On Extracting Relations using Distributional Semantics and a Tree Generalization

René Speck[1] and Axel-Cyrille Ngonga Ngomo[2]

[1] Leipzig University, AKSW Group, Hainstraße 11, Leipzig, Germany, D-04109
speck@informatik.uni-leipzig.de
[2] Paderborn University, DICE Group, Warburger Straße 100, Paderborn, Germany, D-33098
axel.ngonga@upb.de

Abstract. Extracting relations out of unstructured text is essential for a wide range of applications. Minimal human effort, scalability and high precision are desirable characteristics. We introduce a distant supervised closed relation extraction approach based on distributional semantics and a tree generalization. Our approach uses training data obtained from a reference knowledge base to derive dependency parse trees that might express a relation. It then uses a novel generalization algorithm to construct dependency tree patterns for the relation. Distributional semantics are used to eliminate false candidate patterns. We evaluate the performance in experiments on a large corpus using ninety target relations. Our evaluation results suggest that our approach achieves a higher precision than two state-of-the-art systems. Moreover, our results also underpin the scalability of our approach. Our open source implementation can be found at `https://github.com/dice-group/Ocelot`.

Keywords: Distant Supervision, Relation Extraction, Distributional Semantics, Tree Generalization

1 Introduction and Motivation

Knowledge extraction is the process of extracting facts in unstructured text automatically by, for instance, extracting relevant elements such as entities and relationships between these entities. Identifying token spans that constitute entity mentions and assigning types (e.g. `Person`) to these spans as well as relations (e.g. `spouse`) between entity mentions, is a key step to structuring knowledge from unstructured text for further analysis [10,18]. One application area of increasing importance is Question Answering (QA) with systems built on knowledge graphs like DBpedia. Such QA systems are typically composed of two stages: 1) the query analyzer, and 2) the retrieval stage [9,21]. Common techniques in the first stage are, for instance, named entity recognition and linking as well as relation extraction. Consider the following question from the QALD dataset [23]: "Is Michelle Obama the wife of Barack Obama?". A common way to answer this question with a QA system is to produce a semantic representation of this question within the first stages (see Listing 1.1). In the second stage, this semantic

representation is converted into SPARQL (see Listing 1.2) to query the knowledge base for further analytics and to create an answer to this question.

```
@prefix its: <http://www.w3.org/2005/11/its/rdf#> .
@prefix nif: <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
[ a nif:Phrase ;
       nif:anchorOf "Michelle Obama" ;
       nif:beginIndex "3"
       its:taIdentRef <http://dbpedia.org/resource/Michelle_Obama> ] .
[ a rdf:Statement ;
       rdf:subject <http://dbpedia.org/resource/Michelle_Obama> ;
       rdf:predicate <http://dbpedia.org/ontology/spouse> ;
       rdf:object <http://dbpedia.org/resource/Barack_Obama> ] .
[ a nif:Phrase ;
       nif:anchorOf "Barack Obama" ;
       nif:beginIndex "30"
       its:taIdentRef <http://dbpedia.org/resource/Barack_Obama> ].
```

Listing 1.1. Example semantic representation of the question "Is Michelle Obama the wife of Barack Obama?" in an RDF/TURTLE serialization.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
ASK WHERE { dbr:Michelle_Obama dbo:spouse dbr:Barack_Obama}
```

Listing 1.2. A SPARQL query to ask DBpedia for the trueness of the statement.

Collections of semantically-typed relational patterns as provided by Patty [17] and Boa [8] are often used in the first stage of QA systems to match and link word patterns in questions to a knowledge base. For example, HAWK [22] uses Boa, whereas AskNow [5] uses Boa and Patty to match and link word patterns. QA systems require high precision, minimal human effort and scalability from the relation extraction components they rely on. With our approach, we improve upon the precision achieved by the state of the art while keeping the distant supervision and scalability it abides by.

We propose a closed relation extraction approach based on distant supervision by using distributed semantics and a tree generalization process. It extracts sets of trees from a corpus where each set expresses a target relation from a knowledge base. These trees are then generalized using both a tree generalization approach and distributional semantics. One of the main advantages of this paradigm is that it is less sensitive to semantic drift. In the state-of-the-art systems, Boa and Patty one pattern is often matched to several relations which results in a significant number of false positives [21]. For example the pattern "was born" appears 876 times in Patty and corresponds to six DBpedia predicates. In Boa, this pattern appears three times and corresponds to three DBpedia predicates. Whereas each of the computed generalized tree patterns matches to only one relation in our approach.

The rest of this paper is structured as follows. After reviewing previous work in Section 2, we introduce preliminaries in Section 3 and our proposed approach in Section 4. Subsequently, we present our evaluation in Section 5 and the error analyses in Section 6. We conclude by discussing our results in Section 7.

