OntoLearn Reloaded: A Graph-Based Algorithm for Taxonomy Induction

Paola Velardi*
Sapienza University of Rome

Stefano Faralli*
Sapienza University of Rome

Roberto Navigli*
Sapienza University of Rome

In 2004 we published in this journal an article describing OntoLearn, one of the first systems to automatically induce a taxonomy from documents and Web sites. Since then, OntoLearn has continued to be an active area of research in our group and has become a reference work within the community. In this paper we describe our next-generation taxonomy learning methodology, which we name OntoLearn Reloaded. Unlike many taxonomy learning approaches in the literature, our novel algorithm learns both concepts and relations entirely from scratch via the automated extraction of terms, definitions, and hypernyms. This results in a very dense, cyclic and potentially disconnected hypernym graph. The algorithm then induces a taxonomy from this graph via optimal branching and a novel weighting policy. Our experiments show that we obtain high-quality results, both when building brand-new taxonomies and when reconstructing sub-hierarchies of existing taxonomies.

1. Introduction

Ontologies have proven useful for different applications, such as heterogeneous data integration, information search and retrieval, question answering, and, in general, for fostering interoperability between systems. Ontologies can be classified into three main types (Sowa 2000), namely: i) formal ontologies, that is, conceptualizations whose categories are distinguished by axioms and formal definitions, stated in logic to support complex inferences and computations; ii) prototype-based ontologies, which are based on typical instances or prototypes rather than axioms and definitions in logic; iii) lexicalized (or terminological) ontologies, which are specified by subtype-supertype relations and describe concepts by labels or synonyms rather than by prototypical instances.

Here we focus on lexicalized ontologies because, in order to enable natural language applications such as semantically enhanced information retrieval and question answering, we need a clear connection between our formal representation of the

* Dipartimento di Informatica, Sapienza Università di Roma, Via Salaria, 113, 00198 Roma Italy.
E-mail: {velardi,faralli,navigli}@di.uniroma1.it.

Submission received: 17 December 2011; revised submission received: 28 July 2012; accepted for publication: 10 October 2012.

doi:10.1162/COLI_a_00146

domain and the language used to express domain meanings within text. And, in turn, this connection can be established by producing full-fledged lexicalized ontologies for the domain of interest. Manually constructing ontologies is a very demanding task, however, requiring a large amount of time and effort, even when principled solutions are used (De Nicola, Missikoff, and Navigli 2009). A quite recent challenge, referred to as **ontology learning**, consists of automatically or semi-automatically creating a lexicalized ontology using textual data from corpora or the Web (Gomez-Perez and Manzano-Mancho 2003; Biemann 2005; Maedche and Staab 2009; Petasis et al. 2011). As a result of ontology learning, the heavy requirements of manual ontology construction can be drastically reduced.

In this paper we deal with the problem of learning a taxonomy (i.e., the backbone of an ontology) entirely from scratch. Very few systems in the literature address this task. OntoLearn (Navigli and Velardi 2004) was one of the earliest contributions in this area. In OntoLearn taxonomy learning was accomplished in four steps: terminology extraction, derivation of term sub-trees via string inclusion, disambiguation of domain terms using a novel Word Sense Disambiguation algorithm, and combining the sub-trees into a taxonomy. The use of a static, general-purpose repository of semantic knowledge, namely, WordNet (Miller et al. 1990; Fellbaum 1998), prevented the system from learning taxonomies in technical domains, however.

In this paper we present OntoLearn Reloaded, a graph-based algorithm for learning a taxonomy from the ground up. OntoLearn Reloaded preserves the initial step of our 2004 pioneering work (Navigli and Velardi 2004), that is, automated terminology extraction from a domain corpus, but it drops the requirement for WordNet (thereby avoiding dependence on the English language). It also drops the term compositionality assumption that previously led to us having to use a Word Sense Disambiguation algorithm—namely, SSI (Navigli and Velardi 2005)—to structure the taxonomy. Instead, we now exploit textual definitions, extracted from a corpus and the Web in an iterative fashion, to automatically create a highly dense, cyclic, potentially disconnected hypernym graph. An optimal branching algorithm is then used to induce a full-fledged tree-like taxonomy. Further graph-based processing augments the taxonomy with additional hypernyms, thus producing a Directed Acyclic Graph (DAG).

