

November 2015

Joint learning of ontology and semantic parser from text

Janez STARC ^{a,1}, Dunja MLADENIĆ ^a^a *Jožef Stefan International Postgraduate School, Slovenia*

Abstract. Semantic parsing methods are used for capturing and representing semantic meaning of text. Meaning representation capturing all the concepts in the text may not always be available or may not be sufficiently complete. Ontologies provide a structured and reasoning-capable way to model the content of a collection of texts. In this work, we present a novel approach to joint learning of ontology and semantic parser from text. The method is based on semi-automatic induction of a context-free grammar from semantically annotated text. The grammar parses the text into semantic trees. Both, the grammar and the semantic trees are used to learn the ontology on several levels – classes, instances, taxonomic and non-taxonomic relations. The approach was evaluated on the first sentences of Wikipedia pages describing people.

Keywords. ontology learning, semantic parsing, grammar induction, context-free grammar

1. Introduction

One of the ultimate goals of Natural Language Processing (NLP) is machine reading [1], the automatic, unsupervised understanding of text. One way of pursuing machine reading is by semantic parsing, which transforms text into its meaning representation. However, capturing the meaning is not the final goal, the meaning representation needs to be predefined and structured in a way that supports reasoning. Ontologies provide a common vocabulary for meaning representations and support reasoning, which is vital for understanding the text. To enable flexibility when encountering new concepts and relations in text, in machine reading we want to be able to learn and extend the ontology while reading. Traditional methods for ontology learning [2,3] are only concerned with discovering the salient concepts from text. Thus, they work in a macro-reading fashion [4], where the goal is to extract facts from a large collection of texts, but not necessarily all of them, as opposed to a micro-reading fashion, where the goal is to extract every fact from the input text. Semantic parsers operate in a micro-reading fashion. Consequently, the ontologies with only the salient concepts are not enough for semantic parsing. Furthermore, the traditional methods learn an ontology for a particular domain, where the text is used just as a tool. On the other hand, ontologies are used just as tool to represent meaning in the semantic parsing setting. When developing a semantic parser it is

¹Corresponding Author: Janez Starc, Jožef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia; E-mail: janez.starc@ijs.si

not trivial to get the best meaning representation for the observed text, especially if the content is not known yet. Semantic parsing datasets have been created by either selecting texts that can be expressed with a given meaning representation, like Free917 dataset [5], or by manually deriving the meaning representation given the text, like Atis dataset [6]. In both datasets, each unit of text has its corresponding meaning representation. While Free917 uses Freebase [7], which is a very big multi-domain ontology, it is not possible to represent an arbitrary sentence with Freebase or any other existing ontology.

In this paper, we propose a novel approach to joint learning of ontology and semantic parsing, which is designed for homogeneous collections of text, where each fact is usually stated only once, therefore we cannot rely on data redundancy. Our approach is text-driven, semi-automatic and based on grammar induction. It is presented in Figure 1. The input is a seed ontology together with text annotated with concepts from the seed ontology. The result of the process is an ontology with extended instances, classes, taxonomic and non-taxonomic relations, and a semantic parser, which transform basic units of text, i.e sentences, into *semantic trees*. Compared to trees that structure sentences based on syntactic information, nodes of semantic trees contain semantic classes, like location, profession, color, etc. Our approach does not rely on any syntactic analysis of text, like part-of-speech tagging or dependency parsing. The grammar induction method works on the premise of curriculum learning [8], where the parser first learns to parse simple sentences, then proceeds to learn more complex ones. The induction method is iterative, semi-automatic and based on frequent patterns. A context-free grammar (CFG) is induced from the text, which is represented by several layers of semantic annotations. The motivation to use CFG is that it is very suitable for the proposed alternating usage of top-down and bottom-up parsing, where new rules are induced from previously unparseable parts. Furthermore, it has been shown by [9] that CFGs are expressive enough to model almost every language phenomena. The induction is based on a greedy iterative procedure that involves minor human involvement, which is needed for seed rule definition and rule categorization. Our experiments show that although the grammar is ambiguous, it is scalable enough to parse a large dataset of sentences.

The grammar and semantic trees serve as an input for the new ontology. Classes, instances and taxonomic relations are constructed from the grammar. We also propose a method for discovering less frequent instances and their classes, and a supervised method to learn relations between instances. Both methods work on semantic trees.

For experimentation, first sentences of Wikipedia pages describing people are taken as a dataset. These sentences are already annotated with links to other pages, which are also instances of DBpedia knowledge base [10]. Using relations from DBpedia as a training set, several models to predict relations have been trained and evaluated.

The rest of the paper is organized in the following way. The grammar induction approach is presented in Section 2. The ontology induction approach follows in Section 3. In Section 4 we present the conducted experiments with grammar induction, and instance and relation extraction. We examine the related work in Section 5, and conclude with the discussion in Section 6.

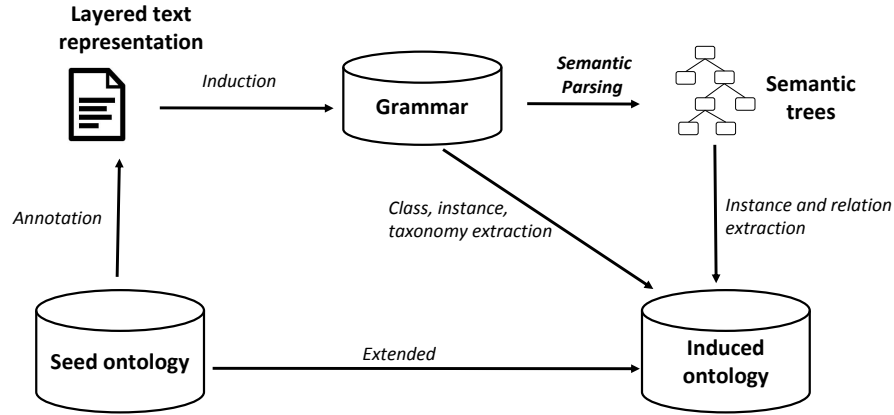


Figure 1. The proposed approach to ontology and grammar induction gets a collection of text and the seed ontology as input and outputs a new ontology. The collection of text is first annotated at different levels including annotations with the concepts from existing ontology. The annotated text is then used for grammar induction, where the text is represented as semantic trees. These are used together with the grammar to induce a new ontology.

2. Grammar induction

In this section, we propose a semi-automatic bootstrapping procedure for grammar induction, which searches for the most frequent patterns and constructs new production rules from them. One of the main challenges is to make the induction in a way that minimizes human involvement and maximizes the quality of semantic trees.