2 Previous Work

Numerous approaches for extracting relations have been developed in the recent past. Approaches for *closed relation extraction* [8,17] (in contrast to *open relation extraction* [3,4,6,12,24]), are based on vocabularies that define relations a priori, i.e., in a domain ontology or an extraction template. Consequently, such systems require no mapping of the extracted relations to a vocabulary and thus produce less uninformative or incoherent elements from unstructured text [1]. *Supervised learning approaches* are a core component of a vast number of relation extraction tools as they offer high precision and recall. The need for manually labeled training data makes these methods not scalable to thousands of relations found on the Web. More promising approaches are *semi-supervised bootstrapping approaches* [2,8,20] and *distant supervision approaches* [1,16,17], since these do not need a complete manually labeled training corpus. In recent years, distant supervision has become an important technique because of the availability of large knowledge bases. It utilizes facts from a knowledge base for labeling mentions of these facts in an unannotated corpus to create a training set. Thus, it fulfills the needs of large-scale applications with minimal human effort.

Boa [8] is a bootstrapping strategy for extracting RDF from unstructured data. Its idea is to use the Web of Data as background knowledge for the extraction of natural language patterns that represent predicates found on the Web of Data. These patterns are used to extract instance knowledge from natural language text. This knowledge is finally fed back into the Web of Data. Boa provides a repository of natural language representations of predicates found on the Web of Data.

Patty [17] is a large resource for textual patterns that denote binary relations between entities based on distant supervision. Patty uses frequent itemset mining and the patterns are semantically typed as well as organized into a subsumption taxonomy with scores, support and confidence. The taxonomy is available online but not in machine-readable data to use it by the community. We asked the authors to provide the source-code and the database with the results as well as the measures but we could not receive it.

One drawback of both state-of-the-art systems, Boa and Patty, is that one pattern can be matched to several relations which results in a significant number of false positives. Thus, this leads to a noisy behavior in applications such as Question Answering. Another drawback of these two systems is that both extract relations that are enclosed by named entities only. For instance, both systems cannot find the relation in the sentence "Michelle Obama and Barack Obama are married." as the verb which mentions a relation is not enclosed by the named entities. We address these drawbacks by operating on dependency parse trees and using a generalization approach inspired by [11] which tackles the semantic drift issue faced by many current approaches.

3 Preliminaries and Notations

In this section we define the terminology and notation used in this paper.

Corpus Let $w \in \Sigma^*$ be a word that is a finite sequence over an alphabet Σ and let s be a sentence that is a finite sequence of words $s = (w_i)_{i=1,2,...}$. We denote the set of words of a sentence s with $W(s)$. A corpus $\mathcal{C}$ is a set of sentences.

Knowledge Base Let $\mathcal{K} = (S, R, P, \gamma)$ be a knowledge base with a set of statements $S \subseteq R \times P \times R$ (i.e. facts), a set of resources R (i.e. things of the real world), a set of predicates P (i.e. relationships between things) and a labelling function that maps each resource and predicate to a word $\gamma : R \cup P \to \Sigma^*$.

Tree Let $T = (V, A, E, \phi_A, \psi)$ be an attributed dependency parse tree (directed and ordered) with a finite set of vertices V, a set of edges $E \subseteq V \times V$, a vertex labelling function family $\phi_A = \{\phi_a | a \in A\}$ where A is a finite set of vertex attributes[3] so that $\phi_a : V \to \Sigma_a^*$ as well as with an edge labelling function $\psi : E \to \Sigma_E^*$.

We refer for a specific tree T with $V(T)$ for vertices, $E(T)$ for edges, $\phi_{T,A}$ for the vertex labelling function family and ψ_T for the edge labelling function. We denote the root vertex with $root_T$ and the root dependency vertex with sn_T.

Subtree Let T be a tree and $v \in V(T)$. The ordered sequence of child vertices of v in T is denoted by $cn_T(v) = (u_i)_{i=1,2,...}$ with $u \in V(T)$. We then denote by $T(v)$ the subtree $T' = (V', A', E', \phi'_A, \psi')$ with $root_{T'} = v$, $V' = cn_T(v) \cup \{v\}$, $E' = E \cap V' \times V'$, $\phi'_A = \phi_{A \restriction V'}$ and $\psi' = \psi_{\restriction E'}$.

