.2. Linking Hypernyms to DBpedia Instances

Once the hypernym has been extracted from the article, it is disambiguated to a DBpedia identifier. The disambiguation algorithm relies on the Wikipedia Search API to resolve the string to a Wikipedia article.

Example 2.
Picking up on the Václav Havel entity, the word "playwright" is used as a query, which returns the Wikipedia article `http://en.wikipedia.org/wiki/Playwright`. This is then translated to the DBpedia URI `http://dbpedia.org/resource/Playwright`.

While this disambiguation approach is simple, it is effective as confirmed both by our evaluation reported in Section 6 and by the recent results of the NIST TAC 2013 (Dojchinovski et al., 2013). In the TAC *English Entity Linking Evaluation* task this algorithm performed at median F1 measure (overall).

2.3. Alignment with the DBpedia Ontology

While formally, the output of the linking phase is already a LOD identifier, the fact that the type is in the `http://dbpedia.org/resource/` namespace (further referenced by prefix `dbpedia`) is not ideal. Concepts from this namespace are typically instances, while this term is used as a class within the LHD dataset:

```
dbpedia:Václav_Havel rdf:type
dbpedia:Playwright.
```

DBpedia already contains a predefined set of classes within the DBpedia Ontology namespace `http://dbpedia.org/ontology/` (further abbreviated as `dbo`) such as `dbo:Person` or `dbo:Work`. The focus of the alignment phase is to map the original type, which is in the `dbpedia` namespace, to the `dbo` namespace.

The mappings are generated using a simple string matching algorithm, which requires total match in concept name (`dbpedia:Person` → `dbo:Person`). For these *exact match* mappings, only the `dbo:` type is output by the generation process.

Additional mappings are generated using substring match (`dbpedia:Catfish` → `dbo:Fish`). These *approximate* mappings were subject to manual verification. The original (`dbpedia:`) type is preserved on the output, with the classes mapping (`dbpedia:` → `dbo:`) available in a separate file.

This simple mapping approach provides a mapping to the DBpedia Ontology for a large number of entities across all three languages. However, in relative terms, this is less than 50% for each language in LHD 1.0. Table 1 gives an overview of LHD 1.0 dataset in terms of size and accuracy for individual languages.

3. Statistical Type Inference (STI)

The STI algorithm provides mapping from most DBpedia resources used as types in LHD 1.0 to classes in the DBpedia Ontology namespace.

Example 3.
Entity Václav Havel has type `http://dbpedia.org/resource/Playwright` in LHD 1.0. STI assigns this entity with additional type `http://dbpedia.org/ontology/Writer`.
It should be noted that the result of type inference is not always a superclass of the original query hypernym hyp_R as in this example.

The inference algorithm builds upon the fact that for many entities with LHD 1.0 types a set of types in the DBpedia Ontology namespace can be obtained from DBpedia (sourced from the infoboxes).

The intuition is that for a specific LHD 1.0 type hyp_R, we identify DBpedia-assigned types that are co-occurring with hyp_R and we note the number of entities on which each of these types co-occurs with hyp_R. The challenge is to identify out of the pool of the correlated types the best one: the type which is the most specific true type for entities assigned with hyp_R as a type.

Our attempt to address this challenge is comprised by two successive algorithms. Candidate Generation (Alg. 1) generates a set of *Candidates* (DBpedia Ontology classes) for the hypernym hyp_R (DBpedia resource). *Candidates* are then pruned by removing classes for which a more specific one exists, and from the remainder the class with the highest number of supporting entities is selected. The candidate pruning and the selection of the best candidate is performed according to Alg. 2.

A detailed description of the two algorithms follows. The Candidate Generation algorithm first identifies the set E_h^{lhd} that contains entities which have as a type in LHD 1.0 the query hypernym hyp_R which should be mapped. Subsequently, a list of distinct DBpedia Ontology types these entities have in DBpedia is saved along with the number of entries in E_h^{lhd}, which have this specific type. The output is a set $Candidates = \{\langle hyp_O, supp\rangle\}$, each class hyp_O associated with the number of supporting entities $supp$.