The input to the process, which is illustrated in Figure 2, is a set of predefined seed grammar rules (see Section 2.5) and a sample of sentences in a layered representation (see Section 2.1) from the dataset. The output of the process is a larger set of rules forming the induced grammar. One rule is added to the grammar on each iteration. At the beginning of each iteration all the sentences are parsed with a top-down parser. The output of parsing a single sentence is a semantic tree – a set of semantic nodes connected into a tree. Here we distinguish two possible outcomes of the parsing: 1) the sentence was completely parsed, which is the final goal and 2) there is at least one part of the sentence that cannot be parsed. From the perspective of a parser the second scenario happens when there is a node that cannot be parsed by any of the rules. We name these nodes – *null nodes* – and they serve as the input for the next step, the rule induction. In this step several rules are constructed by generalization of null nodes. The generalization (see Section 2.4) is based on utilization of semantic annotations and bottom-up composition of the existing rules. Out of the induced rules, a rule with the highest frequency (the one that was generalized from the highest number of null nodes) is added to the grammar. To improve quality of the grammar, the rules are marked by so called property, which instructs the parser how to use the rule (eg., use it in parsing but not in induction). The property vitally affects result of the parsing in the following iterations potentially causing a huge semantic drift for the rest of process. Consequently, a human user needs to mark

the property of each rule. The iterative process runs until a predefined stopping criteria is met. The criteria is either connected to the quality of the grammar or time limitation.

For the sake of transparency of the experiments, the human is involved in the beginning, when the seed rules are created and later when the rule properties are specified. However, in another setting the user could also define new rules in the middle of the bootstrapping procedure.

In the following sections, we describe each component of the process in more details.

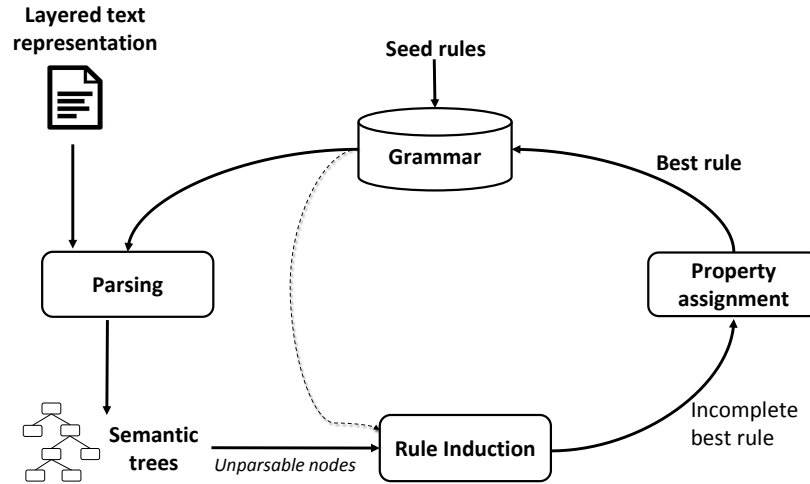


Figure 2. Grammar induction

2.1. Textual data representation

The input textual data needs to be properly structured in order to work best with the proposed algorithms. Shallow NLP tools, like sentence splitting, word tokenization, named entity recognition, might help obtaining this structure. The basic unit is a sentence, represented by several layers. An example is presented in Table 1. Each layer consists of several tokens, which span over one or more words. The basic layer is the lexical layer, where each token represents a single word. All other layers are created from the annotations. Some annotations, like named-entities, may span over several words; some of the words may not have an annotation, thus they are given a *null token*. It is crucial that all algorithms are aware how to deal with a particular layer. For instance, the parser must not break apart a multi-word annotation. Some layers may be derived from others using the seed ontology. For example, *instance* layer contains annotations to instances of the ontology and the derived *class* layer represents the classes of these annotations, which are also from the ontology. Annotation layers are valuable if they provide good means for generalization or connection with the ontology. A *term* is a subpart of the sentence, defined by the starting and ending position in the sentence. It has different interpretation

Layer	Tokens						
lexical	Phil	Madeira	is	a	musician	from	Nashville
small-caps	phil	madeira	is	a	musician	from	nashville
named-entity	Person		-	-	-	-	Location
instance	Phil_Madeira		-	-	Musical_ Artist	-	Nashville_ Tennessee
class	Person		-	-	Profession	-	Location

Table 1. Layered representation of a sentence. Null tokens are expressed as "-".

in each layer. If the interpretation breaks any of the tokens, it is not valid. For instance, term representing *Madeira* is not valid in *named-entity* layer in Table 1 because it breaks *Person*.

2.2. Grammar Definition

Our context-free grammar G is defined by the 5-tuple: $G = (V, \sigma, P, S, R)$, where

- V is a set of *non-terminals*. Each non-terminal represents a semantic class, e.g. $\langle \text{Person} \rangle$, $\langle \text{Color} \rangle$, $\langle \text{Organization} \rangle$. There is also a universal non-terminal $\langle * \rangle$, which can be replaced by any other non-terminal. The same non-terminal replaces all occurrences in a rule. It is used to represent several rules, with a notation. The grammar is still context-free. See *seed rule examples* in Section 2.5.
- σ is a set of *terminals*. Terminal is any existing non-null token from any sentence layer. We denote a terminal by $value\{layer\}$. For instance, $[location]\{named-entity\}$, $Phil_Madeira\{instance\}$. If the terminal is from the lexical layer, the layer is skipped in the denotation.
- P is a set of production rules that represents a relation from $V \rightarrow (V \cup E)^*$. For example,

$$\langle \text{Relation} \rangle ::= \langle \text{Person} \rangle \text{ is } \langle \text{Life Role} \rangle$$

- S is the starting non-terminal symbol. Since non-terminals represent semantic classes, the starting symbol is chosen based on the semantic class of the input examples. If the input examples are sentences, then the appropriate category may be $\langle \text{Relation} \rangle$. While if the input examples are noun phrases, the starting symbol may be a more specific category, like $\langle \text{Job Title} \rangle$.
- R is a set of properties: *positive*, *neutral*, *negative*, *non-inducible*. The property controls the usage of the rule in the parsing and in the rule induction phase. More details are given in the following subsections.

2.3. Parser

For parsing, a recursive descent parser with backtracking was developed. This is a top-down parser, which first looks at the higher level sentence structure and then proceeds down the parse tree to identify low level details of the sentence. The advantage of top-down parsing is the ability to partially parse sentences and to detect unparsable parts of sentences.

The parser takes a layered sentence as an input and returns a semantic tree as an output (see Figure 3). The recursive structure of the program closely follows the structure

of the parse tree. The recursive function *Parse* (see Algorithm 1) takes a term and a non-terminal as input and returns a parse node as an output. The parse node contains the class of node (non-terminal), the *rule* that parsed the node, the *term*, and the list of children nodes. In order for the rule to parse the node, the left-hand side must match the input non-terminal and the right-hand side must match the layered input. In the pattern matching function *Match* (line 6), the right hand side of a rule is treated like a regular expression; non-terminals present the (+) wildcard characters, which match at least one word. The terminals are treated as literal characters, which are matched against the layer that defines them. The result of successfully matched pattern is a list of terms, where each term represents a non-terminal of the pattern. Due to ambiguity of pattern matching there might be several matches. For each of the term – non-terminal pair in every list the *parse* function is recursively called (line 10).

```

1  Parse(Phrase p, Non-terminal n)  output: parse node
2  nodes ← {};
3  foreach rule r of grammar do
4      if n = left side of r then
5          pattern ← right hand side of r;
6          ambiguous lists ← Match(pattern, p) ;
7          foreach term list of ambiguous lists do
8              child nodes ← {};
9              for i ← 0 to size of term list do
10                 child node ← Parse(term listi, pattern.non terminalsi) ;
11                 add child node to child nodes;
12                 add Node(type, p, r, child nodes) to nodes;
13 if nodes is empty then
14     final node ← Node(type, p, null, {});
15 else
16     final node ← argmaxn ∈ nodes r(n) ;
17 if final node is not fully parsed then
18     add final node to induction nodes ;
19 return final node

```

Algorithm 1: Pseudocode of the main function *parse* of the top-down parser.