Our system provides a considerable advancement over the state of the art in taxonomy learning:

- First, excepting for the manual selection of just a few upper nodes, this is the first algorithm that has been experimentally shown to build from scratch a new taxonomy (i.e., both concepts and hypernym relations) for arbitrary domains, including very technical ones for which gold-standard taxonomies do not exist.
- Second, we tackle the problem with no simplifying assumptions: We cope with issues such as term ambiguity, complexity of hypernymy patterns, and multiple hypernyms.
- Third, we propose a novel algorithm to extract an optimal branching from the resulting hypernym graph, which—after some recovery steps—becomes our final taxonomy. Taxonomy induction is the main theoretical contribution of the paper.
- Fourth, the evaluation is not limited, as it is in most papers, to the number of retrieved hypernymy relations that are found in a reference taxonomy.

Instead, we also analyze the extracted taxonomy in its entirety; furthermore, we acquire two "brand new" taxonomies in the domains of ARTIFICIAL INTELLIGENCE and FINANCE.

- Finally, our taxonomy-building workflow is fully implemented and the software components are either freely available from our Web site,[1] or reproducible.

In this paper we extend our recent work on the topic (Navigli, Velardi, and Faralli 2011) as follows: i) we describe in full detail the taxonomy induction algorithm; ii) we enhance our methodology with a final step aimed at creating a DAG, rather than a strict tree-like taxonomical structure; iii) we perform a large-scale multi-faceted evaluation of the taxonomy learning algorithm on six domains; and iv) we contribute a novel methodology for evaluating an automatically learned taxonomy against a reference gold standard.

In Section 2 we illustrate the related work. We then describe our taxonomy-induction algorithm in Section 3. In Section 4 we present our experiments, and discuss the results. Evaluation is both qualitative (on new ARTIFICIAL INTELLIGENCE and FINANCE taxonomies), and quantitative (on WordNet and MeSH sub-hierarchies). Section 5 is dedicated to concluding remarks.

2. Related Work

Two main approaches are used to learn an ontology from text: rule-based and distributional approaches. **Rule-based** approaches use predefined rules or heuristic patterns to extract terms and relations. These approaches are typically based on lexico-syntactic patterns, first introduced by Hearst (1992). Instances of relations are harvested from text by applying patterns aimed at capturing a certain type of relation (e.g., *X is a kind of Y*). Such lexico-syntactic patterns can be defined manually (Berland and Charniak 1999; Kozareva, Riloff, and Hovy 2008) or obtained by means of bootstrapping techniques (Girju, Badulescu, and Moldovan 2006; Pantel and Pennacchiotti 2006). In the latter case, a number of term pairs in the wanted relation are manually picked and the relation is sought within text corpora or the Web. Other rule-based approaches learn a taxonomy by applying heuristics to collaborative resources such as Wikipedia (Suchanek, Kasneci, and Weikum 2008; Ponzetto and Strube 2011), also with the supportive aid of computational lexicons such as WordNet (Ponzetto and Navigli 2009).

Distributional approaches, instead, model ontology learning as a clustering or classification task, and draw primarily on the notions of distributional similarity (Pado and Lapata 2007; Cohen and Widdows 2009), clustering of formalized statements (Poon and Domingos 2010), or hierarchical random graphs (Fountain and Lapata 2012). Such approaches are based on the assumption that paradigmatically-related concepts[2] appear in similar contexts and their main advantage is that they are able to discover relations that do not explicitly appear in the text. They are typically less accurate, however, and the selection of feature types, notion of context, and similarity metrics vary considerably depending on the specific approach used.

1 `http://lcl.uniroma1.it/ontolearn_reloaded` and `http://ontolearn.org`.

2 Because we are concerned with lexical taxonomies, in this paper we use the words *concepts* and *terms* interchangeably.

Recently, Yang and Callan (2009) presented a semi-supervised taxonomy induction framework that integrates contextual, co-occurrence, and syntactic dependencies, lexico-syntactic patterns, and other features to learn an ontology metric, calculated in terms of the semantic distance for each pair of terms in a taxonomy. Terms are incrementally clustered on the basis of their ontology metric scores. In their work, the authors assume that the set of ontological concepts C is known, therefore taxonomy learning is limited to finding relations between given pairs in C. In the experiments, they only use the word senses within a particular WordNet sub-hierarchy so as to avoid any lexical ambiguity. Their best experiment obtains a 0.85 precision rate and 0.32 recall rate in replicating is-a links on 12 focused WordNet sub-hierarchies, such as PEOPLE, BUILDING, PLACE, MILK, MEAL, and so on.