Table 1: LHD 1.0 size and accuracy. The `dbo` column indicates the portion of entities in LHD with type from the DBpedia Ontology namespace.

language	linked (total)	linked `dbo`	acc linked	acc plain
German	825,111	171,847	0.773	0.948
English	1,305,111	513,538	0.857	0.951
Dutch	664,045	78,778	0.884	0.933

Example 4. For entity Václav Havel, the set E_h^{lhd} contains 961 entities with `dbpedia:Playwright` as a type in LHD 1.0. Skipping entities without any type in DBpedia 3.8 or with a type not in the DBpedia Ontology namespace, the list of the following types associated with these entities (and associated numbers of supporting entities) is obtained from DBpedia 3.8 instance file: MusicalArtist:5, Politician:1, OfficeHolder:4, Writer:150, Scientist:1, Agent:277, MilitaryPerson:1, MemberOfParliament:1, Artist:157, Person:277.

The selection process is two stage. First, the algorithm iterates through the candidates removing those which are, as indicated by the numbers of supporting instances, only a superclass of a more specific type on the list of $Candidates$. Higher number of supporting entities implies reliability, however, the specific types tend not to have the highest values. The compromise between specificity and reliability is reflected by the value of the *tradeoff* constant used in Alg. 2. Type hyp_O is removed if there is its subclass hyp'_O in *Candidates*, which has at least *tradeoff* * *supp* supporting entities, where *supp* refers to the support of hyp_O.

Algorithm 1 Candidate Generation

Require: hyp_R - a linked hypernym (DBpedia resource) to be mapped, *LHD* = $\{\langle entity, type\rangle\}$, *DBpedia* = $\{\langle entity, type\rangle\}$
Ensure: *Candidates* = $\{\langle hyp_O, supp\rangle\}$ – set of candidate mappings for hyp_R: hyp_O is a DBpedia Ontology class, $supp$ is the number of supporting entities

```
Candidates := ∅
E^lhd := set of entities from LHD with hyp_R as type
for entity∈ E_h^lhd do
    types := set of types entity has in DBpedia
    for type∈ types do
        if type is not DBpedia ontology class then
            continue
        end if
        if type ∉ Candidates then
            add ⟨ type, 1⟩ to Candidates
        else
            increment support of type by 1 in Candidates
        end if
    end for
end for
return Candidates
```

Example 5.
Candidate pruning removes Politician, Agent, Artist and Person from the list of candidates. Politician is removed in favour of its subclass MemberOfParliament, which has the same number of supporting entities (one). The latter three types are removed in favour of Writer. While Writer has less supporting entities than Artist, the drop in support is within tolerance of the *tradeoff* constant set at 0.2.
The following types survive pruning: OfficeHolder, Scientist, MemberOfParliament, Writer, MusicalArtist. Finally, the algorithm selects hyp_O^{opt}=Writer as the type with the highest number of supporting instances. The value of *supp*=150 is the confidence value for this mapping.

4. Generating LHD 2.0 with STI

The mappings to DBpedia ontology in LHD 1.0 were achieved by a naive ontology mapping algorithm. For each entity-linked hypernym pair, the algorithm tries to find a DBpedia Ontology concept based on a textual match. As seen from Table 1 this approach did not yield a match for a substantial number of entities, which were left with DBpedia resources as their types. The STI algorithm is used to assign additional type to these entities.
The LHD 2.0-draft is the result of the execution of the STI algorithm on all 791,573 entities in LHD 1.0 with type in

Algorithm 2 Candidate Pruning and Selection

Require: *Candidates* = $\{\langle hyp_O, supp\rangle\}$
Ensure: hyp_O^{opt} – class from DBpedia Ontology

```
discardMade := true
while discardMade do
    discardMade := false
    for {⟨hyp_O, supp⟩} ∈Candidates do
        minSupp = supp * tradeoff;
        if ∃⟨hyp'_O, supp'⟩ ∈ Candidates: hyp_O ≠ hyp'_O,
        hyp'_O subclass of hyp_O, supp' > minSupp
        then
            remove ⟨hyp_O, supp⟩ from Candidates
            discardMade := true
            break
        end if
    end for
end while
return hyp_O^opt: type with highest supp from Candidates
```

Table 2: LHD 2.0 Generation Overview

Input		Output (mapped)
entities	791,573	755,319
distinct classes	20,474	11,117

the `dbpedia` namespace.[3] Out of 20,474 distinct classes these entities had, the STI algorithm generated 11,117 mappings to DBpedia ontology classes. These mappings were used to assign a DBpedia ontology type to 755,319 entities. The STI execution in terms of input and output is summarized by Table 2.