Since the grammar is ambiguous, a term can be parsed in multiple ways. There are two types of ambiguity. Two or more rules can expand the same term and one rule can expand the term in more than one way. For each ambiguity one node is created, and the best node according to *reliability measure* is selected to be the result (line 16). The reliability measure $r(n)$ is

$$r(n) = \begin{cases} 1, & \text{if node is fully parsed} \\ \beta \cdot (1 - tp(n)) + (1 - \beta) \frac{\sum_{c \in C(n)} |c| \cdot r(c)}{\sum_{c \in C(n)} |c|}, & \text{if node is partially parsed} \\ 0, & \text{if node is null} \end{cases} \quad (1)$$

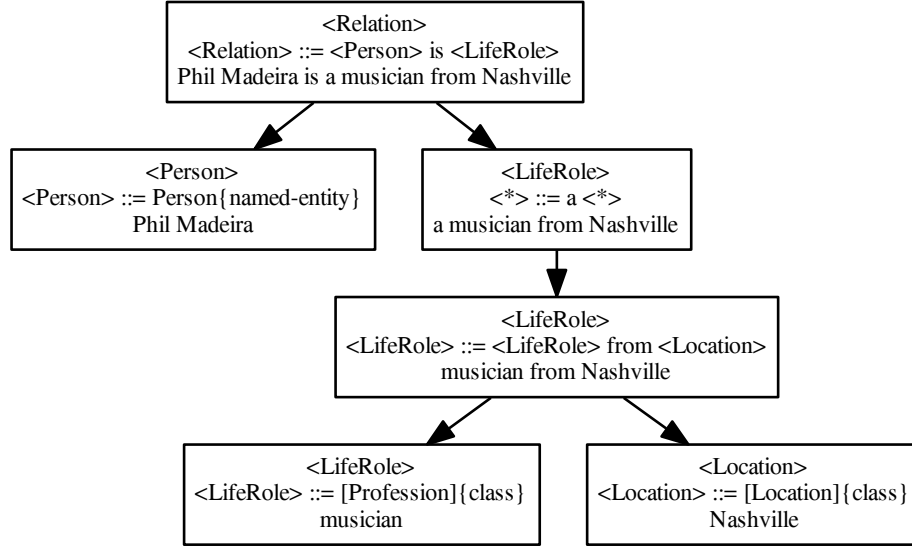


Figure 3. An example of a semantic tree, which is the result of parsing the input from Table 1. The tree is fully parsed. Each node has three rows: the class, the rule and the term interpreted in the lexical layer.

where $tp(n)$ is the trigger probability of the rule that parsed the node n , β is a predefined weight, $C(n)$ is the set of children of n , and $|c|$ is the length of the term of node c . The trigger probability of the rule is the probability that a the right-hand side of the rule pattern matches a random term in the dataset and it is estimated after the rule is induced. The range of the measure is between 0 and 1. The measure was defined in such a way that the more text the node parses, the higher is the reliability (the second summand in the middle row of Eq. 1). On the other hand, nodes with rules that are more frequently matched have lower reliability; this penalizes rules that are very loosely defined (the first summand in the middle row of Eq. 1). The β parameter was set to 0.05, using grid search, with average F1 score from relation extraction experiment from Section 4.4 as a performance measure.

If none of the rules match the term, a *null node* is created and added to the list of nodes, which will be later used for grammar induction (line 18). Note that even if a null node is discarded, because it is not the most reliable, it will still be used in the grammar induction step. A node is *fully parsed* if the node itself and all of its descendants are parsed. If a node is parsed and if at least one of its descendants is not parsed, then the node is *partially parsed*. All nodes that are not fully parsed are added to the list for induction.

Since the ambiguity of the grammar may make parsing computationally infeasible, several optimization techniques are used. Memoization [11] is used to reduce the complexity from exponential time to $\mathcal{O}(n^3)$ [12], where n is the length of the sentence. The parser does not support ϵ productions mainly because the grammar induction will not produce them. The patterns that do not contain terminals are the most ambiguous. At most two non-terminals are allowed, and the maximal length of the term that corresponds to the first non-terminal is three tokens. We argue that this is not a huge limitation, since

November 2015

the way human languages are structured, usually two longer terms are connected with a word, like comma or a verb. Furthermore, the way how our induction works, these connectors do not get generalized and become a terminal in the rule. There was an attempt to introduce rules with negative property. Whenever such rule fully parses a node, that indicates that the current parsing path is incorrect. This allows the parser to back-track sooner and also prevents adding null sister nodes (null sister nodes are in this case usually wrong) to the rule induction. However, it turned out that negative rules actually slow down the parsing, since the grammar gets bigger. It is better to mark these rules as neutral, therefore they are not added to the grammar.

2.4. Rule induction

The goal of the rule induction step is to convert the null nodes from the parsing step into rules. Out of these rules, the *most frequent* one is promoted. The term from the null node is *generalized* to form the right side of the rule. The class non-terminal of the null node will present the left side of the rule. Recently induced rule will parse all the nodes, from which it was induced, in the following iterations. Additionally, some rules may parse the children of those nodes.

Generalization

Generalization is done in two steps. First, terms are generalized on the layer level. The output of this process is a sequence of tokens, which might be from different layers. For each position in the term a single layer is selected, according to predefined layer order. In the beginning, term is generalized with the first layer. All the non-null tokens from this layer are taken to be part of the generalized term. All the positions of the term that have not been generalized are attempted to be generalized with the next layer, etc. The last layer is without null-tokens, therefore each position of the term is assigned a layer. Usually, this is the lexical layer. For example, top part of Table 2 shows generalization of term from Table 1. The layer list is constructed manually. Good layers for generalization are typically those that express semantic classes of individual terms. Preferably, these types are not too general (loss of information) and not too specific (larger grammar).

In the next step of generalization, tokens are further generalized using a greedy bottom-up parser using the rules from the grammar. The right sides of all the rules are matched against the input token term. If there is a match, the matched sub-term is *replaced* with the left side of the rule. Actually, in each iteration all the disjunct matches are replaced. To get only the disjunct matches, overlapping matches are discarded greedily, where longer matches have the priority. This process is repeated until no more rules match the term. An example is presented in the lower part of Table 2.

The bottom-up parsing algorithm needs to be fast because the number of unexpanded nodes can be very high due to ambiguities in the top-down parsing. Consequently, the algorithm is greedy, instead of exhaustive, and yields only one result. Aho-Corasick string matching algorithm [13] is selected for matching for its ability to match all the rules simultaneously. Like the top-down parser, this parser generates partial parses because the bottom-up parser will never fully parse – the output is the same as the non-terminal type in the unexpanded node. This would generate a cyclical rule, i.e. $\langle Class \rangle ::= \langle Class \rangle$. However, this never happens because the top-down parser would already expand the null node.

