Automatic typing of DBpedia entities

Aldo Gangemi[1], Andrea Giovanni Nuzzolese[1,2], Valentina Presutti[1]
Francesco Draicchio[1], Alberto Musetti[1], and Paolo Ciancarini[1,2]

[1] STLab-ISTC Consiglio Nazionale delle Ricerche, Rome, Italy.
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy.

Abstract. We present *Tìpalo*, an algorithm and tool for automatically typing DBpedia entities. Tìpalo identifies the most appropriate types for an entity by interpreting its natural language definition, which is extracted from its corresponding Wikipedia page abstract. Types are identified by means of a set of heuristics based on graph patterns, disambiguated to WordNet, and aligned to two top-level ontologies: WordNet supersenses and a subset of DOLCE+DnS Ultra Lite classes. The algorithm has been tuned against a golden standard that has been built online by a group of selected users, and further evaluated in a user study.

1 Introduction

Wikipedia is a large-scale resource of content capturing encyclopedic knowledge collaboratively described by the crowds. Entities described in Wikipedia are formally represented in DBpedia, the RDF translation of information from many localized versions of Wikipedia, eminently the English one. There are DBpedia datasets providing types for entities, but a large number of them is still untyped, or has a very specialized type, and types are taken from ontologies that have heterogenous granularities or assumptions (e.g., 272 infobox-based types in the DBpedia ontology (DBPO)[3] against almost 290,000 category-based in YAGO [17]). This situation makes it difficult to identify a proper reference ontology for Wikipedia, or to reuse DBpedia knowledge with a good precision. While it is reasonable to have limited semantic homogeneity on the Web, it is highly desirable to bring a more organized and complete typing to DBpedia entities. Knowing *what* a certain entity is, e.g., a person, organization, place, instrument, etc., is key for enabling a number of desirable functionalities such as *type coercion* [10], data pattern extraction from links [14], entity summarization (cf. Google Knowledge Graph), automatic linking, etc.

The two *de facto* reference ontologies for DBpedia resources are currently DBPO and YAGO. Both provide types for DBpedia entities, and in particular YAGO has high performances as far as typing quality is concerned. However, their coverage is partial, both extensionally (number of typed resources), and intensionally (conceptual completeness), since they rely on Wikipedia categories, and infoboxes (that are not included in all Wikipedia pages). In addition, the number of resources that could be typed from Wikipedia content is even larger than the number of Wikipedia pages: for example, many DBpedia entities are

[3] http://dbpedia.org/ontology/

referenced by fragments of Wikipedia pages. Our aim is to enable automatic typing of entities by exploiting the natural language (NL) definitions from their corresponding Wikipedia pages. Hence, without relying on categorization, or on the presence of structured data such as Wikipedia infoboxes.

Although there are numerous Natural Language Processing (NLP) approaches to learning ontologies from text, they need training phases that can take a long time, and may need a huge manually annotated corpus in order to perform training. When dealing with large-scale corpora such as Wikipedia, we need to identify sustainable procedures as far as speed is concerned. Therefore, none of the existing NLP resources (cf. Section 2) can be directly used to perform the automatic typing of DBpedia entities that we propose here.

We present *Tìpalo*, a tool that automatically assigns types to DBpedia entities based on their NL definitions as provided by their corresponding Wikipedia pages. We use a tool (cf. FRED [16]) that implements deep parsing methods based on frame semantics for deriving RDF and OWL representations of NL sentences. On top of it, we have implemented a procedure to extract types from the RDF representation of definitions. The procedure is tailored to Wikipedia pages, and reuses a tool for word sense disambiguation (cf. UKB [1]) to automatically link the extracted types to WordNet. We also use alignments of WordNet to two top-level ontologies: WordNet *super senses*, and DUL+DnS Ultralite[4].

Results show that Tìpalo can extend typing of DBpedia entities with high accuracy, and support the incremental definition of a Wikipedia ontology that emerges from what is written in the articles, rather than from metadata or statistical observations.

The contribution of this paper can be summarized as follows:

- a tool, named *Tìpalo*, implementing a process for automatically typing DBpedia entities, based on their NL definitions, which is fast enough to be used in realistic projects, while performing with good precision and recall;
- a sample Wikipedia ontology, incrementally built with Tìpalo, encoded in two semantic resources: (i) the Wikipedia entity types dataset, containing automatically typed (and evaluated) DBpedia entities extracted from 627 definitions; (ii) the Wikipedia class taxonomy dataset, including WordNet types, WordNet super senses, DUL types, and new defined types put in a `rdfs:subClassOf` taxonomy;
- an updated mapping between WordNet 3.0 synsets and top-level ontology classes, released as RDF datasets;
- a golden standard of 100 typed entities, manually annotated through a collaborative effort supported by an online tool that manages user agreement.

The whole procedure can be executed by means of a web service[5]. In order to favor reuse and repeatability of our approach and experiments, the web service, tools, and resources that we have produced are all publicly available from the Wikipedia Ontology page[6].