$\leq$ Relation For trees T_1 and T_2, we have $T_1 \leq T_2$ iff the following holds:

1. if sn_{T_2} exists, then:
   (a) $sn_{T_1} = root_{T_1}$, $sn_{T_2} = root_{T_2}$
   (b) $\phi_{T_1,lemma}(root_{T_1}) = \phi_{T_2,lemma}(root_{T_2})$
   (c) $\phi_{T_1,pos}(root_{T_1}) = \phi_{T_2,pos}(root_{T_2})$
2. if sn_{T_2} does not exist, then:
   (a) $attribute := \phi_{T_2,general}(root_{T_2})$
   (b) if $attribute = label$, then $A^* := \{label, lemma, pos\}$
   (c) if $attribute = lemma$, then $A^* := \{lemma, pos\}$
   (d) for each a in A^*
   $\phi_{T_1,a}(root_{T_1}) = \phi_{T_2,a}(root_{T_2})$
   (e) if $attribute \in \{pos, ner, domain, range\}$, then
   $\phi_{T_1,attribute}(root_{T_1}) = \phi_{T_2,attribute}(root_{T_2})$
3. for each edge $(root_{T_2}, v_2)$ in $E(T_2)$ there exists an edge $(root_{T_1}, v_1)$ in $E(T_1)$ with $\psi_{T_2}(root_{T_2}, v_2) = \psi_{T_1}(root_{T_1}, v_1)$ such that: $T(v_1) \leq T(v_2)$

We define $T_1 \simeq T_2$ as $T_1 \leq T_2$ and $T_2 \leq T_1$. $T_1 < T_2$ is defined as $T_1 \leq T_2$ and $T_1 \not\simeq T_2$.

[3] $A := \{label, lemma, pos, ner, domain, range, general\}$ are the vertex attributes used throughout this paper.

4 Approach

This section initially presents an overview of the data flow and subsequently provides insights into each package of our proposed framework.

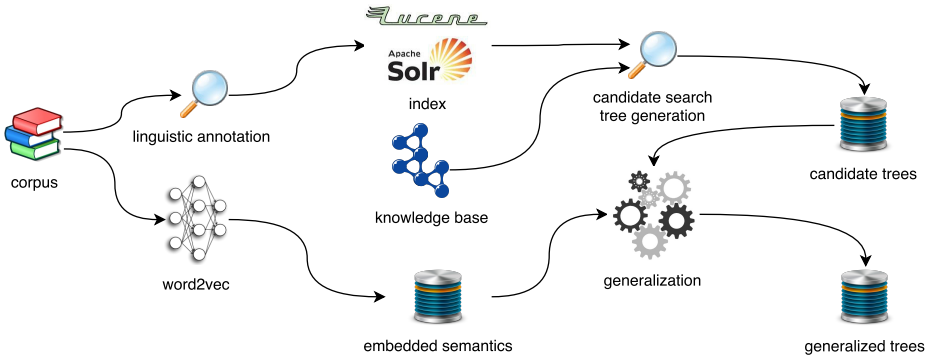

Fig. 1. The data flow of the proposed framework.

4.1 Overview

The data flow of our framework, dubbed Ocelot, is depicted in Figure 1. The goal is to harvest generalized dependency tree patterns, which are useful for a wide range of applications to extract relations from unstructured text.

Ocelot starts by preprocessing the corpus with natural language processing tools to acquire linguistic annotations. These annotations are stored in an index for a fast search. Thereafter, it queries a knowledge base for predicates which are the target relations, related resources as well as labels of these resources. These labels serve as search keywords to query candidate sentences in the index that might contain target relations. The dependency parse trees on these candidate sentences are created and stored. In the generalization step, the candidate trees are generalized by linguistic annotations as well as are scored and ranked. Ocelot relies on distant supervision and thus introduces errors by semantic drift [2,20]. To reduce this drift, it filters ambiguous trees. Ocelot utilizes embedded semantics by training word2vec [14] on the corpus. The vector representation of the labels from the knowledge base for the predicates as well as from other sources, for instance Wordnet, are used in the generalization step to filter out ambiguities among trees to reduce semantic drift. In the following, we explain each of these steps in detail.

4.2 Linguistic Annotation

We begin with an input corpus, which is first preprocessed. The core of the preprocessing consists of removing possible markup from the corpus (e.g. HTML

tags). We then sample the frequency distribution of the sentences' length (number of tokens in a sentence including the end punctuation). On this distribution, the mean μ and standard deviation σ are calculated to filter out sentences that are very long and thus require long processing time in the framework. Ocelot then selects sentences with a minimum of four[4] and a maximum of $\mu + 2\sigma$ tokens. Linguistic annotations (lemmas, POS-tags, named entities)[5] are computed for the selected sentences (with Stanford Core NLP in our current implementation). Based on the assumption "if two entities participate in a relation, at least one sentence that mentions these two entities might express that relation" stated in [19], we discard sentences which contain less than two named entities. The remaining sentences and annotations are stored in a Solr index for a fast search.