November 2015

Layered generalization					
Person	is	a	Profession	from	Location
class	small-caps	small-caps	class	small-caps	class

Bottom-up parsing			
Person	is	a	<Life Role>
class	small-caps	small-caps	non-terminal

Table 2. Two step generalization of term from Table 1 (Phil Madiera is a musician from Nashville). The layer list was constructed in a reverse order from Table 1. In this example, the rule *<Life Role> ::= Profession from Location* is the only rule that was used in the bottom-up parsing.

Property assignment

The last step of the iteration is assigning the property to the newly induced rule. Property controls the role of the rule in the parsing and induction. The default property is *positive*, which defines the default behavior of the rule in all procedures. Rules with *neutral* property are not used in any procedure. They also cannot be re-induced. Some rules are good for parsing, but may introduce errors in the induction. These rules should be given *non-inducible* property. For instance, rule *<Date> ::= <Number>* is a candidate for the non-inducible property, since years are represented by a single number. On the contrary, not every number is a date.

In our experiments, the assignment was done manually. The human user sees the induced rule and few examples of the null nodes, from which it was induced. This should provide enough information for the user to decide in a few seconds, which property to assign. After the stopping criteria is met, the iterative procedure can continue automatically by assigning positive property to each rule. Initial experimenting showed that just a single mistake in the assignment can cause a huge drift, making all further rules wrong.

2.5. Seed rules

Before the start, a list of seed rules may be needed in order for grammar induction to be successful. Since this step is done manually, it is reasonable to have a list of seed rules short and efficient. Seed rules can be divided in three groups: domain independent linguistic rules, class rules, top-level domain rules. *Domain independent linguistic rules*, such as

<_> ::= a <_> <Relation> ::= <Relation> .
 <_> ::= an <_> <_> ::= <_> and <_>
 <_> ::= the <_> <_> ::= <_> , <_> and <_>,

parse the top and mid-level nodes. They can be applied on many different datasets. *Class rules* connect class tokens, like named-entity tokens with non-terminals. For example,

<Location> ::= [location]{named-entity} <Date> ::= [date]{named-entity}
 <Location> ::= [Location]{class} <Film> ::= [Film]{class}

They parse the leaf nodes of the trees. On the other hand, *top-level domain rules*, define the basic structure of the sentence. For example,

<Relation> ::= <Person> is <Life Role>

November 2015

As the name suggests, they parse nodes close to the root. Altogether, these rule groups parse on all levels of the tree, and may already be enough to parse the most basic sentences, but more importantly, they provide the basis for learning to parse more complex sentences.

The decision on which and how many seed rules should be defined relies on human judgment whether the current set of seed rules is powerful enough to ignite the bootstrapping procedure. This judgment may be supported by running one iteration and inspecting the top induced rules.

3. Ontology induction

This section describes how to utilize the grammar and manipulate semantic trees to discover ontology components in the textual data.

3.1. Ontology induction from grammar

We propose a procedure for mapping grammar components to ontology components. In particular, classes, instances and taxonomic relations are extracted.

First, we distinguish between instances and classes in the grammar. Classes are represented by all non-terminals and terminals that come from a layer populated with classes, for example, *named-entity* layer and *class* layer from Table 1. Instances might already exist in the *instance* layer, or they are created from rules, whose right hand side contains only tokens from the *lexical* layer. These tokens represent the label of the new instance. For instance rule $\langle \textit{Profession} \rangle ::= \textit{software engineer}$ is a candidate for instance extraction.

Furthermore, we distinguish between class and instance rules. *Class rules* have a single symbol representing a class on the right-hand side. Class rules map to *subClassOf* relations in the ontology. If the rule is positive, then the class on the right side is the subclass of the class on the left side. For instance, rule $\langle \textit{Organization} \rangle ::= \langle \textit{Company} \rangle$ yields relation (*subClassOf Company Organization*).

On the other hand, *instance rules* have one or more symbols representing an instance on the right side, and define the *isa* relation. If the rule is positive, then the instance on the right side is a member of a class on the left side. For instance, rule $\langle \textit{Profession} \rangle ::= \textit{software engineer}$ yields relation (*isa SoftwareEngineer Profession*). If class or instance rule is neutral then the relation can be treated as false. Note that many other relations may be inferred by combining newly induced relations and relations from the seed ontology. For instance, induced relation (*subClassOf new-class seed-class*) and seed relation (*isa seed-class seed-instance*) are used to infer a new relation (*isa new-class seed-instance*).

In this section, we described how to discover relations on the taxonomic level. In the next section, we describe how to discover relations between instances.

3.2. Relation extraction from semantic trees

We propose a method for learning relations from semantic trees, which tries to solve the same problem as the classical relation extraction methods. Given a dataset of positive relation examples that represent one relation type, e.g. *birthPlace*, the goal is to discover new unseen relations.

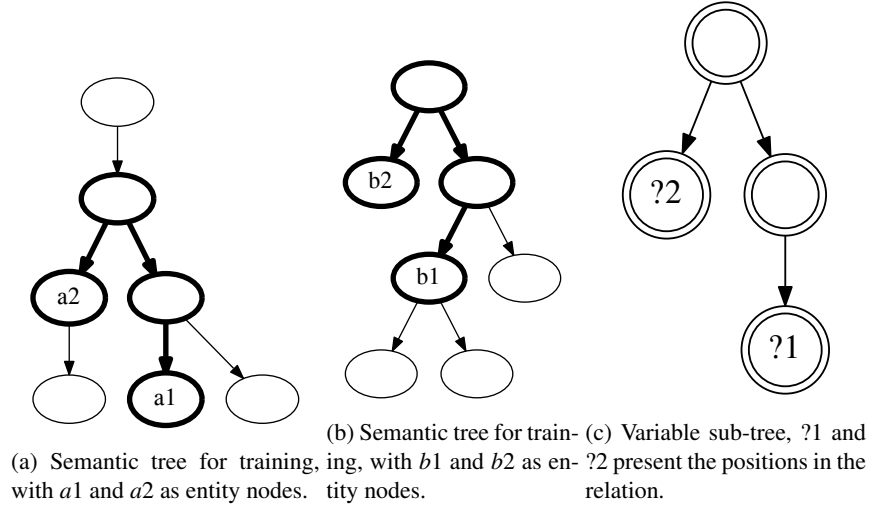


Figure 4. Given the relation $(r\ a1\ a2)$ and semantic tree on 4a, and relation $(r\ b1\ b2)$ and semantic tree on 4b, the same variable sub-tree is extracted (4c).

The method is based on the assumption that a relation between entities is expressed in the shortest path between them in the semantic tree [14]. The input for training are sentences in layered representation, corresponding parse trees, and relation examples. Given a relation from the training set, we first try to identify the sentence containing each entity of the relation. The relation can have one, two, or even more entities. Each entity is matched to the layer that corresponds to the entity type. For example, strings are matched to the lexical layer; ontology entities are matched to the layer containing such entities. The result of a successfully matched entity is a sub-term of the sentence. In the next step, the corresponding semantic tree is searched for a node that contains the sub-term.