Snow, Jurafsky, and Ng (2006) propose the incremental construction of taxonomies using a probabilistic model. In their work they combine evidence from multiple supervised classifiers trained on very large training data sets of hyponymy and cousin relations. Given the body of evidence obtained from all the relevant word pairs in a lexico-syntactic relation, the taxonomy learning task is defined probabilistically as the problem of finding the taxonomy that maximizes the probability of having that evidence (a supervised logistic regression model is used for this). Rather than learning a new taxonomy from scratch, however, this approach aims at attaching new concepts under the appropriate nodes of an existing taxonomy (i.e., WordNet). The approach is evaluated by manually assessing the quality of the single hypernymy edges connecting leaf concepts to existing ones in WordNet, with no evaluation of a full-fledged structured taxonomy and no restriction to a specific domain. A related, weakly supervised approach aimed at categorizing named entities, and attaching them to WordNet leaves, was proposed by Pasca (2004). Other approaches use formal concept analysis (Cimiano, Hotho, and Staab 2005), probabilistic and information-theoretic measures to learn taxonomies from a folksonomy (Tang et al. 2009), and Markov logic networks and syntactic parsing applied to domain text (Poon and Domingos 2010).

The work closest to ours is that presented by Kozareva and Hovy (2010). From an initial given set of root concepts and basic level terms, the authors first use Hearst-like lexico-syntactic patterns iteratively to harvest new terms from the Web. As a result a set of hyponym–hypernym relations is obtained. Next, in order to induce taxonomic relations between intermediate concepts, the Web is searched again with surface patterns. Finally, nodes from the resulting graph are removed if the out-degree is below a threshold, and edges are pruned by removing cycles and selecting the longest path in the case of multiple paths between concept pairs. Kozareva and Hovy's method has some limitations, which we discuss later in this paper. Here we note that, in evaluating their methodology, the authors discard any retrieved nodes not belonging to a WordNet sub-hierarchy (they experiment on PLANTS, VEHICLES, and ANIMALS), thus it all comes down to Yang and Callan's (2009) experiment of finding relations between a pre-assigned set of nodes.

In practice, none of the algorithms described in the literature was actually applied to the task of creating a new taxonomy for an arbitrary domain of interest truly from scratch. Instead, what is typically measured is the ability of a system to reproduce as far as possible the relations of an already existing taxonomy (a common test is WordNet or the Open Directory Project[3]), when given the set of domain concepts. Evaluating against a gold standard is, indeed, a reasonable validation methodology. The claim to be

3 http://www.dmoz.org/.

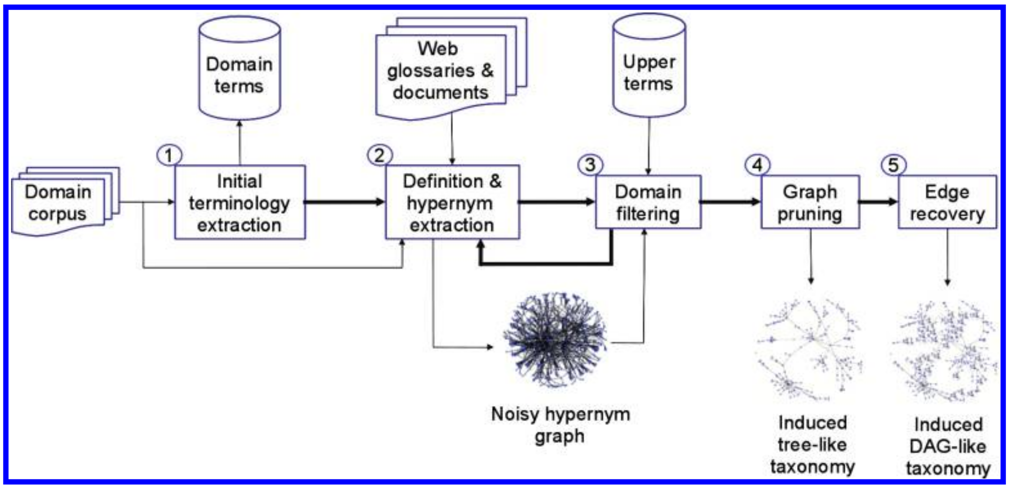
Domain terms
Web glossaries & documents
Upper terms
Domain corpus
1
Initial terminology extraction
2
Definition & hypernym extraction
3
Domain filtering
4
Graph pruning
5
Edge recovery
Noisy hypernym graph
Induced tree-like taxonomy
Induced DAG-like taxonomy