4.3 Candidate Selection

This step's main functions are to find candidate sentences from the index that might express target relations and to parse these candidate sentences to dependency parse trees. We rely on background knowledge from the given knowledge base to search for candidates. In the first step, the predicates with the highest numbers of resource instances are chosen from the knowledge base. These selected predicates serve as target relations in Ocelot. For each target relation p, the candidate selection queries $\mathcal{K}$ for the set

$$S_p = \{(s, p, o) : (s, p^*, o) \in \mathcal{K} \rightarrow p = p^*\}. \tag{1}$$

With the labelling function γ, given by the knowledge base, we get the labels for resources that we employ to search in the index. As some extended labeling functions are available for some knowledge bases (e.g. [7] proposes an extension of the method originally proposed in [13] to gather additional labels DBpedia), we assume the existence of a method which can generate extended labels for any resource $r \in R$ and call this method $\pi(r)$. Then, the sets Π_s and Π_o of all labels for all subject resp. object resources of a target relation are given by

$$\Pi_s = \bigcup_{(s,p,o) \in S_p} \pi(s) \cup \gamma(s) \text{ and } \Pi_o = \bigcup_{(s,p,o) \in S_p} \pi(o) \cup \gamma(o). \tag{2}$$

Therewith, the set Ω contains candidate sentences with tokens that mention subject and object resources of a target relation

$$\Omega = I(\mathcal{C}, \Pi_s) \cap I(\mathcal{C}, \Pi_o), \text{ with } I(A, B) = \{a | a \in A \wedge \exists b \in (B \cap W(a))\}\ . \tag{3}$$

To reduce semantic drift, only candidate sentences with tokens which mention subject and object resources and which are tagged as named entities by the linguistic annotation package are collected. These candidate sentences are parsed to candidate dependency parse trees.

[4] The shortest sentence with a relation has at least two tokens for the named entity arguments, one token for the relation mention and one for the end punctuation.

[5] Seven types are applied (`Place`, `Person`, `Organization`, `Money`, `Percent`, `Date`, `Time`).

4.4 Embedded Semantics

This step serves as preprocessing for the subsequent generalization step. We create word-level embeddings on the corpus and use predicate labels from the knowledge base to find semantically similar words in several sources. We trained the continuous skip-gram model [15] implemented in the open-source software word2vec[6] on the corpus. This model is based on the idea that similar words are more likely to be neighbours if their word level embeddings represent lexical and semantic regularities. Thus, this model predicts words within a certain range to either side of a current word and captures syntactic and semantic regularities of words. We retrieve labels from the knowledge base for each of our target relations as well as from Wikidata[7] and merge them for each relation. We call these labels "seed labels". Then, we sum up the vector representation of each of the seed labels to one vector, the seed vector. Thereafter, for each seed label we query OxfordDictionary,[8] Wordnik[9] and Wordnet[10] to find similar words. To reduce semantic drift in this step, we rearrange these similar words to the seed labels with the help of the vector representations. Hence, for each similar word we choose its vector representation and calculate the cosine similarity between this vector and all the seed vectors to measure the similarity. We rearrange all similar words to the relation where the cosine similarity between the seed vector of a relation and the vector of the similar word has the highest value.

4.5 Generalization

The input of the generalization steps are the candidate trees as well as the results of the embedded semantics module. The goal is to generalize, filter, score and rank the candidate trees. Function 1 together with Function 2 define the algorithm to generalize the extracted dependency parse trees.

Function 1 takes two input parameters, i.e., two trees T_1 and T_2, and returns a generalized or an empty tree T. In the first line, T is initialized with an empty tree. In the next two lines, the root vertices of both trees are preserved and Function 2 is called to generalize the vertices of the trees. The generalized tree is stored in T. In line 4, a set is generated containing all edge labels from outgoing edges of the root vertices that have the same edge labels in both trees. In lines 5 to 7, we iterate over all outgoing edges of the root vertices in the trees that have the same labels. For each combination, Function 1 is recursively computed. Lines 10 to 13 show that only edges which do not subsume another edge are preserved. Finally, line 14 adds the edge to tree T.