At this point, each entity has a corresponding *entity node*. Otherwise, the relation is discarded from the learning process. Given the entity nodes, a minimum spanning tree containing all of them is extracted. If there is only one entity node, then the resulting subtree is the path between this node and the root node. The extracted sub-tree is converted to a *variable tree*, so that different semantic trees can have the same variable sub-trees, for example see Figure 4. The semantic nodes of the sub-tree are converted into variable nodes, by retaining the class and the rule of the node, as well as the places of the children in the original tree. For entity nodes also the position in the relation is memorized. Variable tree extracted from a relation is a positive example in the training process. For negative examples all other sub-trees that do not present any relations are converted to variable trees. Each variable node represents one feature. Therefore, a classification algorithm, such as logistic regression can be used for training.

When predicting, all possible sub-trees of the semantic tree are predicted². If a sub-tree is predicted as positive, then the terms in the leaf nodes represent the arguments of the relation.

²The number of leaf nodes of these sub-trees must match the number of arguments of the relation. Also, if the relation has two or more arguments, it is predicted several times, each time with the different position numbers in the entity nodes.

4. Experiments

In this section, we present experiments evaluating the proposed approach. We have conducted experimentation on Wikipedia–DBpedia dataset (Section 4.1). First, we have induced a grammar on the Wikipedia dataset (Section 4.2) to present its characteristics, and the scalability of the approach. In the next experiment, we present a method for discovering less prominent instances (Section 4.3). The last experiment demonstrates one application of semantic parsing – the supervised learning of DBpedia relations (Section 4.4).

4.1. Datasets

The datasets for experiments were constructed from English Wikipedia and knowledge bases DBpedia [10] and Freebase [7]. DBpedia provides structured information about Wikipedia articles that was scraped out of their infoboxes. First sentences of Wikipedia pages describing people were taken as the *textual dataset*, while DBpedia relations expressing facts about the same people were taken as the *dataset for supervised relation learning*. Note that each DBpedia instance has a Wikipedia page. A set of person instances was identified by querying DBpedia for instances that have a person class. For the textual dataset, Wikipedia pages representing these entities were parsed by the in-house Wikipedia markup parser³ to convert the markup into plain text. Furthermore, the links to other Wikipedia pages were retained. Here is an example of a sentence in plain text:

“Victor Francis Hess (24 June 1883 – 17 December 1964) was an Austrian-American physicist, and Nobel laureate in physics, who discovered cosmic rays.”

Using the Stanford OpenNLP [15] on plain texts we obtained sentence and token splits, and named-entity annotation. Notice, that only the first sentence of each page was retained and converted to the proposed layered representation (see Section 2.1). The layered representation contains five layers: *lexical* (plain text), *named-entity* (named entity recognizer), *wiki-link* (Wikipedia page in link – DBpedia instance⁴), *dbpedia-class* (class of Wikipedia page in Dbpedia) and *freebase-class* (class of Wikipedia page in Freebase). Freebase also contains its own classes of Wikipedia pages. For the last two layers, there might be several classes per Wikipedia page. Only one was selected using a short priority list of classes. If none of the categories is on the list, then the category is chosen at random. After comparing the *dbpedia-class* and *freebase-class* layers, only *freebase-class* was utilized in the experiments because more *wiki-link* tokens has a class in *freebase-class* layer than in *dbpedia-class* layer.

There are almost 1.1 million sentences in the collection. The average length of a sentence is 18.3 words, while the median length is 13.8 words. There are 2.3 links per sentence.

The dataset for supervised relation learning contains all relations where a person instance appears as a subject in DBpedia relation. For example,

dbpedia:Victor_Francis_Hess dbpedia-owl:birthDate 1883-06-24

³The markup parsing of Wikipedia markup is non-trivial. To extract only the plain text many elements, like images and tables, have been discarded, as well as some elements that appear in the middle of sentence, like pronunciation and citation

⁴Each Wikipedia page has a corresponding Dbpedia instance.

There are 119 different relation types (unique predicates), having from just a few relations to a few million relations. Since DBpedia and Freebase are available in RDF format, we used the RDF store for querying and for storage of existing and new relations.

4.2. Grammar Induction Experiments

The grammar was induced on 10.000 random sentences taken from the dataset described in Section 4.1. First, a list of 45 seed nodes was constructed. There were 22 domain independent linguistic rules, 17 category rules and 6 top-level rules. The property assignment was done by the authors. In every iteration, the best rule is shown together with the number of nodes it was induced from, and ten of those nodes together with the sentences they appear in. The goal was set to stop the iterative process after two hours. We believe this is the right amount of time to still expect quality feedback from a human user.

There were 689 new rules created. A sample of them is presented in Table 3. Table 4 presents the distributions of properties. Around 36% of rules were used for parsing (non neutral rules). Together with the seed rules there are 297 rules used for parsing. Different properties are very evenly dispersed across the iterations. Using the procedure for conversion of grammar rules into taxonomy presented in Section 3, **33** classes and subClassOf relations, and **95** instances and isa relations were generated.

The grammar was also tested by parsing a sample of 100.000 test sentences. A few statistic are presented in Table 4. More than a quarter of sentences were fully parsed, meaning that they do not have any null leaf nodes. Coverage represents the fraction of words in a sentence that were parsed (words that are not in null-nodes). The number of operations shows how many times was the *Parse* function called during the parsing of a sentences. It is highly correlated with the time spend for parsing a sentence, which is on average 0.16ms. This measurement was done on a single CPU core. Consequently, it is feasible to parse a collection of a million sentences, like our dataset. The same statistics were also calculated on the training set, the numbers are very similar to the test set. The fully parsed % and coverage are even slightly lower than on the test set. Some of the statistics were calculated after each iteration, but only when a non neutral rule was created. The graphs in Figure 5 show how have the statistics changed over the course of the grammar induction. Graph 5a shows that coverage and the fraction of fully parsed sentences are correlated and they grow very rapidly at the beginning, then the growth starts to slow down, which indicates that there is a long tail of unparsed nodes/sentences. In the following section, we present a concept learning method, which deals with the long tail. Furthermore, the number of operations per sentence also slows down (see Graph 5c) with the number of rules, which gives a positive sign of retaining computational feasibility with the growth of the grammar. Graph 5b somewhat elaborates the dynamics of the grammar induction. In the earlier phase of induction many rules that define the upper structure of the tree are induced. These rules can rapidly increase the depth and number of null nodes, like *rule 1* and *rule 2*⁵. They also explain the spikes on Graph 5d. Their addition to the grammar causes some rules to emerge on the top of the list with a significantly higher frequency. After these rules are induced the frequency gets back to the previous values and slowly decreases over the long run.

⁵rule1 – $\langle LifeRole \rangle ::= \langle LifeRole \rangle \text{ in } \langle Location \rangle$,
rule2 – $\langle LifeRole \rangle ::= \langle LifeRole \rangle \text{ of } \langle Organization \rangle$

number	rule	property
1	<PersonAttr> ::= born <Date>	none
101	<LifeRole> ::= born in <Location>	none
201	<Location> ::= <Location> from <Date>	neutral
301	<Person> ::= <Location> from <Date>	neutral
401	<OrderOfChivalry> ::= (<PersonAttr>)	neutral
501	<LifeRole> ::= who <Action> in <Event>	none
601	<Date> ::= the university	neutral

Table 3. A sample of induced rules.