[4] `http://www.ontologydesignpatterns.org/ont/dul/DUL.owl`

[5] `http://wit.istc.cnr.it/stlab-tools/tipalo`

[6] The Wikipedia ontologypage `http://www.stlab.istc.cnr.it/WikipediaOntology/`

2 Related work

The main approaches for typing DBpedia entities are: (i) the DBpedia project [11], which manually created a DBpedia ontology (DBPO) based on a limited number of Wikipedia infobox templates. Currently, DBpedia entities having DBPO types are ∼1.83M[7] (version 3.7), against almost 4M Wikipedia pages (August 2012). Besides its limited extensional coverage, DBPO suffers from limited intensional coverage [15] due to the manual extraction procedure on infobox templates that exist only for a subset of Wikipedia page types; (ii) YAGO [17], an ontology extracted from Wikipedia categories and infoboxes, and aligned to a subset of WordNet. YAGO's coverage is larger than DBPO (∼2.7M entities), however still incomplete and its intensional completeness is affected by its reliance on Wikipedia infoboxes and categories. In [15] learning techniques as well as rule-based approaches for automatic typing of DBpedia entities have been analyzed. The analysis confirmed the difficulty of this task, and highlighted the limits posed by the reliance on Wikipedia categories and infoboxes.

Relevant work related to our method includes Ontology Learning and Population (OL&P) techniques [2]. Typically OL&P is implemented on top of machine learning methods, hence it requires large corpora, sometimes manually annotated, in order to induce a set of probabilistic rules. Such rules are defined through a training phase that can take a long time. Examples of such methods include [3,20,18]. All these methods would be hardly applicable to large corpora such as Wikipedia due to the time and resources they require, if all the potential types present in NL descriptions need to be extracted. Other approaches to OL&P use either lexico-syntactic patterns [9], or hybrid lexical-logical techniques [19], but to our knowledge no practical tools have emerged so far for doing it automatically while preserving high quality of results. [5] works great for large-scale information extraction centered on binary relation extraction. However, its resulting triplet graphs are not interlinked and feature a low recall of relevant syntactic structures, making it too limited for the automatic typing task.

The method presented in this paper differs from most existing approaches, by relying on a component named FRED [16], which implements a logical interpretation of NL represented in Discourse Representation Theory (DRT). FRED is fast and produces an OWL-based graph representation of an entity description, including a taxonomy of types. We parse FRED's output graph, and apply a set of heuristics, so that we can assign a set of types to an entity in a very efficient way. FRED is an example of *machine reading*.

Terms used for describing entity types are often polysemous i.e. they can have more than one meaning. We have empirically observed (on a sample of ∼800 entity definitions) that polysemous terms occur in 70% of descriptions; hence, the word sense disambiguation (WSD) task is relevant in this context. A good survey of WSD methods is [12]. We use UKB [1] that shows high accuracy, with some impact on the speed of the process (cf. Section 5). A promising resource for disambiguation is BabelNet [13], which has produced a substantial alignment

[7] M → millions

between WordNet and Wikipedia concept-like entities. However, its currently available dataset is not suitable for implementing a direct WSD functionality.

As a final remark, we mention a recent work on terminology extraction [8] which describes a number of wikipedia markup conventions that are useful hooks for defining heuristics. Some of them have been reused in our tool.

3 Data sources

In the context of this work, we have used and produced a number of resources.

Wikipedia and DBpedia. Wikipedia is a collaboratively built multilingual encyclopedia on the Web. Each Wikipedia page usually refers to a single entity, and is manually associated to a number of categories. Entities referenced by Wikipedia pages are represented in DBpedia, the RDF version of Wikipedia. Currently, English Wikipedia contains 4M articles[8], while DBpedia wikilink dataset counts ∼15M distinct entities (as of version 3.6). One main motivation of this big difference in size is that many linked resources are referenced by *sections* of Wikipedia pages, hence lacking explicit categorization or infoboxes. However they have a URI, and a NL description, hence they are a rich source for linked data. Out of these ∼15M resources, ∼2.7 are typed with YAGO classes and ∼1.83M are typed with DBpedia classes. We use Wikipedia page contents as input for the *definition extractor* component (cf. Section 4), for extracting entity definitions.

WordNet 3.0 and WordNet 3.0 supersense RDF. WordNet[9] is a large database of English words. It groups words into sets of synonyms, called *synsets*, each expressing a different concept. Although WordNet includes different types of words such as verbs and adjectives, for the sake of this work we limit the scope to nouns. Words that may express different meanings, i.e. polysemous words, are related to different synsets. In this work, we use the WordNet 3.0 RDF porting[10] in order to identify the type of an entity. Hence when such type is expressed by a polysemous word we need to identify the most appropriate one. To this aim we exploit a WSD engine named UKB, as described in Section 4. Furthermore, WordNet 3.0 includes relations between synsets and supersenses, which are broad semantic categories. WordNet contains 41 supersenses, 25 of which are for nouns. We have produced a resource named *WordNet 3.0 Supersense RDF*[11] that encodes such alignments as RDF data. This RDF dataset is used by the *type matcher* (cf. Section 4) for producing triples relating entities and supersenses.

OntoWordNet (OWN) 2012 is a RDF resource that updates and extends OWN[7]. OWN is an OWL version of WordNet, which includes semantic alignments between synsets and DULplus types. DULplus[12], extends DUL[13], which

[8] Source: http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia, Aug 2012

[9] WordNet, http://wordnet.princeton.edu/

[10] http://semanticweb.cs.vu.nl/lod/wn30/

[11] http://www.ontologydesignpatterns.org/wn/wn30/wordnet-supersense.rdf

[12] Dolce Ultra Lite Plus ontology, http://www.ontologydesignpatterns.org/ont/wn/dulplus.owl

[13] Dolce Ultra Lite ontology, http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