Comparison of LHD 1.0 and LHD 2.0 is described by Table 4. In LHD 2.0, a large part of the types in the `dbpedia` namespace is replaced by types in the `dbo` namespace. It should be emphasized that LHD 1.0 and LHD 2.0 types are complementary. E.g. for Václav Havel, the LHD 1.0 type is `dbpedia:Playwright` and the LHD 2.0 type is `dbo:Writer`.

The selection of the best type, as performed by the STI algorithm, aims at balancing the specificity and reliability of the inferred type. In order to obtain high coverage with specific types, relatively low value of the *tradeoff* parameter (0.2) was used to generate the LHD 2.0-draft dataset.

As seen in Table 3, the mappings with high support values are all correct, while the quality of the mappings deteriorates with decreasing support. The specificity of the mapped types is satisfactory.

The LHD 2.0 dataset associates the type directly with the entity with `rdf:type` relation. The reasons for this modeling choice are as follows. The type inference task is aimed at assigning types to entities that have hyp_R as a type, which is not necessarily the same task as retrieving the most specific true supertype for hyp_R. The role of hyp_R in the STI algorithm is limited to a feature, shared by similar entities, which is used to identify the correlated type. This difference can be illustrated on the following special cases:

STI returns more precise type. LHD associates entities with types based on one-word hypernyms obtained from article free text. If this word is discriminative, i.e. specific to entities of a certain more specific type, the roles in the STI generated mapping, $hyp_R \rightarrow hyp_O^{opt}$, are effectively reversed: hyp_R is subclass of hyp_O^{opt}.

[3]This implies that entities with a DBpedia ontology type found with LHD 1.0 exact match (cf. Subsection 2.2.) were excluded from the STI mapping process.

Table 3: Inferred Mappings $hyp_R \rightarrow hyp_O^{opt}$ (DBpedia resource $\rightarrow$ DBpedia ontology class). Sample from 11,117 mappings sorted by support.

rank	support	`dbpedia`	`dbo`
1	16876	Commune	Settlement
2	13640	Footballer	SoccerPlayer
3	8370	Actress	Person
4	7077	District	Settlement
5	6037	Club	SoccerClub
6	5132	Television program	TelevisionShow
7	4982	Highway	Road
8	4629	Settlement	Settlement
9	4292	Frog	Amphibian
10	4246	Township	Town
11	4038	Singer	MusicalArtist
12	3478	Episode	TelevisionEpisode
13	3236	Townships	Town
14	3155	Suburb	Settlement
15	2869	Musician	MusicalArtist
16	2790	County	AdministrativeRegion
17	2595	Pitcher	BaseballPlayer
18	2472	Character (arts)	ComicsCharacter
19	2467	EP	Album
20	2024	Director	Person
...	...	...	...
1000	21	Contractor	Company
1001	21	Minority	EthnicGroup
1002	21	Peoples	EthnicGroup
1003	21	Cat	Mammal
1004	21	Rail	Station
1005	21	Reggae	MusicalArtist
1006	21	Punk	Band
1007	21	Disney	Film
1008	21	Ray	Fish
1009	21	Rugby	RugbyPlayer
1010	21	Glands	AnatomicalStructure
1011	21	Complication	Disease
1012	21	Superheroes	ComicsCharacter
1013	21	Emcee	MusicalArtist
1014	21	Tortoise	Reptile
1015	21	Communes	Settlement
1016	21	Cutters	Ship
1017	21	Wasp	Insect
1018	21	Passerine	Bird
1019	21	Flycatcher	Bird
1020	20	Garden	Park
...	...	...	...
11111	1	Acetophenone	ChemicalCompound
11112	1	Acetal	ChemicalCompound
11113	1	Absurd	Album
11114	1	Abrams	Album
11115	1	Abizaid	Album
11116	1	Abdul	Album
11117	1	AB	Company

Example 6. Consider entity Diego Maradona. The corresponding entry in LHD 1.0 is
`dbpedia:Diego_Maradona rdf:type dbpedia:Manager`
The STI algorithm yields type `dbo:SoccerManager`. The corresponding LHD 2.0 statement is:
`dbpedia:Diego_Maradona rdf:type dbo:SoccerManager`
The explanation is that entities with the "manager" hypernym (disambiguated to `dbpedia:Manager` linked hypernym) tend to be soccer managers.