Grammar	
positive rules	231
non-inducible rules	21
neutral rules	437
Parsing	
fully parsed sentences	25.63%
avg. coverage	78.52%
avg. tree depth	6.96
avg. number of leaf nodes	6.69
avg. number of null leaf nodes	1.98
avg. number of operations	320.3
avg parsing time	0.16 ms

Table 4. Statistics of test set.

4.3. Instance extraction

In this section, we present an experiment with a method for discovering new instances, which appear in the long tail of null nodes. Note that the majority of the instances were already placed in the ontology by the method in Section 3.1. Here, less prominent instances are extracted to increase the coverage of semantic parsing. The term and the class of the null node will form an *isa* relation. The class of the node represents the class of the relation. The terms are converted to instances. They are first generalized on the layer level (see Section 2.1). The goal is to exclude non-atomic terms, which do not represent instances. Therefore, only terms consisting of one *wiki-link* token or exclusively of lexical tokens are retained. The relations were sorted according to their frequency. We observe that accuracy of the relations drops with the frequency. Therefore, relations that occurred less than three times were excluded. The number and accuracy for six classes is reported in Table 5. Other classes were less accurate. For each class, the accuracy was manually evaluated on a random sample of 100 instance relations. Taking into account the estimated accuracy, there were more than 13.000 correct *isa* relations.

4.4. Relation extraction

In this section, we present an experiment of the relation extraction methods presented in Section 3.2. The input for the supervision is the DBpedia relation dataset from Section 4.1. The subject (first argument) of every relation is a person DBpedia instance –

November 2015

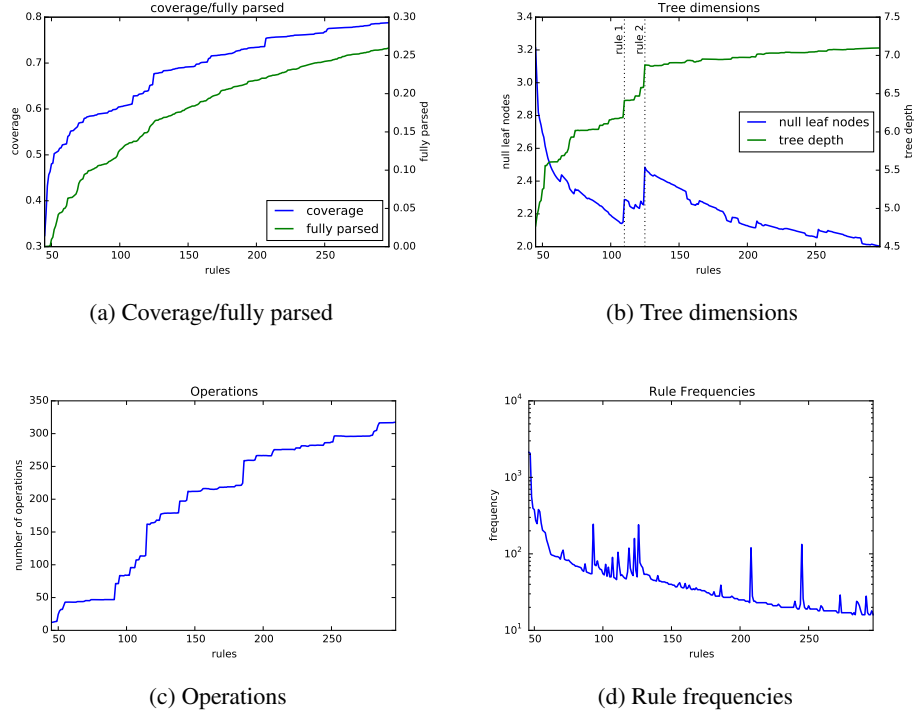


Figure 5. Statistics of the iterative process

Class	Relations	Accuracy
Life role	15356	60
Person	5427	49
Order of chivalry	1319	32
Date	1310	46
Action	967	78
Field of study	25	80

Table 5. Instance relations per category.

person Wikipedia page. In the beginning, the first sentence of that Wikipedia page has been identified in the textual dataset. If the object (last argument) of this relation matches a sub-term of this sentence, then the relation is eligible for experiments. We distinguish three types of values in objects. DBpedia resources are matched with *wiki-link* layer. Dates get converted to the format that is used in English Wikipedia. They are matched against the *lexical* layer, and so are the string objects.

Only relation types that have 200 or more eligible relations have been retained. This is 74 out of 119 relations. The macro average number of eligible relations per relation type is 17.7%. While the micro average is 23.8%, meaning that roughly a quarter of all DBpedia person relations are expressed in the first sentence of their Wikipedia page. For

Model	Precision	Converted Recall	Recall	Converted F1	F1
Basic	72.2	48.3	28.2	52.1	36.4
Net	61.6	77.6	42.5	63.3	46.9
LR	71.9	84.2	50.2	77.0	55.5
LRC	72.4	84.4	50.3	77.3	55.7
LRCL	76.2	85.5	50.7	80.0	57.1

Table 6. Performance of various relation extraction models.

the rest of this section, all stated averages are micro-averages.

The prediction problem is designed in the following way. Given the predicate (relation type) and the first argument of the relation (person), the model predicts the second argument of the relation (object). Because not all relations are functional, like for instance *child* relation, there can be several values per predicate–person pair; on average there are 1.1. Since only one argument of the relation is predicted, the variable trees presented in Section 3.2, will be paths from the root to a single node. Analysis of variable tree extraction shows that on average 60.8% of eligible relations were successfully converted to variable trees (the object term exactly matches the term in the node). Others were not converted because 8.2% of the terms were split between nodes and 30.9% terms are sub-terms in nodes instead of complete terms. Measuring the diversity of variable trees shows that a distinct variable tree appeared 2.7 times on average.

Several models based on variable trees were trained for solving this classification problem:

- *Basic (Basic model)* – The model contains positive trained variable trees. In the prediction, if the test variable tree matches one of the trees in the model, then the example is predicted positive.
- *Net (Automaton model)* – All positive variable trees are paths with start and end points. In this model they are merged into a net, which acts as a deterministic automaton. If the automaton accepts the test variable tree, then it is predicted positive. An example of automaton model is presented in Figure 6.
- *LR (Logistic regression)* – A logistic regression model is trained with positive and negative examples, where nodes in variable trees represents features.
- *LRC (Logistic regression + Context nodes)* – All leaf nodes that are siblings of any of the nodes in the variable tree are added to the *LR* model.
- *LRCL (Logistic regression + Context nodes + Lexical Tokens)* – Tokens from the lexical layer of the entity nodes are added to the *LRC* as features.

For training all or a maximum of 10,000 eligible relations was taken for each of 74 relation types. A 10-fold cross validation was performed for evaluation. The results are presented in Table 6. The converted recall and converted F1 score presents recall and F1 on converted examples, which are the one, where relations were successfully converted into variable trees. The performance increases with each model, however the interpretability decreases. We also compared our method to the conditional random fields(CRF). In the CRF method, tokens from all layers with window size 7 were taken as features for sequence prediction. On the converted examples CRF achieved F1 score of 80.8, which is comparable to our best model’s (LRCL) F1 score of 80.0.