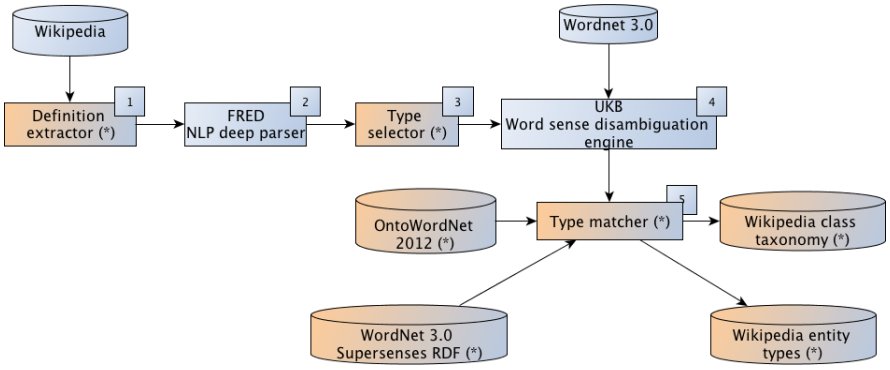

Fig. 1: Pipeline implemented by Tìpalo for automatic typing of DBpedia entities based on their natural language descriptions as provided in their corresponding Wikipedia pages. Numbers indicate the order of execution of a component in the pipeline. The output of a component i is passed as input to the next $i+1$ component. (*) denotes datasets and tools developed in this work, which are part of our contribution (cf. Section 3).

is the OWL light version of DOLCE + DnS [6] foundational ontology. OWN 2012 contains mappings between 859 general synsets and 60 DULplus classes. Such mappings have been propagated through the transitive closure of the hyponym relation in order to cover all ∼82000 synsets. In the context of this work, we have updated OWN to the WordNet 3.0 version, and performed a revision of the manual mapping relations. Furthermore, we have defined a lightweight foundational ontology called Dolce Zero[14], whose classes generalize a number of DULplus classes used in OWN. We have used a combination of 23 Dolce Zero and DULplus classes for building a sample Wikipedia ontology. The reduction to 23 classes has been made in order make it comparable to the WordNet supersense set, and to simplify the task of evaluators.

4 The automatic typing procedure

Tìpalo is based on a pipeline of components and data sources, described below, which are applied in the sequence illustrated in Figure 1.

1. Extracting definitions from Wikipedia pages (definition extractor). The first step, performed by the **definition extractor**, consists in extracting the definition of a DBpedia entity from its corresponding Wikipedia page abstract. We identify the shortest text including information about the entity type. Typically, an entity is defined in the first sentence of a Wikipedia page abstract, but sometimes the definition is expressed in one of the following sentences, can be a combination of two sentences, or even implicit. We rely on a set of heuristics based on lexico-syntactic patterns and Wikipedia markup conventions in order to extract such sentences. A useful Wikipedia convention is the use of bold characters for visualizing the name of the referred entity in the page abstract: for example consider the Wikipedia page referring to "Vladimir

[14] `http://www.ontologydesignpatterns.org/d0.owl`

Fig. 2: First paragraph of the Wikipedia page abstract for the entity "Vladimir Kramnik".

Kramnik"[15] and the first paragraph of its abstract, depicted in Figure 2. Let us represent such paragraph as a sequence of n sentences $\{s_1, ..., s_n\}$. Typically, the bold words referring to the entity (*bold-name*) are included in a sentence $s_i, (i = 1, ..., n)$ that provides its definition according to a syntactic form of the type: "*bold–name <copula><predicative nominal||predicative adjective>*" (where *<copula>* is usually a form of the verb *to be*) e.g., "**Vladimir Borisovich Kramnik** is a Russian chess grandmaster". However, this is not always the case: sometimes, the sentence s_i containing the *bold-name* does not include any *<copula>*, while a *<copula>* can be found together with a co-reference to the entity, in one of the following sentences s_j. In such cases, we extract the entity definition by concatenating these two sentences i.e. $s_i + s_j$. If the abstract does not contain any *bold-name*, we inspect s_1: if it contains a *<copula>* we return s_1, otherwise we concatenate s_1 with the first of the next sentences e.g., s_i, containing a *<copula>* (i.e. $s_1 + s_i$). If none of the above is satisfied, we return s_1. We also apply additional heuristics for dealing with parentheses, and other punctuation. For the example in Figure 2 we return s_1: *Vladimir Borisovich Kramnik is a Russian chess grandmaster*, which contains the *bold-name* as well as a *<copula>*.

2. Natural language deep parsing of entity definitions (FRED). Once the entity definition has been extracted, it should be parsed and represented in a logical form that includes a set of types. In order to accomplish this task we use **FRED**[16], a tool that we have presented in [16]. It performs ontology learning by relying on Boxer [4], which implements *computational semantics*, a deep parsing method that produces a logical representation of NL sentences in DRT. FRED implements an alignment model and a set of heuristics for transforming DRT representations to RDF and OWL representations. In the context of this work, FRED is in charge of "reading" an entity NL definition, and producing its OWL representation, including a taxonomy of types. For example, given the above definition for **Vladimir Kramnik**, FRED returns the OWL graph depicted in Figure 3, containing the following taxonomy[17].

```
wt:RussianChessGrandmaster rdfs:subClassOf wt:ChessGrandmaster
wt:ChessGrandmaster rdfs:subClassOf wt:Grandmaster
```

[15] http://en.wikipedia.org/wiki/Vladimir_Kramnik

[16] FRED is available online at http://wit.istc.cnr.it/stlab-tools/fred

[17] wt: http://www.ontologydesignpatterns.org/ont/wikipedia/type/

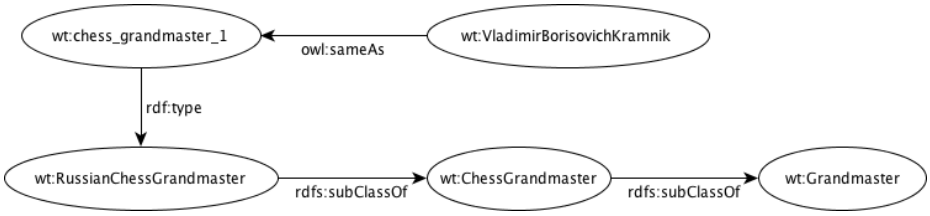

Fig. 3: FRED result for the definition "Vladimir Borisovich Kramnik is a Russian chess grandmaster."