Function 2 defines the part of the algorithm to generalize vertices of two trees, T_1 and T_2. This function takes an empty or partly generalized tree T together with the trees T_1 and T_2 as well as the root vertices of these trees $v_1 = root_{T_1}$

[6] `https://code.google.com/archive/p/word2vec`
[7] `https://www.wikidata.org`
[8] `https://www.oxforddictionaries.com`
[9] `https://www.wordnik.com`
[10] `https://wordnet.princeton.edu`

Function 1: $generalize(T_1, T_2)$

```
1  initialize T with V(T) = ∅ and E(T) = ∅;
2  v_1 = root_{T_1}; v_2 = root_{T_2};
3  generalizeVertices(T, T_1, v_1, T_2, v_2) ;
4  L = {ψ_{T_1}(v_1, v_1') | (v_1, v_1') ∈ E(T_1), (v_2, v_2') ∈ E(T_2), ψ_{T_1}(v_1, v_1') = ψ_{T_2}(v_2, v_2')} ;
5  foreach l in L do
6  |   foreach v_1' with (v_1, v_1') ∈ E(T_1) and ψ_{T_1}(v_1, v_1') = l do
7  |   |   foreach v_2' with (v_2, v_2') ∈ E(T_2) and ψ_{T_2}(v_2, v_2') = l do
8  |   |   |   v' = root(generalize(T(v_1'), T(v_2'))) ;
9  |   |   |   add = true ;
10 |   |   |   foreach v_p with (v, v_p) ∈ E(T) do
11 |   |   |   |   if add = true then
12 |   |   |   |   |   if T(v_p) ≤ T(v') then add = false ;
13 |   |   |   |   |   if T(v') < T(v_p) then remove (v, v_p) from E(T);
14 |   |   |   if add = true then add edge (v, v') to E(T);
15 return T;
```

and $v_2 = root_{T_2}$. The generalized tree is stored in T. In the first line, the function compares the root vertices with the root dependency vertices of the trees. If the vertices have the same labels, a new vertex is created with this label and is set as root along with the root dependency vertex. In case the given vertices differ from the root dependency vertices but have the same label, lemma or POS-tag, a new vertex is created and is set with common attributes in lines 9 to 16. In lines 18 to 22, vertices without the same lemma and POS-tags are compared and added to the generalized tree in cases where their other attributes are equal.

After the tree generalization steps, Ocelot filters false candidate tree patterns with the embedded semantics package. It retains tree patterns that contain one of the labels that occur in the label set of the corresponding target relation.

Through the generalization process, the number of trees that a tree generalizes is observed. The trees are ranked by this number and by the number of vertices. Thus, a generalized tree that generalizes the most trees and has the fewest number of vertices has the highest rank.

Figure 2 illustrates a generalized tree pattern on two example sentences. The red edges in the dependency parse trees mark deleted edges, the remaining edges together with the linguistic annotations in bold font symbolize the resulting generalized dependency parse tree. Named entity arguments are illustrated in curly brackets and POS-tags in square brackets.

5 Evaluation

In this section, we present our experimental setup as well as the quantitative and qualitative evaluations we carried out.

Function 2: $generalizeVertices(T, T_1, v_1, T_2, v_2)$

```
1  if sn_{T_1} = v_1 and sn_{T_2} = v_2 then
2      if φ_{T_1,label}(v_1) = φ_{T_2,label}(v_2) then
3          initialize a new vertex v ;
4          sn_T := v; root(T) := v ;
5          φ_{T,label}(v) := φ_{T_1,label}(v_1);
6          V(T) := V(T) ∪ v;
7  else
8      if φ_{T_1,lemma}(v_1) = φ_{T_2,lemma}(v_2) and φ_{T_1,pos}(v_1) = φ_{T_2,pos}(v_2) then
9          initialize a new vertex v ;
10         φ_{T,lemma}(v) := φ_{T_1,lemma}(v_1);
11         φ_{T,pos}(v) := φ_{T_1,pos}(v_1);
12         V(T) := V(T) ∪ v;
13         if φ_{T_1,label}(v_1) = φ_{T_2,label}(v_2) then
14             φ_{T,label}(v) := φ_{T_1,label}(v_1);
15             φ_{T,general}(v) := label;
16         else φ_{T,general}(v) := lemma ;
17     else
18         foreach a ∈ {pos, ner, domain, range} do
19             if φ_{T_1,a}(v_1) = φ_{T_2,a}(v_2) then
20                 φ_{T,general}(v) := a;
21                 φ_{T,a}(v) := φ_{T_1,a}(v_1);
22                 V(T) := V(T) ∪ v;
```

5.1 Setup

For the experiments, we used the English Wikipedia as corpus $\mathcal{C}$ and DBpedia as knowledge base $\mathcal{K}$. For the index, we chose the implementation of Apache Solr with Lucene. The index contained 93,499,905 sentences in total with an average of $\mu = 22$ tokens per sentence and with a standard deviation of $\sigma = 15.8$ tokens. Our pipeline processed 88.44% of the sentences in the index. For the target relations queried from the DBpedia knowledge base, we chose the top ten of each combination of resource types[11] we took into account. Thus, we ended up with 90 target relations. Table 1 depicts an excerpt with the top-three target relations of each combination from DBpedia.