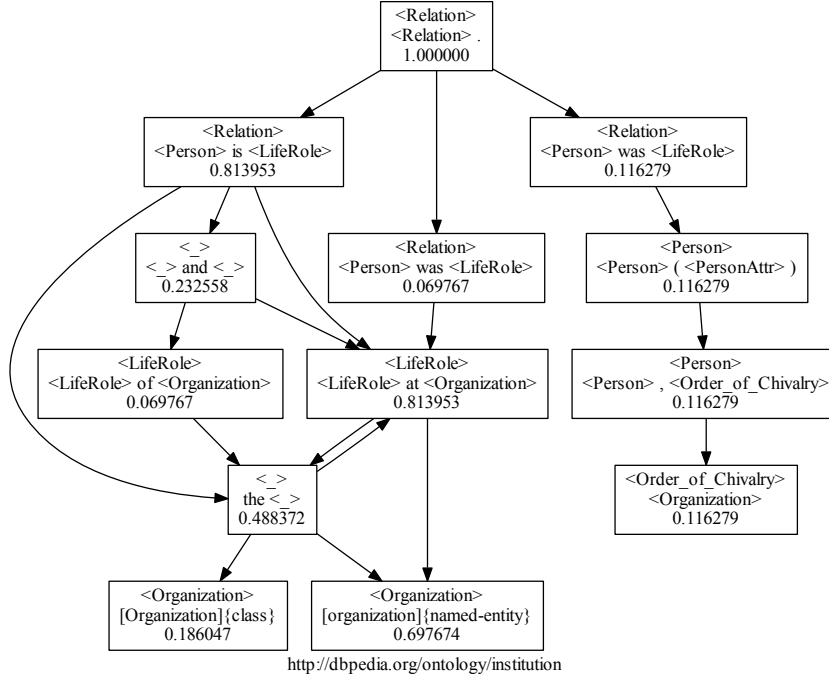


Figure 6. The automaton model for *institution* relation. On the bottom of each node, the fraction of training variable trees that contain this node, is displayed.

5. Related Work

There are many known approaches to ontology learning and semantic parsing, however, to the best of our knowledge, this is the first work to jointly learn an ontology and semantic parser. In the following sections, we make comparisons to other work on semantic parsing, ontology learning, grammar induction and others.

5.1. Semantic parsing

The goal of semantic parsing is to map text to meaning representations. Several approaches have used Combinatory categorial grammar (CCG) and lambda calculus as a meaning representation [16,17]. CCG grammar closely connects syntax and semantics with a lexicon, where each entry consist of a term, a syntactical category and a lambda statement. Similarly, our context-free grammar contains production rules. Some of these rules do not contain lexical tokens (the grammar is not lexicalized), which gives ability to express some relations with a single rule. For instance, to parse *jazz drummer*, rule $\langle \text{Musician.Type} \rangle ::= \langle \text{Musical.Genre} \rangle \langle \text{Musician.Type} \rangle$ is used to directly express the relation, which determines the genre of the musician. Lambda calculus may provide a more formal meaning representation than semantic trees, but the lexicon of CCG requires

mappings to lambda statements. Other approaches use dependency-based compositional semantics [18], ungrounded graphs [19], etc. as meaning representations.

Early semantic parsers were trained on datasets, such as Geoquery [20] and Atis [6], that map sentences to domain-specific databases. Later on datasets for question answering based on Freebase were created – Free917 [5] and WebQuestions [21]. These datasets contain short questions from multiple domains, and since the meaning representations are formed of Freebase concepts, they allow reasoning over Freebase’s ontology, which is much richer than databases in GeoQuery and Atis. All those datasets were constructed by either forming sentences given the meaning representation or vice-versa. Consequently, systems that were trained and evaluated on these datasets, might not work on sentences that cannot be represented by the underlying ontology. To overcome this limitation [17] developed an open vocabulary semantic parser. Their approach uses a CCG parser on questions to form lambda statements, which besides Freebase vocabulary contain underspecified predicates. These lambda statements are together with answers – Freebase entities – used to learn a low-dimensional probabilistic database, which is then used to answer fill-in-the-blank natural language questions. In a very similar fashion, [22] defines underspecified entities, types and relations, when the corresponding concept does not exist in Freebase. In contrast, the purpose of our method is to identify new concepts and ground them in the ontology.

5.2. Ontology Learning

Many ontology learning approaches address the same ontology components as our approach. However, their goal is to learn only the salient concepts for a particular domain, while our goal is to learn all the concepts (including instances, like particular organizations), so that they can be used in the meaning representation. As survey by [23] summarizes, the learning mechanisms are based either on statistics, linguistics, or logic. Our approach is unique because part of our ontology is constructed from the grammar. Many approaches use lexico-syntactic patterns for ontology learning. These are often based on dependency parses, like in [3,24]. Our approach does not rely on linguistic preprocessing, which makes it suitable for non-standard texts and poorly resourced languages. Our approach also builds patterns, however in form of grammar rules. Instead of lexico-syntactic patterns, which contain linguistic classes, our approach models semantic patterns, which contain semantic classes, like *Person* and *Color*. These patterns are constructed in advance, which is sometimes difficult because the constructor is not always aware of all the phenomena that is expressed in the input text. Our approach allows to create a small number of seed patterns in advance, then explore other patterns through the process of grammar learning. A similar bootstrapping semi-automatic approach to ontology learning was developed in [25], where the user validates lexicalizations of a particular relation to learn new instances, and in [26], where the user validates newly identified terms, while in our approach the user validates grammar rules to learn the composition of whole sentences. A similar approach with combining DBpedia with Wikipedia for supervised learning has been taken in [27], however their focus is more on lexicalization of relations and classes.

5.3. Grammar induction

Our goal was to develop a semi-automatic method that induces a grammar suitable for our scenario, in which an ontology is extracted, and text is parsed into semantic trees.

A survey by [28] compares several papers on grammar induction. According to their classification, our method falls into unsupervised, text-based (no negative examples of sentences) methods. Many such methods induce context-free grammars. However, their focus is more on learning syntactic structures rather than semantic. This is evident in evaluation strategies, where their parse trees are compared against golden parse trees in treebanks, like Penn treebank [29], which are annotated according to syntactic policies. Furthermore, our grammar should not be limited to a specific form, like for instance Chomsky normal form or Greibach normal form, instead it may contain arbitrary context-free rules. Several algorithms, like ours, employ the greedy strategy of grammar induction, where the grammar is updated with the best decision at each step. Whereas our method adds a rule after all sentences are parsed, The Incremental Parsing algorithm [30] updates the grammar after each sentence. This is also done in ADIOS method [31], where it has been shown that order of sentences affects the grammar. Our method employs frequency analysis and human supervision to control the grammar construction, while others use Minimum Description Length principle [32], clustering of sequences [33], or significance of word co-occurrences [34].

5.4. Other Approaches

Related work linking short terms to ontology concepts [35] is designed similarly as our approach in terms of bootstrapping procedure to induce patterns. But instead of inducing context-free grammar production rules, suggestions for rewrite rules that transform text directly to ontology language are provided. Another bootstrapping semi-automatic approach was developed for knowledge base population [36]. The task of knowledge base population is concerned only with extracting instances and relations given the ontology. In our work we also extract the backbone of the ontology – classes and taxonomic relations. Also, many other approaches focus only on one aspect of knowledge extraction, like taxonomy extraction [37,38] or relation extraction [14,39]. Combining these approaches can lead to cumbersome concept matching problems. This problem was also observed by [40]. Their system OntoUSP tries to overcome this by unsupervised inducing and populating a probabilistic grammar to solve question answering problem. However, the result are logical-form clusters connected in an *isa* hierarchy, not grounded concepts, which are connected with an existing ontology.