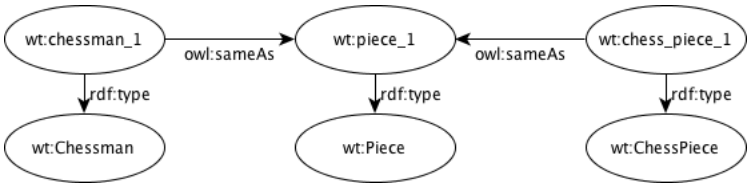

Fig. 4: FRED result for the definition "Chess pieces, or chessmen, are the pieces deployed on a chessboard to play the game of chess."

3. Selection of types and type-relations from the OWL graph (type selector). This step requires to identify, in FRED output graph, the paths providing typing information about the analyzed entity, and to discard the rest. Furthermore, we want to distinguish the case of an entity that is represented as an individual e.g. **Vladimir Kramnik**, from the case of an entity that is more appropriately represented as a class e.g., **Chess piece**. FRED output looks differently in these two situations as well as depending on the type of definition e.g., including a *copula* or parenthetic terms. For example, consider the entity **Chess piece**, which is a class entity, and is defined by *"Chess pieces, or chessmen, are the pieces deployed on a chessboard to play the game of chess."*. FRED output graph for such definition is depicted in Figure 4[18]. In this case, the graph paths encoding typing information comply with a different pattern from the one in Figure 3. The role of the type selector is to recognize a set of graph patterns that allow to distinguish between an entity being a class or an individual, and to select the concepts to include in its graph of types. To implement the type selector, we have identified a set of graph patterns (GP), and defined their associated heuristics by following similar criteria as lexico-syntactic patterns [9], extended with the exploitation of RDF graph topology and OWL semantics. Currently, we use 10 GPs: 4 of them identifying class entities, and 6 for individual entities. Firstly, the type selector distinguishes if an entity is either an individual or a class entity: given an entity e, it is an individual if it participates in a graph pattern of type e `owl:sameAs` x, it is a class if it participates in a graph pattern of type x `rdf:type` e. As empirically observed, these two situations are mutually exclusive. After performing this distinction, the

[18] For space reasons, we include only the portion of the graph of interest in this context. Readers interested in visualizing the complete graph can submit the sentence to FRED online `http://wit.istc.cnr.it/stlab-tools/fred`.

ID	graph pattern (GP)	inferred axioms
gp_1	e owl:sameAs x && x domain:aliasOf y && y owl:sameAs z && z rdf:type C	e rdf:type C
gp_2	e rdf:type x && x owl:sameAs y && y domain:aliasOf z && w owl:sameAs z && w rdf:type C	e rdf:type C
gp_3	e owl:sameAs x && x [r] y && y rdf:type C	e rdf:type C
gp_4	e owl:sameAs x && x rdf:type C	e rdf:type C
gp_5	e dul:associatedWith x && x rdf:type C	e rdf:type C
gp_6	(e owl:sameAs x && x anyP y && y rdf:type C) ‖ (e anyP x && x rdf:type C)	e rdf:type C

Table 1: Graph patterns and their associated type inferred triples for individual entities. Order reflects priority of detection. [r] $\in R =$ {wt:speciesOf, wt:nameOf, wt:kindOf, wt:varietyOf, w:typeOf, wt:qtyOf, wt:genreOf, wt:seriesOf}); [anyP] $\in \{*\} - R$.

type selector follows a priority order for GP detection and executes the heuristics associated with the first matching GP. Tables 1 and 2 respectively report the GP sets and their associated heuristics by following the priority order used for detection, for individual entities and class entities.

ID	graph pattern (GP)	inferred axioms
gp_7	x rdf:type e && x owl:sameAs y && y [r] z && z rdf:type C	e rdfs:subClassOf C
gp_8	x rdf:type e && x owl:sameAs y && y rdf:type C	e rdfs:subClassOf C
gp_9	x rdf:type e && e dul:associatedWith y && y rdf:type C	e rdfs:subClassOf C
gp_{10}	(x rdf:type e && x owl:sameAs y && y [anyP] z && z rdf:type C) ‖ (x rdf:type e && y [anyP] x && y rdf:type C)	e rdfs:subClassOf C

Table 2: Graph patterns and their associated type inferred triples for class entities. [r] $\in R =$ {wt:speciesOf, wt:nameOf, wt:kindOf, wt:varietyOf, w:typeOf, wt:qtyOf, wt:genreOf, wt:seriesOf}); [anyP] $\in \{*\} - R$.