Figure 1
The OntoLearn Reloaded taxonomy learning workflow.

"automatically building" a taxonomy needs also to be demonstrated on new domains for which no a priori knowledge is available, however. In an unknown domain, taxonomy induction requires the solution of several further problems, such as identifying domain-appropriate concepts, extracting appropriate hypernym relations, and detecting lexical ambiguity, whereas some of these problems can be ignored when evaluating against a gold standard (we will return to this issue in detail in Section 4). In fact, the predecessor of OntoLearn Reloaded, that is, OntoLearn (Navigli and Velardi 2004), suffers from a similar problem, in that it relies on the WordNet taxonomy to establish paradigmatic connections between concepts.

3. The Taxonomy Learning Workflow

OntoLearn Reloaded starts from an initially empty directed graph and a corpus for the domain of interest (e.g., an archive of artificial intelligence papers). We also assume that a small set of upper terms (*entity*, *abstraction*, etc.), which we take as the end points of our algorithm, has been manually defined (e.g., from a general purpose taxonomy like WordNet) or is available for the domain.[4] Our taxonomy-learning workflow, summarized in Figure 1, consists of five steps:

1. **Initial Terminology Extraction (Section 3.1):** The first step applies a term extraction algorithm to the input domain corpus in order to produce an initial domain terminology as output.

2. **Definition & Hypernym Extraction (Section 3.2):** Candidate definition sentences are then sought for the extracted domain terminology. For each term t, a **domain-independent classifier** is used to select well-formed definitions from the candidate sentences and extract the corresponding hypernyms of t.

4 Although very few domain taxonomies are available, upper (core) concepts have been defined in several domains, such as MEDICINE, ART, ECONOMY, and so forth.

3. **Domain Filtering (Section 3.3):** A domain filtering technique is applied to filter out those definitions that do not pertain to the domain of interest. The resulting domain definitions are used to populate the directed graph with hypernymy relations connecting *t* to the extracted hypernym *h*. Steps (2) and (3) are then iterated on the newly acquired hypernyms, until a termination condition occurs.
4. **Graph Pruning (Section 3.4):** As a result of the iterative phase we obtain a dense hypernym graph that potentially contains cycles and multiple hypernyms for most nodes. In this step we combine a novel weighting strategy with the Chu-Liu/Edmonds algorithm (Chu and Liu 1965; Edmonds 1967) to produce an optimal branching (i.e., a tree-like taxonomy) of the initial noisy graph.
5. **Edge Recovery (Section 3.5):** Finally, we optionally apply a recovery strategy to reattach some of the hypernym edges deleted during the previous step, so as to produce a full-fledged taxonomy in the form of a DAG.

We now describe in full detail the five steps of OntoLearn Reloaded.[5]

3.1 Initial Terminology Extraction

Domain terms are the building blocks of a taxonomy. Even though in many cases an initial domain terminology is available, new terms emerge continuously, especially in novel or scientific domains. Therefore, in this work we aim at fully automatizing the taxonomy induction process. Thus, we start from a text corpus for the domain of interest and extract domain terms from the corpus by means of a terminology extraction algorithm. For this we use our term extraction tool, TermExtractor,[6] that implements measures of domain consensus and relevance to harvest the most relevant terms for the domain from the input corpus.[7] As a result, an initial domain terminology $T^{(0)}$ is produced that includes both single- and multi-word expressions (such as, respectively, *graph* and *flow network*). We add one node to our initially empty graph $G_{noisy} = (V_{noisy}, E_{noisy})$ for each term in $T^{(0)}$—that is, we set $V_{noisy} := T^{(0)}$ and $E_{noisy} := \emptyset$.