5.2 Quantitative Evaluation

In the quantitative evaluation, we first manually evaluated the filter approach to reduce semantic drift based on the embedded semantics package. Then, we compared the F1-Score, Precision and Recall of the results with the embedded semantics filter with the results of two state-of-the-art systems, Patty and Boa.

The precision (P), i.e., how many of the generalized trees express the correct relation for the top-k ranked generalized trees with and without the filter approach through the embedded semantics package, is depicted in Table 2. Each

[11] In our approach we utilize `Organization`, `Person` and `Place`.

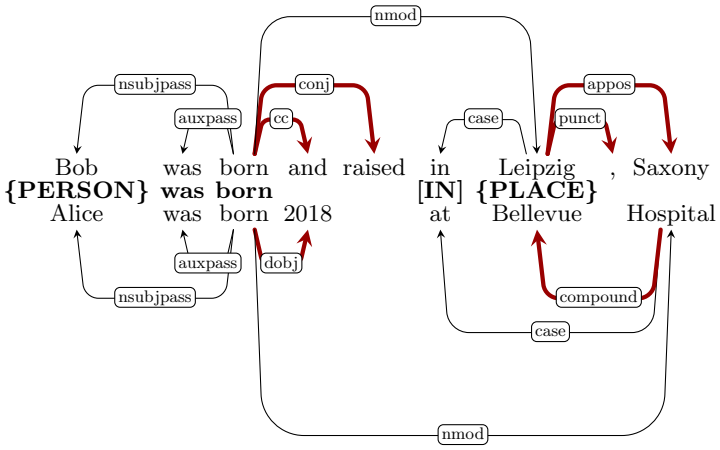

Fig. 2. The generalization process on two example sentences: "Bob was born and raised in Leipzig, Saxony." and "Alice was born 2018 at Bellevue Hospital.". The red edges mark deleted edges, the remaining edges together with the linguistic annotations in bold font symbolize the resulting generalized dependency tree pattern. Named entity arguments are illustrated in curly brackets and POS-tags in square brackets.

Table 1. Excerpt of top-three predicates for each domain/range combination.

rdfs:domain \ rdfs:range	`Organization`	`Person`	`Place`
`Organization`	dbo:sisterStation dbo:affiliation dbo:broadcastNetwork	dbo:bandMember dbo:formerBandMember dbo:notableCommander	dbo:hometown dbo:ground dbo:headquarter
`Person`	dbo:almaMater dbo:formerTeam dbo:debutTeam	dbo:parent dbo:child dbo:spouse	dbo:deathPlace dbo:birthPlace dbo:nationality
`Place`	dbo:tenant dbo:operator dbo:governingBody	dbo:leaderName dbo:architect dbo:saint	dbo:district dbo:locatedInArea dbo:department

row shows the top-k ranked trees, i.e., sorted by the number of trees a generalized tree generalizes and the number of vertices in a tree. The columns with NF denote the results without the filter and F with the filter. For instance, the top-1 ranked trees without filtering are 55 in total with a precision of 58.18%. With filtering, we obtain 19 top-1 ranked trees with a precision of 94.74%. Our results show that the precision without filtering decreases with higher values of k but that the precision with filtering remains more stable. For example, the precision without filtering for $k = 5$ is 2.93% points lower than for $k = 1$ while it decreases by only 0.3% when filtering is used. Because of the significant increase of the precision overall with filtering, we decided to filter the trees.

Table 2. Precision and number of trees, without filter (NF) and with filter (F).

top k	NF		F	
	# trees	P	# trees	P
1	55	58.18	19	94.74
2	102	57.84	30	93.33
3	143	57.34	40	95.00
4	182	54.95	47	93.62
5	219	55.25	54	94.44

We manually compared the patterns of Boa and Patty with the generalized trees of our approach Ocelot. The results are depicted in Table 3. We manually assessed the patterns for each tool with the measures precision (P), recall (R) and F-Score (F1). To be comparable with the other systems we created a pattern-like representation from our trees. We compared the top 1–5 patterns for the four target relations (`spouse`, `birthPlace`, `deathPlace` and `subsidiary`) supported by all three systems. Our approach reached higher values on all five k for all three measures.

Table 3. Precision, Recall and F-Score averaged over `spouse`, `birthPlace`, `deathPlace` and `subsidiary` for the top k patterns. Best results are in bold font.