The rationale behind GP priority order resides in ontology design choices as well as in the way the current implementation of the type selector works. Sometimes, an entity definition from Wikipedia includes typing information from a "domain-level" as well as a "meta-level" perspective. For example, from the definition[19] *"Fast chess is a type of chess game in which each side is given less time to make their moves than under the normal tournament time controls of 60 to 180 minutes per player."* we can derive that "Fast chess" is a *type* (meta-level type) as well as a *chess game* (domain-level type). This situation makes FRED output include a GP detecting "type" as a type i.e., gp_8, as well as a GP detecting "chess game" as a type i.e., gp_7, as depicted in Figure 5. In this version of Tìpalo our goal is to type DBpedia entities only from a domain-level perspective. Furthermore, in its current implementation, the type selector executes only one heuristics: that associated with the first GP that matches in FRED output graph. Given the above rationale, gp_7 is inspected before gp_8. The same rationale applies to GP for individual entities, illustrated in Table 1. For the `dbp:Fast_chess`[20] example, the type selector detects that the entity is a class and the first GP detected is gp_7, hence it produces the additional triples:

```
dbp:Fast_chess rdfs:subClassOf wt:ChessGame
wt:ChessGame rdfs:subClassOf wt:Game
```

The execution of Tìpalo pipeline on a sample set of randomly selected ∼800 Wikipedia entities[21] has shown that the most frequent GPs are gp_4 and gp_8, which is not surprising, since they are the most common linguistic patterns for definitions. Table 3 reports the frequency of each GP on the sample set.

[19] http://en.wikipedia.org/wiki/Fast_chess

[20] dbp: http://dbpedia.org/resource/

[21] Details about the definition of the sample set are given in Section 5.

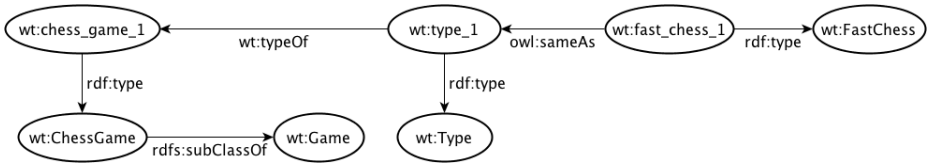

Fig. 5: FRED result for the definition "Fast chess is a type of chess game in which each side is given less time to make their moves than under the normal tournament time controls of 60 to 180 minutes per player."

GP	**frequency** (%)	**GP**	**frequency** (%)	**GP**	**frequency** (%)	**GP**	**frequency** (%)	**GP**	**frequency** (%)
gp_1	0	gp_2	0.15	gp_3	3.98	gp_4	79.34	gp_5	0
gp_6	0.31	gp_7	1.11	gp_8	11.46	gp_9	0	gp_{10}	3.65

Table 3: Normalized frequency of GPs on a sample set of ∼800 randomly selected Wikipedia entities.

The *type selector* implements an additional heuristics: it detects if any of the terms referring to a type in the graph can be referenceable as a DBpedia entity. For example, the term "chess" in the definition of "Fast chess" is resolvable to `dbp:Chess`. In such case, the type selector produces the following triple:

```
dbp:Fast_chess rdfs:subClassOf dbp:Chess
```

This additional heuristics improves the internal linking within DBpedia, resulting in higher cohesion of the resource graph.

By following the defined heuristics, we are able to select the terms that refer to the types of an entity e, and to create a namespace of Wikipedia types that captures the variety of terms used in Wikipedia definitions[22].

4. Word sense disambiguation engine (UKB). After having identified the concepts expressing the types of an entity and their taxonomical relations, we have to gather their correct sense: we need a WSD tool. A possible way to achieve this goal is to identify alignments between the type terms and WordNet terms. We have approached this issue by applying two alternative solutions on a sample set of 100 entity definitions. The first approach involves UKB [1], a tool which returns the WordNet synset for a term, looking for the one that fits best the context given by the entity definition. UKB provids good results in terms of precision and recall although its speed performance needs improvement in order to apply it on a large dataset such as Wikipedia. We have plans for dealing with this issue in the next future (cf. Secion 5.1). The second solution is to select the most frequent WordNet sense for a given term, which is very efficient in terms of speed, but shows lower precision and recall[23]. This step allows us to assign a WordNet type (corresponding to the identified synset) to an entity. Referring to the above example (i.e., definition of fast chess), we produce the following additional triples[24]:

```
wt:ChessGame owl:equivalentTo wn30syn:synset-chess-noun-2
wt:Game owl:equivalentTo wn30syn:synset-game-noun-1
```

[22] Wikipedia class taxonomy, wt: = `http://www.ontologydesignpatterns.org/ont/wikipedia/type/`

[23] The current implementation of Tìpalo relies on UKB for word sense disambiguation.

[24] wn30syn: = `http://purl.org/vocabularies/princeton/wn30/instances/`

5. Identifying other Semantic Web types (type matcher). So far the typing process produces a set of newly defined concepts, and disambiguates them to a WordNet sense. The final step consists in linking such concepts to other Semantic Web ontologies, in order to support shared interpretation and linked data enrichment. In order to exemplify this task with respect to the goal of gathering top-level types[25], we have produced and published two RDF datasets (see Section 3) containing alignments between WordNet synsets and Super Senses (broad lexico-semantic categories), and between WordNet synsets and some foundational ontology classes. The *type matcher* exploits these alignments in order to produce additional `rdf:type` triples. For example, for the entity `dbp:Fast_chess`, the type matcher produces e.g. the following triples:

```
wt:ChessGame rdfs:subClassOf d0:Activity
wt:Game rdfs:subClassOf wn30:supersense-noun_act
```

meaning that the term "chess game" associated with the WordNet sense `wn30syn:synset-chess-noun-2` (as provided by the WSD component) is aligned to the class `Activity` of Dolce Zero[26] ontology. Analogously, the term "game" with its sense `wn30syn:synset-game-noun-1` is aligned to the WordNet super sense "act".