In Table 1 we show an excerpt of our ARTIFICIAL INTELLIGENCE and FINANCE terminologies (cf. Section 4 for more details). Note that our initial set of domain terms (and, consequently, nodes) will be enriched with the new hypernyms acquired during the subsequent iterative phase, described in the next section.

3.2 Definition and Hypernym Extraction

The aim of our taxonomy induction algorithm is to learn a hypernym graph by means of several iterations, starting from $T^{(0)}$ and stopping at very general terms U, that we take as the end point of our algorithm. The upper terms are chosen from WordNet topmost

5 A video of the first four steps of OntoLearn Reloaded is available at `http://www.youtube.com/watch?v=-k3cOEoI_Dk`.
6 `http://lcl.uniroma1.it/termextractor`.
7 TermExtractor has already been described in Sclano and Velardi (2007) and in Navigli and Velardi (2004); therefore the interested reader is referred to these papers for additional details.

Table 1
An excerpt of the terminology extracted for the ARTIFICIAL INTELLIGENCE and FINANCE domains.

ARTIFICIAL INTELLIGENCE		
acyclic graph	parallel corpus	flow network
adjacency matrix	parse tree	pattern matching
artificial intelligence	partitioned semantic network	pagerank
tree data structure	pathfinder	taxonomic hierarchy

FINANCE		
investor	shareholder	open economy
bid-ask spread	profit maximization	speculation
long term debt	shadow price	risk management
optimal financing policy	ratings	profit margin

synsets. In other words, U contains all the terms in the selected topmost synsets. In Table 2 we show representative synonyms of the upper-level synsets that we used for the ARTIFICIAL INTELLIGENCE and FINANCE domains. Seeing that we use high-level concepts, the set U can be considered domain-independent. Other choices are of course possible, especially if an upper ontology for a given domain is already available.

For each term $t \in T^{(i)}$ (initially, $i = 0$), we first check whether t is an upper term (i.e., $t \in U$). If it is, we just skip it (because we do not aim at extending the taxonomy beyond an upper term). Otherwise, definition sentences are sought for t in the domain corpus and in a portion of the Web. To do so we use Word-Class Lattices (WCLs) (Navigli and Velardi 2010, introduced hereafter), which is a domain-independent machine-learned classifier that identifies definition sentences for the given term t, together with the corresponding hypernym (i.e., lexical generalization) in each sentence.

For each term in our set $T^{(i)}$, we then automatically extract definition candidates from the domain corpus, Web documents, and Web glossaries, by harvesting all the sentences that contain t. To obtain on-line glossaries we use a Web glossary extraction system (Velardi, Navigli, and D'Amadio 2008). Definitions can also be obtained via a lightweight bootstrapping process (De Benedictis, Faralli, Navigli 2013).

Finally, we apply WCLs and collect all those sentences that are classified as definitional. We show some terms with their definitions in Table 3 (first and second column, respectively). The extracted hypernym is shown in italics.

Table 2
The set of upper concepts used in OntoLearn Reloaded for AI and FINANCE (only representative synonyms from the corresponding WordNet synsets are shown).

ability#n#1	abstraction#n#6	act#n#2	code#n#2
communication#n#2	concept#n#1	data#n#1	device#n#1
discipline#n#1	entity#n#1	event#n#1	expression#n#6
research#n#1	instrumentality#n#1	knowledge#n#1	knowledge domain#n#1
language#n#1	methodology#n#2	model#n#1	organization#n#1
person#n#1	phenomenon#n#1	process#n#1	property#n#2
quality#n#1	quantity#n#1	relation#n#1	representation#n#2
science#n#1	system#n#2	technique#n#1	theory#n#1

Table 3
Some definitions for the ARTIFICIAL INTELLIGENCE domain (defined term in bold, extracted hypernym in italics).

Term	Definition	Weight	Domain?
adjacency matrix	an **adjacency matrix** is a zero-one *matrix*	1.00	✓
flow network	in graph theory, a **flow network** is a directed *graph*	0.57	✓
flow network	global cash **flow network** is an online *company* that specializes in education and training courses in teaching the entrepreneurship	0.14	×

Table 4
Example definitions (defined terms are marked in bold face, their hypernyms in italics).