STI corrects LHD 1.0 error. The fact that LHD is based on one word hypernyms causes LHD to assign incorrect type if the modifiers before the hypernym alter the meaning. If the LHD 1.0 hypernym is sufficiently discriminative, the STI inferred type hyp_O^{opt} is actually a correct type for most entities with the original linked hypernym hyp_R, although the mapping $hyp_R \rightarrow hyp_O^{opt}$ is incorrect.

Example 7.
STI outputs mapping `dbpedia:Processor` → `dbo:Software`. While this mapping is obviously incorrect, assigning `dbo:Software` to entities with originally `dbpedia:Processor` as a type is correct in many cases. This is due to the fact that a substantial number of entities with `dbpedia:Processor` type are *word* processors.

5. Dataset Quality

Human evaluation of the correctness was performed separately for the entire English, German and Dutch datasets each represented by a randomly drawn 1,000 articles. The evaluation for English were done by three annotators. The evaluation for German and Dutch were done by the best performing annotator from the English evaluation. It can be seen that for LHD 1.0 the accuracy for "plain text" hypernyms exceeds 0.9 for all languages, while the accuracy of the disambiguated linked hypernyms (a mix of concepts in `dbpedia` and `dbo` namespaces) is in the $0.77 - 0.88$ range (Table 1). The English dataset was subject to further analysis, with evaluation results reported for its twelve interesting partitions. The results, guidelines and raw as well as aggregated evaluation data can be found at `http://ner.vse.cz/datasets/linkedhypernyms/`.
LHD 2.0-draft places all types into the `dbo` namespace (ref. to Table 4) for English and a substantial number for German and Dutch. The impact of the mapping algorithm on the accuracy was measured on samples taken from all 755,319 English Wikipedia entities for which STI algorithm output a mapping. These entities were divided to three partitions depending on whether for the given entity the linked hypernym assigned is the same as the type assigned to this entity in DBpedia.
From each partition a sample of 1,000 entities was drawn and manually evaluated by one annotator (a graduate student). The annotation guidelines were the same, which were used for the evaluation for the LHD dataset (Kliegr, 2013). The guidelines, as well as the annotation results, are available on the dataset website.

Table 4: LHD 2.0 vs LHD 1.0 DBpedia Ontology coverage comparison

language	size	`dbo` (1.0)	`dbo` (2.0)
German	825,111	171,847	615,801
English	1,305,111	513,538	1,268,857
Dutch	664,045	78,778	283,626

Table 5: LHD 2.0 Accuracy for English

dataset	size	linked	plain
Overlaps with DBpedia	278,805	0.906	0.913
DBpedia assigns diff. type	69,331	0.350	0.807
Entity not in DBpedia	407,183	0.525	0.939
All, weighted average	755,319	0.650	0.917

Example.
Václav Havel falls into the "DBpedia assigns different type" partition, since `dbo:Writer` is not among the types assigned to this entity in DBpedia.

The resulting accuracy (Table 5) significantly varies by partition. The worst result – accuracy of only 0.35 – is partly due to the fact that already the plain text hypernym extracted by the lexico-syntactic patterns was incorrect. It should be noted that the partition with best accuracy ("Overlaps with DBpedia") does not contribute new information, since the `rdf:type` triples in this partition are already present in DBpedia.