6. Discussion

We have presented an approach for joint ontology learning and semantic parsing. The approach was evaluated by building an ontology representing biographies of people. The first sentences of person Wikipedia pages and the combination of DBpedia and Freebase were used as a dataset. This dataset was suitable for our approach, because the text is equipped with human tagged annotations, which are already linked to the ontology. In other cases a named entity disambiguation would be needed to obtain the annotations. The next trait of the dataset, that is suitable for our approach, is the homogeneous style of writing. Otherwise, if the style was more heterogeneous, the users would have to participate in more iterations to achieve the same level of coverage. The participation of the users may be seen a cost, but on the other hand it allows them to learn about the

dataset without reading it all. The users does not learn so much about specific facts as they learn about the second order information, like what types of relations are expressed and their distribution.

Semantic trees offer a compact tree-structured meaning representation, which could be exploited for scenarios not covered by this paper, like relation type discovery and question answering. Furthermore, they can be used for more interpretable representation of meaning, like the automaton representation in Figure 6, compared to some other methods, like the one based on neural networks [41]. Our approach may not be superior on one specific part of the ontology learning, but it rather provides an integrated approach for learning on several levels of the ontology. Also, our approach does not use syntactic analysis, like part of speech tags or dependency parsing, which makes our approach more language independent and useful for non-standard texts, where such analysis is not available. On the other hand, we are looking into integrating syntactic analysis for future work. One scenario is to automatically detect the property of the rule. Another idea for future work is to integrate some ideas from other grammar induction methods to detect meaningful patterns without relying on the annotation of text.

Acknowledgements

This work was supported by Slovenian Research Agency and the ICT Programme of the EC under XLike (FP7-ICT-288342-STREP) and XLime (FP7-ICT-611346).

References

- [1] Oren Etzioni, Michele Banko, and Michael J Cafarella. Machine reading. In *AAAI*, volume 6, pages 1517–1519, 2006.
- [2] Philipp Cimiano and Johanna Völker. Text2onto. In *Natural language processing and information systems*, pages 227–238. Springer, 2005.
- [3] Amal Zouaq and Roger Nkambou. Evaluating the Generation of Domain Ontologies in the Knowledge Puzzle Project. *IEEE Trans. on Knowl. and Data Eng.*, 21(11):1559–1572, November 2009.
- [4] Tom M. Mitchell, Justin Betteridge, Andrew Carlson, Estevam Hruschka, and Richard Wang. Populating the Semantic Web by Macro-reading Internet Text. In *Proceedings of the 8th International Semantic Web Conference*, pages 998–1002, 2009.
- [5] Qingqing Cai and Alexander Yates. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2013.
- [6] Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the Scope of the ATIS Task: The ATIS-3 Corpus. In *Proceedings of the Workshop on Human Language Technology*, pages 43–48, 1994.
- [7] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, 2008.
- [8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.
- [9] Laura Kallmeyer. *Parsing Beyond Context-Free Grammars*. Springer Publishing Company, Incorporated, 1st edition, 2010.
- [10] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195, 2015.

- [11] Peter Norvig. Techniques for automatic memoization with applications to context-free parsing. *Comput. Linguist.*, 17(1):91–98, March 1991.
- [12] Richard A. Frost and Rahmatullah Hafiz. A new top-down parsing algorithm to accommodate ambiguity and left recursion in polynomial time. *SIGPLAN Not.*, 41(5):46–54, May 2006.
- [13] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June 1975.
- [14] Razvan C. Bunescu and Raymond J. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731, 2005.
- [15] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.
- [16] Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, 2011.
- [17] Jayant Krishnamurthy and Tom M Mitchell. Learning a compositional semantics for Freebase with an open predicate vocabulary. *Transactions of the Association for Computational Linguistics*, 3:257–270, 2015.
- [18] Percy Liang, Michael I Jordan, and Dan Klein. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446, 2013.
- [19] Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392, 2014.
- [20] John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055, 1996.
- [21] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*, pages 1533–1544, 2013.
- [22] Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. Scalable semantic parsing with partial ontologies. In *Proceedings of the 2015 Association for Computational Linguistics*, July 2015.
- [23] Wilson Wong, Wei Liu, and Mohammed Bennis. Ontology learning from text: A look back and into the future. *ACM Computing Surveys (CSUR)*, 44(4):20, 2012.
- [24] Johanna Völker, Pascal Hitzler, and Philipp Cimiano. Acquisition of OWL DL axioms from lexical resources. In *The Semantic Web: Research and Applications*, pages 670–685. Springer, 2007.
- [25] Wei Liu, Albert Weichselbraun, Arno Scharl, and Elizabeth Chang. Semi-automatic ontology extension using spreading activation. *Journal of Universal Knowledge Management*, 1:50–58, 2005.
- [26] Christopher Brewster, Fabio Ciravegna, and Yorick Wilks. User-centred ontology learning for knowledge management. In *Natural Language Processing and Information Systems*, pages 203–207. Springer, 2002.
- [27] Sebastian Walter, Christina Unger, and Philipp Cimiano. ATOLL—A framework for the automatic induction of ontology lexica. *Data & Knowledge Engineering*, 94:148–162, 2014.
- [28] Arianna D’Ulizia, Fernando Ferri, and Patrizia Grifoni. A survey of grammatical inference methods for natural language learning. *Artificial Intelligence Review*, 36(1):1–27, 2011.
- [29] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.
- [30] Yoav Seginer. Fast unsupervised incremental parsing. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 384, 2007.
- [31] Zach Solan, David Horn, Eytan Ruppín, and Shimon Edelman. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11629–11634, 2005.
- [32] Stephen Watkinson and Suresh Manandhar. A psychologically plausible and computationally effective approach to learning syntax. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*, page 20, 2001.
- [33] Alexander Clark. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*, page 13, 2001.
- [34] Christian Hähnig, Stefan Bordag, and Uwe Quasthoff. UnsuParse: unsupervised Parsing with unsuper-

November 2015

- vised Part of Speech Tagging. In *LREC*, 2008.
- [35] Janez Starc and Dunja Mladenić. Semi-automatic rule construction for semantic linking of relation arguments. In *Proceedings of the 17th International Multiconference Information Society - IS 2014*, 2013.
 - [36] Travis Wolfe, Mark Dredze, James Mayfield, Paul McNamee, Craig Harman, Tim Finin, and Benjamin Van Durme. Interactive knowledge base population. *arXiv preprint arXiv:1506.00301*, 2015.
 - [37] Roberto Navigli, Paola Velardi, and Stefano Faralli. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three, IJCAI'11*, pages 1872–1877, 2011.
 - [38] Rion Snow, Daniel Jurafsky, and Y. Andrew Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808, 2006.
 - [39] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106, March 2003.
 - [40] Hoifung Poon and Pedro Domingos. Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10*, pages 296–305, 2010.
 - [41] Phil Blunsom, Nando de Freitas, Edward Grefenstette, Karl Moritz Hermann, et al. A deep architecture for semantic parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, 2014.