The described five steps compose the Tìpalo automatic typing procedure, whose output feeds incrementally a Wikipedia ontology based on the entity definitions provided by the crowds, hence able to reflect the richness of natural language and with a potentially complete domain coverage. The Wikipedia ontology is encoded in two semantic resources i.e., the Wikipedia entity type dataset and the Wikipedia class taxonomy dataset, which are described in Section 5.

5 Results and evaluation

In this section we report the results of our work, and evaluate them in terms of precision, recall and time of computation. Our final goal is to incrementally build a *Wikipedia ontology* based on how users describe things, hence able to capture the richness of NL definitions. To this aim, we have developed a web service, called *Tìpalo*, which, given a description of an entity, produces a RDF named graph describing its typing information by means of DBpedia entities, WordNet synsets and supersenses, and foundational ontology classes. Besides the RDF resources described in Section 3, and the methods in Section 4, we have developed a novel resource and two evaluation tools.

Wikipedia ontology. We have produced a demonstrating Wikipedia ontology[27] by analyzing a randomly selected sample of 800 Wikipedia pages. The resulting Wikipedia ontology consists of two RDF datasets, one containing `rdf:type` triples defining DBpedia entity types, and another containing ontology classes related by means of `rdfs:subClassOf` and `owl:equivalentTo` axioms. The two

[25] Any other aligned ontology, or ontology matching component can be used in order to perform arbitrary type matching.

[26] Dolce Zero, `http://www.ontologydesignpatterns.org/ont/d0.owl`.

[27] Link available at the Wikipedia ontology page: `http://stlab.istc.cnr.it/stlab/WikipediaOntology/`

datasets can be queried through a SPARQL endpoint (or downloaded as dump files) either as a whole graph or as separated named graphs, each associated with a single entity. Each named graph has an ID starting with `dbpedia_` followed by a DBpedia entity local name e.g., `dbpedia_Vladimir_Kramnik`.

Reusable evaluation tools. Additionally, we have developed two tools for evaluating our method, one for collaboratively building a golden standard, and the other for evaluating the Wikipedia ontology (both tools are described in Section 5.1).

5.1 Evaluation

We evaluate our work considering the accuracy of types assigned to the sample set of Wikipedia entities, and the soundness of the induced taxonomy of types for each DBpedia entity. The accuracy of types has been measured in two ways: (i) in terms of precision and recall against a golden standard of 100 entities, and (ii) by performing a user study. The soundness of the induced taxonomies has been assessed in a user study.

Building a sample set of Wikipedia pages. We have performed our experiments on a sample set of ∼800 randomly selected Wikipedia pages. From the 800 set, we have removed all pages without an abstract text, e.g. *redirect* pages, categories, and images. The resulting sample includes 627 pages with the following characteristics: (i) each page has a corresponding DBpedia entity, (ii) each DBpedia entity has either a DBpedia type, a YAGO type, or no type, (ii) 67.62% of the corresponding DBpedia entities have a YAGO type, 15.47% have a DBPO type, and 30% of them have no type.

Building a golden standard. We have built a manually annotated golden standard of Wikipedia entity types based on the sample set used for our experiments. To support this process we have developed a web-based tool named *WikipediaGold*[28] that manages argumentation among users in order to support them in discussing and reaching agreement on decisions (agreement was considered reached with at least 70% users giving the same answer). Ten users with expertise in ontology design (four senior researchers and six PhD students in the area of knowledge engineering) have participated in this task, and have reached agreement on 100 entities. We have used such 100 entities as a golden standard for evaluating and tuning our method. The golden standard can be retrieved from the cited Wikipedia Ontology page, and it can be useful for future development and for comparing our work with possible other approaches to this same task.

WikipediaGold is based on a simple user task, repeated iteratively: given an entity e e.g., `dbp:Vladimir_Ramnik`, WikipediaGold visualizes its definition e.g., *"Vladimir Borisovich Kramnik is a Russian chess grandmaster."* and asks users to:

[28] Available online at `http://wit.istc.cnr.it/WikipediaGold`, demonstrating video at `http://wit.istc.cnr.it/stlab-tools/video/`

Component	precision	recall	F-measure (F1)
Type selector	.93	.90	.92
WSD (UKB)	.86	.82	.84
WSD (most frequent sense)	.77	.73	.75
Type matcher (Supersense)	.73	.73	.73
Type matcher (DUL+/D0)	.80	.80	.80

Table 4: Performance evaluation of the individual pipeline components.

- indicate if e refers to a concept/type or to a specific instance. Users can select either "is a" or "is a type of" as possible answers. This value allows us to evaluate if our process is able to distinguish entities which are typical individuals, from those that are typical classes;
- copy and paste the terms in the definition that identifies the types of e, or indicate a custom one, if the definition does not contain any. In our example, a user could copy the term *"Russian chess grandmaster"*. This information is meant to allow us evaluating the performances of the *type selector*;
- select the most appropriate concepts for classifying e from two lists of terms. The first list includes 21 WordNet supersenses, and the second list includes 23 classes from DULplus and Dolce Zero. Each type is accompanied by a describing gloss and some examples to inform the user about its intended meaning. In the example, users can select the type "Person" available in both lists. The two lists of concepts are available online at the Wikipedia ontology page.