6. Preliminary Comparison with Standard Matching Approaches

Prior to designing the presented STI algorithm, we have attempted to solve the problem with standard ontology matching tools.
As an input, the same set of entities as for LHD 2.0 generation was used: LHD 1.0 entities with type in the `dbpedia` namespace, for which no (even approximate) mapping to DBpedia Ontology was found in LHD 1.0. These 736,000 entities had more than 19,000 distinct types. In order to enable applying ontology matching tools we built simple flat ontology, where each distinct type is a subclass of the most general class owl:Thing. Next, DBpedia ontology 3.8 was used as the target ontology for mapping.[4]
We applied several ontology matching tools which participated in the Ontology Alignment Evaluation Initiative (OAEI).[5] However, some matchers had problems to match those two ontologies. Mostly because of a large size of the

[4] Mappings to schema.org classes, which are also present in the DBpedia ontology, were ignored.
[5] `http://oaei.ontologymatching.org/`

Table 6: Ontology Alignment Results

	input classes	input entities
all	19,421	736,294
tool	mapped classes	mapped entities
Falcon	109	13,720
LogMapLt	45	6,350
YAM++	65	1,978
STI	**10,721**	**701,516**

flat ontology (more than 19,000 classes). Finally, we managed to obtain some results from three ontology matching tools representing recent high-quality ontology matchers: Falcon, LogMapLt and YAM++ (their details are presented below). The number of mapped classes is reported in Table 6. The results indicate that only for a fraction (less than 1%) of the LHD types a matching class in the DBpedia Ontology was found by these tools with a default setting.

Falcon (Hu and Qu, 2008) consists of three matching components. *V-Doc* component matches ontology entities (e.g. classes) in context by comparison of vectors comprising strings of entities to be matched and their neighbours. *I-Sub* component is string based technique considering not only similarity of entity string but also their dissimilarity. Finally *GMO* component inspects structural similarity of two ontologies to be matched.

LogMapLt (Jiménez-Ruiz et al., 2013) is a light-weight version of the LogMap. While LogMap contains components for graph matching and reasoning, LogMapLt only has efficient string matching techniques based on stemming techniques and inverted indexes built from input ontologies.

YAM++ (Ngo and Bellahsene, 2012) contains string based matchers considering strings of entities including their annotations. It also compares structural aspect by considering input ontologies as graphs which enables similarity propagation. In final step, YAM++ checks coherency of resulted mappings with its logical based checking module.

Since the input flat ontology was structurally simple, matchers could not take advantage of their structural oriented components and had to rely on their string based techniques. This is empirically confirmed by a qualitative analysis of the results indicating that the mappings provided by these tools were obtained based on string similarity, lemmatization and a dictionary of synonyms. Falcon, which output the highest number of mappings, could be used to assign a DBpedia Ontology type to less than 2% of the entities with a DBpedia resource as a type. Since all the tools had insufficiently small coverage, no exact evaluation of the quality of the mappings was performed. However, an informal evaluation is presented below.

LogMapLt and YAM++ provided relatively reliable mappings, while for Falcon most mappings were obviously incorrect. In order to illustrate the output of the individual tools, we present randomly selected ten mappings for each tool. The first type comes from the `dbpedia` namespace, the second type from the `dbo` namespace. Sample of results for the STI algorithm are depicted in Table 3. It should be noted that while the STI algorithm naturally aimed at subsumption mappings, ontology matching tools primarily provides equivalence mappings, i.e. $\equiv$ relation.

Falcon: `Worker` $\equiv$ `Work`, `Settler` $\equiv$ `Letter`, `Crickets` $\equiv$ `Cricketer`, `Rape` $\equiv$ `Grape`, `Toll` $\equiv$ `Atoll`, `Railway` $\equiv$ `RailwayLine`, `Count` $\equiv$ `Country`, `Steroid` $\equiv$ `Asteroid`, `Lectins` $\equiv$ `Insect`, `Churches` $\equiv$ `Church`.

LogMapLt: `Artiste` $\equiv$ `Artist`, `Organizations` $\equiv$ `Organisation`, `Planting` $\equiv$ `Plant`, `Libraries` $\equiv$ `Library`, `Deputies` $\equiv$ `Deputy`, `Production` $\equiv$ `Product`, `Chanson` $\equiv$ `Song`, `Spaceflight` $\equiv$ `YearInSpaceflight`, `Route` $\equiv$ `Road`, `Activities` $\equiv$ `Activity`.

YAM++: `Deputies` $\equiv$ `Deputy`, `Constelation` $\equiv$ `Constellation`, `Taxes` $\equiv$ `Tax`, `Modeller` $\equiv$ `Model`, `Projection` $\equiv$ `Project`, `Paint` $\equiv$ `Painting`, `Naming` $\equiv$ `Name`, `Volcanoes` $\equiv$ `Volcano`, `Elect` $\equiv$ `Election`, `Athletics` $\equiv$ `Athlete`.