top k	Boa P/R/F1	Patty P/R/F1	Ocelot P/R/F1
1	75.00/8.120/14.58	75.00/9.550/16.67	**100.0/13.12/22.92**
2	62.50/12.66/20.94	62.50/15.39/24.24	**87.50/21.23/33.64**
3	58.33/18.51/27.86	66.67/24.94/35.36	**91.67/34.35/48.93**
4	56.25/23.05/32.42	62.50/29.48/38.99	**91.67/40.19/54.73**
5	60.00/32.60/41.46	60.00/34.03/42.29	**86.67/43.77/56.55**

5.3 Qualitative Evaluation

We evaluated the quality of our approach against the state-of-the-art tool Boa. For the relation extraction with Boa, we chose the top-10 patterns from the Boa index as well as the top-10 from Ocelot. We compared the relation extraction results of Boa and Ocelot on the first 100 sentences of the top-three viewed articles about persons in Wikipedia. The results are shown in Table 4. We replaced named entities in sentences with their types as this is the preprocessing step for both tools. The × indicates that the system found no relation in the sentence. The bold marked relations in the table indicate correct extractions. With Boa, we extracted one correct relation, `birthPlace`, on one sentence, "(`Person`) was born in (`Place`)", but also a false positive relation `deathPlace` on the same sentence. With Ocelot, we were able to extract four correct relations.

A benefit of Ocelot is that it finds relations that not only enclosed by the named entities like Boa and Patty. That is the reason why Ocelot extracts the relation in: "(`Person`) and (`Person`) were married" but Boa and Patty cannot.

Table 4. Example relation extraction with Boa and Ocelot.

Examples	Boa	Ocelot
(`Person`) and his wife (`Person`)	×	**dbo:spouse**
(`Person`) and (`Person`) were married	×	**dbo:spouse**
(`Person`) met (`Person`)	dbo:spouse	dbo:spouse
(`Person`) was born in (`Place`)	dbo:deathPlace **dbo:birthPlace**	**dbo:birthPlace**
(`Person`) was born in 1905 in (`Place`)	×	**dbo:birthPlace**
(`Person`) returned to (`Place`)	dbo:deathPlace dbo:birthPlace	×
(`Person`) moved to (`Place`)	dbo:deathPlace dbo:birthPlace	×

6 Error Analysis

Data extracted from semi-structured sources, such as DBpedia, often contains inconsistencies as well as misrepresented and incomplete information [25]. For instance, at the time of writing this paper, the DBpedia resource `dbr:England` is a subtype of `dbo:Person` and a `dbo:MusicalArtist`, instead of being an instance of `dbo:Place` and of `dbo:PopulatedPlace`. Consequently, the data used by our approach for distance supervision was partly erronenous. For example, the labels of `dbr:England` served as labels for target relations with person arguments, e.g. `spouse`, because `dbr:England` is of the wrong type in DBpedia.

The integration of multiple knowledge bases and a type check over multiple knowledge bases could potentially solve this type mismatch.

The low recall of Ocelot might be due to the missing coreference resolution system in the proposed approach. We aim to integrate such an approach into our framework in future works. Due to the filtering of trees with the embedded semantics package, it might be the case that trees counting as true positive are filtered out because their semantic is not covered by the embedded semantics package. Increasing the number of external sources may increase the recall of our system.

7 Conclusion

In this paper we presented our approach Ocelot, a distant supervised closed relation extraction approach based on distributional semantics and a tree generalization. In a two-fold evaluation, quantitative and qualitative, we showed that our approach harvests generalized dependency tree patterns of high quality, and that it extracts relations from unstructured text with its generalized trees of higher precision than two state-of-the-art systems.

With our contribution we push forward the quality of relation extraction and thus the quality of applications in areas such as Knowledge Base Population and Semantic Question Answering. Moreover, we provide the source-code of our approach together with the version numbers of all utilized tools and all settings as well as the datasets used in this paper at `https://github.com/dice-group/Ocelot`. We have now integrated the results of this work, the generalized trees, into the Fox framework, which can be found at `http://fox-demo.aksw.de`.

The main advantages of this framework are that it is open source, and provides additional features such as named entity recognition and disambiguation, linked data by several RDF serialisations[12] and a freely usable RESTful web service that is ready to use by the community. We have now provided a system for knowledge extraction out of unstructured text that presents the extracted entities and relations in a machine readable format to the community.

8 Acknowledgement

This work has been supported by the H2020 project HOBBIT (no. 688227), the BMWI projects GEISER (no. 01MD16014E) and OPAL (no. 19F2028A), the EuroStars projects DIESEL (no. 01QE1512C) and QAMEL (no. 01QE1549C).