For each answer, users can optionally include a comment motivating their choice. When there is disagreement among users about an entity, WikipediaGold submits it again to users who have already analyzed it. In these cases a user can see other users' choices and comments, and decide if either to keep her decision, or to change it. In both cases, a comment motivating own decision must be entered.

Evaluation against the golden standard. Our evaluation is based on measuring precision and recall of the output of the three main steps of the process, against the golden standard: (i) type selection (step 3), (ii) word sense disambiguation (WSD) (step 4), and (iii) type matching (step 5). We also measure precision and recall of the overall process output.

Typing process	precision	recall	F-measure (F1)
WordNet types	.76	.74	.75
Supersenses	.62	.60	.61
Dul+/D0	.68	.66	.67

Table 5: Performance evaluation of the overall process.

The results shown in Table 4 indicate the performances of the individual components. The type selector stands out as the most reliable component ($F1 = .92$), which confirms our hypothesis that a rich formalization of definitions and a good design of graph patterns are a healthy approach to entity typing. The WSD task

has been performed with two approaches: we analyze its performance by executing UKB as well as a most-frequent-sense-based (MFS) approach. UKB shows to perform better ($F1 = .84$) than MFS ($F1 = .75$), suggesting that Wikipedia definitions often include polysemous senses, and that the used language tends to be specialized i.e., polysemous terms are used with different senses. The type matcher performs better with DULplus/Dolce Zero types than with WordNet supersenses, which shows an improvement with respect to the state of the art considering that WordNet super senses are considered an established and reliable semantic resource when used as a top-level ontology for WordNet.

Table 5 illustrates the performance of the overall automatic typing process. As expected, the steps that map the extracted types to WordNet types, super senses, and top-level ontologies tend to decrease the initial high precision and recall of the type selector. In fact, when put into a pipeline, errors typically reinforce previous ones, producing in this case an overall decrease of $F1$ from .92 of the type selection step to .75 of the combined type selection and WSD, to .67 with the addition of DULplus/Dolce Zero alignment (type matcher). However, the modularity of our process enables to reuse the results that are actually useful to a certain project, e.g. discarding a step that performs worse.

The good performances observed in our evaluation experiments make us claim that using Tìpalo brings advantages when compared to the most prominent existing approaches i.e., DBpedia project [11] and YAGO [17] to DBpedia entity typing, for the following reasons: (i) Tìpalo potentially ensures complete coverage of Wikipedia domains (intensional coverage) as it is able to capture the reachness of terminology in NL definitions and to reflect it in the resulting ontology, while DBpedia and YAGO depend both on the limited intensional completeness of infobox templates and Wikipedia categories, (ii) Tìpalo is independent from the availability of structured information such as infobox templates and Wikipedia categories, hence ensuring higher extensional completeness as most Wikipedia entities have a definition while many of them lack infoboxes.

A direct comparison of our results with DBpedia and YAGO approaches occurred to be unfeasible in the scope of this paper because the two approaches differ from ours on important aspects: they use different reference type systems; they rely on Wikipedia categories or infobox templates while we rely on the NL descriptions used for defining Wikipedia entities by the crowds, hence it is difficult (if not impossible) to compare the derived vocabularies; finally, the granularity of their type assignments is heterogeneous. These cases make it hard to define criteria for performing a comparison between the accuracy of the automatically assigned types. Hence, we could not consider either DBpedia or YAGO suitable golden standards for this specific task, which motivates the construction of a specific golden standard.

Evaluation by user study. In order to further verify our results, we have conducted a user study. We have implemented a second Web-based tool, named *WikipediaTypeChecker*[29], for supporting users in expressing their judgement on

[29] Available online at `http://wit.istc.cnr.it/WikipediaTypeChecker`, demonstrating video at `http://wit.istc.cnr.it/stlab-tools/video`

the accuracy of Tìpalo types assigned to the sample set of Wikipedia entities. WikipediaTypeChecker is available online.

WikipediaTypeChecker asks users to evaluate the accuracy of Tìpalo types, the soundness of the induced taxonomies, and the correctness of the selected meaning of types, by expressing a judgement on a three-valued scale: *yes, maybe, no*. Users' task, given an entity with its definition, consists of three evaluation steps. Consider for example the entity `dbp:Fast_chess`: in the first step, users evaluate the accuracy of the assigned types by indicating the level of correctness of proposed types. In this example, for the entity "Fast chess" three types are proposed: "Chess game", "Game", and "Activity"; in the second step users validate the soundness of the induced taxonomy of types for an entity. In this example, the proposed taxonomy is `wt:ChessGame rdfs:subClassOf wt:Game`; in the third step users evaluate the correctness of the meaning of individual types (i.e. WSD). For example, the proposed meaning for "Chess game" is "a board game for two players who move their 16 pieces according to specific rules; the object is to checkmate the opponent's king". Five users with expertise in knowledge engineering have participated in the user study (three PhD students and two senior researchers). For each entity and for each evaluation step, we have computed the average value of judgements normalized to an interval [0,1], which gives us a value for the precision of results. The results are shown in Table 6, with a (high) inter-rater agreement (Kendall's W) of .79[30].

Task	Type extraction	Taxonomy induction	WSD
Correctness	.84	.96	.81

Table 6: Results of the user-based evaluation, values are expressed in percentage and indicate precision of results. Inter-rater agreement (Kendall's W) is .79, Kendall's W ranges from 0 (no agreement) to 1 (complete agreement).