While it is clear that the proposed STI algorithm provides a marked improvement in terms of coverage over the three standard ontology matching tools engaged, it should be noted that the corresponding tools consumed different inputs. The STI algorithm used as the only input instances of the target ontology classes, while the other three tools did not work with this information. Therefore, it is worth considering instance based ontology matchers in future. On the other hand, our algorithm did not perform any string matching or dictionary search.

7. Conclusion and Future Work

This paper introduced the Linked Hypernyms Dataset, to the best of our knowledge the most comprehensive effort to supplement DBpedia and YAGO ontologies with types extracted from plain text of Wikipedia articles. The presented work on LHD 2.0-draft is an effort to improve the linkage of this dataset to the LOD cloud by mapping the hypernyms to DBpedia Ontology classes using a statistical co-occurrence based algorithm.

Through the proposed algorithm, the number of entities in LHD with a type from the DBpedia Ontology is increased significantly: by 750 thousand entities for the English dataset, 200.000 for Dutch and 440.000 thousand for German. For the English dataset, the coverage with DBpedia Ontology classes exceeds 95%.

At 0.65 for English, the accuracy of the inferred types is significantly lower than is the average accuracy of 0.86 for LHD 1.0 types. More research, possibly complemented by manual refinement of the mappings, is required to reach the original level accuracy maintaining the types in the DBpedia Ontology namespace. Currently, the algorithm takes on the input only one type for entity, which is a type assigned by LHD, while for some entities additional types are available in DBpedia and YAGO ontologies. Coverage could be further improved, if these additional types are used in the candidate generation phase, and accuracy improved if they are used in the candidate pruning/selection phase. Next, it is also worth inspecting how the STI algorithm could benefit from traditional ontology matching techniques. This would need more thorough analysis of available ontology

matching tools/techniques.

In this paper, we focused on evaluation of LHD 2.0-draft in terms of its possible use for enrichment of the DBpedia knowledge-base. As a future work, it would be interesting to investigate the complementarity of the dataset, as well as of the algorithmic approach, with Babelnet (Navigli and Ponzetto, 2012). Babelnet is a large semantic knowledge base, which integrates lexicographic and encyclopedic knowledge from WordNet and Wikipedia.

The Linked Hypernyms dataset (both LHD 1.0 and LHD 2.0-drat) is downloadable from `http://ner.vse.cz/datasets/linkedhypernyms/`. The dataset is released under the Creative Commons license. Additionally, the set of 11.000 mappings from DBpedia resources to DBpedia Ontology classes (as exemplified by Table 3) with associated confidence values was also made available.

The LHD 1.0 and 2.0-draft were generated from the following Wikipedia snapshots: December 1st, 2012 (German), October 11th, 2012 (Dutch), September 18th, 2012 (English). The LHD dataset is used by the `Entityclassifier.eu` wikifier (Dojchinovski and Kliegr, 2013).

Acknowledgements

This research was supported by the European Union's 7th Framework Programme via the LinkedTV project (FP7-287911).

8. References

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia-a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165.

Milan Dojchinovski and Tomas Kliegr. 2013. Entityclassifier.eu: Real-time classification of entities in text with Wikipedia. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, ECMLPKDD'13, pages 654–658. Springer-Verlag.

Milan Dojchinovski, Tomáš Kliegr, Ivo Lašek, and Ondřej Zamazal. 2013. Wikipedia search as effective entity linking algorithm. In *Text Analysis Conference (TAC) 2013 Proceedings*. NIST. To appear.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61.

Wei Hu and Yuzhong Qu. 2008. Falcon-AO: A practical ontology matching system. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):237–239.

Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Ian Horrocks. 2013. LogMap and LogMapLt results for OAEI 2013. *Ontology Matching*.

Tomáš Kliegr. 2013. Linked hypernyms: Enriching DBpedia with Targeted Hypernym Discovery. Submitted.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193(0):217 – 250.

DuyHoa Ngo and Zohra Bellahsene. 2012. YAM++: a multi-strategy based approach for ontology matching task. In *Knowledge Engineering and Knowledge Management*, pages 421–425. Springer.