References

1. Augenstein, I., Maynard, D., Ciravegna, F.: Relation extraction from the web using distant supervision. In: International Conference on Knowledge Engineering and Knowledge Management. pp. 26–41. Springer (2014)

[12] We provided an example of an RDF serialisation of the framework in Listing 1.1.

2. Curran, J.R., Murphy, T., Scholz, B.: Minimising semantic drift with mutual exclusion bootstrapping. In: In Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics. pp. 172–180 (2007)
3. Del Corro, L., Gemulla, R.: Clausie: Clause-based open information extraction. In: Proceedings of the 22Nd International Conference on World Wide Web. pp. 355–366. WWW '13, ACM, New York, NY, USA (2013). ,
4. Draicchio, F., Gangemi, A., Presutti, V., Nuzzolese, A.G.: Fred: From natural language text to rdf and owl in one click. In: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (eds.) The Semantic Web: ESWC 2013 Satellite Events. pp. 263–267. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
5. Dubey, M., Dasgupta, S., Sharma, A., Hoffner, K., Lehmann, J.: Asknow: A framework for natural language query formalization in sparql. In: Proc. of the Extended Semantic Web Conference 2016 (2016),
6. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for Open Information Extraction, pp. 1535–1545 (2011)
7. Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngonga Ngomo, A.C., Speck, R.: Defacto - temporal and multilingual deep fact validation. Web Semantics: Science, Services and Agents on the World Wide Web (2015),
8. Gerber, D., Ngonga Ngomo, A.C.: Bootstrapping the linked data web. In: 1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011 (2011)
9. Höffner, K., Walter, S., Marx, E., Usbeck, R., Lehmann, J., Ngonga Ngomo, A.C.: Survey on challenges of Question Answering in the Semantic Web. Semantic Web Journal **8**(6) (2017),
10. Krause, S., Li, H., Uszkoreit, H., Xu, F.: Large-scale learning of relation-extraction rules with distant supervision from the web. In: International Semantic Web Conference. pp. 263–278. Springer (2012)
11. Lehmann, J., Bühmann, L.: Autosparql: Let users query your knowledge base. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) The Semantic Web: Research and Applications. pp. 63–79. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
12. Mausam, Schmitz, M., Bart, R., Soderland, S., Etzioni, O.: Open language learning for information extraction. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. pp. 523–534. EMNLP-CoNLL '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012),
13. Mendes, P.N., Jakob, M., Garcia-Silva, A., Bizer, C.: Dbpedia spotlight: Shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems (I-Semantics) (2011)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR (2013),
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. pp. 3111–3119. NIPS'13, Curran Associates Inc., USA (2013),
16. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP. pp. 1003–1011. Association for Computational Linguistics (2009),

17. Nakashole, N., Weikum, G., Suchanek, F.: Patty: A taxonomy of relational patterns with semantic types. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. pp. 1135–1145. EMNLP-CoNLL '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012),
18. Ren, X., Wu, Z., He, W., Qu, M., Voss, C.R., Ji, H., Abdelzaher, T.F., Han, J.: Cotype: Joint extraction of typed entities and relations with knowledge bases. In: Proceedings of the 26th International Conference on World Wide Web. pp. 1015–1024 (2017)
19. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 148–163. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
20. Riloff, E., Jones, R.: Learning dictionaries for information extraction by multi-level bootstrapping. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence. pp. 474–479. AAAI '99/IAAI '99, American Association for Artificial Intelligence, Menlo Park, CA, USA (1999),
21. Singh, K., Mulang', I.O., Lytra, I., Jaradeh, M.Y., Sakor, A., Vidal, M.E., Lange, C., Auer, S.: Capturing knowledge in semantically-typed relational patterns to enhance relation linking. In: Proceedings of the Knowledge Capture Conference. pp. 31:1–31:8. K-CAP 2017, ACM, New York, NY, USA (2017). ,
22. Usbeck, R., Ngomo, A.C.N., Bühmann, L., Unger, C.: Hawk–hybrid question answering using linked data. In: European Semantic Web Conference. pp. 353–368. Springer (2015)
23. Usbeck, R., Ngonga Ngomo, A.C., Haarmann, B., Krithara, A., Röder, M., Napolitano, G.: 7th open challenge on question answering over linked data (QALD-7). In: Semantic Web Evaluation Challenge. pp. 59–69. Springer International Publishing (2017),
24. Yates, A., Cafarella, M., Banko, M., Etzioni, O., Broadhead, M., Soderland, S.: Textrunner: Open information extraction on the web. In: Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations. pp. 25–26. NAACL-Demonstrations '07, Association for Computational Linguistics, Stroudsburg, PA, USA (2007),
25. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment for Linked Data: A survey. Semantic Web Journal (2015),