These results confirm those observed in the evaluation against a golden standard (cf. Tables 4 and 5). In this case, we have split the evaluation of the correctness of extracted types between *assigned types* (.84), and *induced taxonomy* (.96): their combination is comparable to the precision value observed for the type selector against the golden standard (.93). The performance of the WSD task is a bit lower (.81 against .86 precision), which suggests the need for additional evaluation of WSD performance, and exploration of possible alternative solutions.

Estimating time performance. In the scope of this work, we could only perform a preliminary estimation of time performances, since we have run the process on simple desktop machines. The workflow process and storage data have been executed on an Intel Pentium DualCore 2.80GHz with 1GB RAM, while UKB and FRED run on a Quad-Core Intel Xeon 2,26 GHz RAM 32 GB Processor Interconnect Speed: 5.86 GT/s. With this setting, the whole process takes maximum ∼11 seconds per definition (depending on its complexity). Type

[30] Kendall's W is a coefficient of concordance used for assessing agreement among raters. It ranges from 0 (no agreement) to 1 (complete agreement), and is particularly suited in this case as it makes no assumptions regarding the nature of the probability distribution and handles any number of distinct outcomes.

selection and top-level matching are instantaneous, while there is a bottleneck due to UKB performance: our processes become extremely fast (maximum ∼2 seconds per definition) if we remove UKB and disambiguate terms by selecting synsets with the most-frequent-sense-based method (with some degradation in precision and recall). We remark that in this implementation UKB and FRED are used as web services and remotely invoked (one machine is in Rome and another is in Bologna), hence suffering from delay due to network latency. We are confident that by parallelizing the whole process on a more robust cluster, and deploying all components and data sources locally, the speed will significantly increase[31], which reasonably suggest the applicability of the process on a large-scale dataset such as Wikipedia.

6 Conclusions and future work

We have presented Tìpalo, an implemented method that formalizes entity definitions extracted from Wikipedia for automatically typing DBpedia entities and linking them to other DBpedia resources, WordNet, and foundational ontologies. We have experimented Tìpalo on a sample set of ∼800 Wikipedia entities. Results have been evaluated against a golden standard and a user study, and are up to the task, specially for the pure type selection task. In ongoing work, we are deploying the tool on a more robust cluster for improving time performances and experimenting on a large-scale resource such as the whole Wikipedia. The medium-term goal is to incrementally build a Wikipedia ontology that reflects the richness of terminology expressed by natural language, crowd sourced definitions of entities.

References

1. E. Agirre and A. Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th conference of the European chapter of the Association for Computational Linguistics (EACL-2009)*, Athens, Greece, 2009. The Association for Computer Linguistics.
2. P. Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer, 2006.
3. P. Cimiano and J. Vlker. Text2onto - a framework for ontology learning and data-driven change discovery, 2005.
4. J. R. Curran, S. Clark, and J. Bos. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic, 2007.
5. O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open information extraction: The second generation. In *IJCAI*, pages 3–10. IJCAI/AAAI, 2011.
6. A. Gangemi. Norms and plans as unification criteria for social collectives. *Autonomous Agents and Multi-Agent Systems*, 17(1):70–112, 2008.
7. A. Gangemi, R. Navigli, and P. Velardi. The ontowordnet project: extension and axiomatization of conceptual relations in wordnet. In *in WordNet, Meersman*, pages 3–7. Springer, 2003.

[31] We have plans to perform additional tests in such an environment in the immediate future.

8. S. Hartmann, G. Szarvas, and I. Gurevych. Mining multiword terms from wikipedia. In M. T. Pazienza and A. Stellato, editors, *Semi-Automatic Ontology Development: Processes and Resources*, pages 226–258. IGI Global, Hershey, PA, USA, 2012.
9. M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.
10. A. Kalyanpur, J. W. Murdock, J. Fan, and C. A. Welty. Leveraging community-built knowledge for type coercion in question answering. In *International Semantic Web Conference (2)*, pages 144–156, 2011.
11. J. Lehmann, C. Bizer, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A Crystallization Point for the Web of Data. *Journal of Web Semantics*, 7(3):154–165, 2009.
12. R. Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2), 2009.
13. R. Navigli and S. P. Ponzetto. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* Uppsala, Sweden, 11–16 July 2010, pages 216–225, 2010.
14. A. G. Nuzzolese, A. Gangemi, V. Presutti, and P. Ciancarini. Encyclopedic Knowledge Patterns from Wikipedia Links. In L. Aroyo, N. Noy, and C. Welty, editors, *Proceedings fo the 10th International Semantic Web Conference (ISWC2011)*, pages 520–536. Springer, 2011.
15. A. G. Nuzzolese, A. Gangemi, V. Presutti, and P. Ciancarini. Type inference through the analysis of wikipedia links. In *WWW2012 Workshop on Linked Data on the Web (LDOW 2012)*. CEUR, 2012.
16. V. Presutti, F. Draicchio, and A. Gangemi. Knowledge extraction based on discourse representation theory and linguistic frames. In *EKAW: Knowledge Engineering and Knowledge Management that matters (to appear)*. Springer, 2012.
17. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.
18. H. Tanev and B. Magnini. Weakly supervised approaches for ontology population. In *Proceedings of the 2008 conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pages 129–143, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.
19. J. Völker and S. Rudolph. Lexico-logical acquisition of owl dl axioms – an integrated approach to ontology refinement, 2008.
20. R. Witte, N. Khamis, and J. Rilling. Flexible ontology population from text: The owlexporter. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, editors, *LREC*. European Language Resources Association, 2010.