[In arts, a **chiaroscuro**]$_{\text{DF}}$ [is]$_{\text{VF}}$ [a monochrome *picture*]$_{\text{GF}}$.
[In mathematics, a **graph**]$_{\text{DF}}$ [is]$_{\text{VF}}$ [a *data structure*]$_{\text{GF}}$ [that consists of . . .]$_{\text{REST}}$.
[In computer science, a **pixel**]$_{\text{DF}}$ [is]$_{\text{VF}}$ [a *dot*]$_{\text{GF}}$ [that is part of a computer image]$_{\text{REST}}$.
[**Myrtales**]$_{\text{DF}}$ [are an order of]$_{\text{VF}}$ [*flowering plants*]$_{\text{GF}}$ [placed as a basal group . . .]$_{\text{REST}}$.

3.2.1 Word-Class Lattices. We now describe our WCL algorithm for the classification of definitional sentences and hypernym extraction. Our model is based on a formal notion of textual definition. Specifically, we assume a definition contains the following fields (Storrer and Wellinghoff 2006):

- The DEFINIENDUM field (DF): this part of the definition includes the **definiendum** (that is, the word being defined) and its modifiers (e.g., "In computer science, a *pixel*");
- The DEFINITOR field (VF): which includes the verb phrase used to introduce the definition (e.g., "is");
- The DEFINIENS field (GF): which includes the **genus phrase** (usually including the hypernym, e.g., "a *dot*");
- The REST field (RF): which includes additional clauses that further specify the *differentia* of the definiendum with respect to its genus (e.g., "that is part of a computer image").

To train our definition extraction algorithm, a data set of textual definitions was manually annotated with these fields, as shown in Table 4.[8] Furthermore, the single- or multi-word expression denoting the hypernym was also tagged. In Table 4, for each sentence the *definiendum* and its hypernym are marked in bold and italics, respectively. Unlike other work in the literature dealing with definition extraction (Hovy et al. 2003; Fahmi and Bouma 2006; Westerhout 2009; Zhang and Jiang 2009), we covered not only a variety of definition styles in our training set, in addition to the classic *X is a Y* pattern, but also a variety of domains. Therefore, our WCL algorithm requires no re-training when changing the application domain, as experimentally demonstrated by Navigli and Velardi (2010). Table 5 shows some non-trivial patterns for the VF field.

8 Available on-line at: `http://lcl.uniroma1.it/wcl`.

Table 5
Some nontrivial patterns for the VF field.

is a term used to describe	is a specialized form of
is the genus of	was coined to describe
is a term that refers to a kind of	is a special class of
can denote	is the extension of the concept of
is commonly used to refer to	is defined both as

Starting from the training set, the WCL algorithm learns generalized definitional models as detailed hereafter.

Generalized sentences. First, training and test sentences are part-of-speech tagged with the TreeTagger system, a part-of-speech tagger available for many languages (Schmid 1995). The first step in obtaining a definitional pattern is word generalization. Depending on its frequency we define a word class as either a word itself or its part of speech. Formally, let $\mathcal{T}$ be the set of training sentences. We first determine the set F of words in $\mathcal{T}$ whose frequency is above a threshold θ (e.g., *the*, *a*, *an*, *of*). In our training sentences, we replace the defined term with the token $\langle$TARGET$\rangle$ (note that $\langle$TARGET$\rangle \in F$).

Given a new sentence $s = t_1, t_2, \dots, t_n$, where t_i is the i-th token of s, we generalize its words t_i to word classes t'_i as follows:

$$t'_i = \begin{cases} t_i & \text{if } t_i \in F \\ POS(t_i) & \text{otherwise} \end{cases}$$

that is, a word t_i is left unchanged if it occurs frequently in the training corpus (i.e., $t_i \in F$); otherwise it is replaced with its part of speech ($POS(t_i)$). As a result we obtain a generalized sentence s'. For instance, given the first sentence in Table 4, we obtain the corresponding generalized sentence: "In NNS, a $\langle$TARGET$\rangle$ is a JJ NN," where NN and JJ indicate the noun and adjective classes, respectively. Generalized sentences are doubly beneficial: First, they help reduce the annotation burden, in that many differently lexicalized sentences can be caught by a single generalized sentence; second, thanks to their reduction of the definition variability, they allow for a higher-recall definition model.

Star patterns. Let $\mathcal{T}$ again be the set of training sentences. In this step we associate a star pattern $\sigma(s)$ with each sentence $s \in \mathcal{T}$. To do so, let $s \in \mathcal{T}$ be a sentence such that $s = t_1, t_2, \dots, t_n$, where t_i is its i-th token. Given the set F of most frequent words in $\mathcal{T}$, the star pattern $\sigma(s)$ associated with s is obtained by replacing with * all the tokens $t_i \notin F$, that is, all the tokens that are non-frequent words. For instance, given the sentence "In arts, a chiaroscuro is a monochrome picture," the corresponding star pattern is "In *, a $\langle$TARGET$\rangle$ is a *," where $\langle$TARGET$\rangle$ is the defined term.

Sentence clustering. We then cluster the sentences in our training set $\mathcal{T}$ on the basis of their star pattern. Formally, let $\Sigma = (\sigma_1, \dots, \sigma_m)$ be the set of star patterns associated with the sentences in $\mathcal{T}$. We create a clustering $\mathcal{C} = (C_1, \dots, C_m)$ such that $C_i = \{s \in \mathcal{T} : \sigma(s) = \sigma_i\}$, that is, C_i contains all the sentences whose star pattern is σ_i.

As an example, assume σ_3 = "In *, a $\langle$TARGET$\rangle$ is a *." The first three sentences reported in Table 4 are all grouped into cluster C_3. We note that each cluster C_i contains sentences whose degree of variability is generally much lower than for any pair of sentences in $\mathcal{T}$ belonging to two different clusters.

Word-class lattice construction. The final step consists of the construction of a WCL for each sentence cluster, using the corresponding generalized sentences. Given such a cluster $C_i \in \mathcal{C}$, we apply a greedy algorithm that iteratively constructs the WCL.

Let $C_i = \{s_1, s_2, \ldots, s_{|C_i|}\}$ and consider its first sentence $s_1 = t_1, t_2, \ldots, t_n$. Initially, we create a directed graph $G = (V, E)$ such that $V = \{t_1, \ldots, t_n\}$ and $E = \{(t_1, t_2), (t_2, t_3), \ldots, (t_{n-1}, t_n)\}$. Next, for each $j = 2, \ldots, |C_i|$, we determine the alignment between sentence s_j and each sentence $s_k \in C_i$ such that $k < j$ according to the following dynamic programming formulation (Cormen, Leiserson, and Rivest 1990, pages 314–319):

$$M_{a,b} = \max \{M_{a-1,b-1} + S_{a,b}, M_{a,b-1}, M_{a-1,b}\}, \tag{1}$$

where $a \in \{0, \ldots, |s_k|\}$ and $b \in \{0, \ldots, |s_j|\}$, $S_{a,b}$ is a score of the matching between the a-th token of s_k and the b-th token of s_j, and $M_{0,0}$, $M_{0,b}$ and $M_{a,0}$ are initially set to 0 for all values of a and b.

The matching score $S_{a,b}$ is calculated on the generalized sentences s'_k and s'_j as follows:

$$S_{a,b} = \begin{cases} 1 & \text{if } t'_{k,a} = t'_{j,b} \\ 0 & \text{otherwise} \end{cases}$$

where $t'_{k,a}$ and $t'_{j,b}$ are the a-th and b-th tokens of s'_k and s'_j, respectively. In other words, the matching score equals 1 if the a-th and the b-th tokens of the two generalized sentences have the same word class.

Finally, the alignment score between s_k and s_j is given by $M_{|s_k|,|s_j|}$, which calculates the minimal number of misalignments between the two token sequences. We repeat this calculation for each sentence s_k ($k = 1, \ldots, j-1$) and choose the one that maximizes its alignment score with s_j. We then use the best alignment to add s_j to the graph G: We add to the set of nodes V the tokens of s'_j for which there is no alignment to s'_k and we add to E the edges $(t'_1, t'_2), \ldots, (t'_{|s_j|-1}, t'_{|s_j|})$.

Example. Consider the first three definitions in Table 4. Their star pattern is "In *, a ⟨TARGET⟩ is a *." The corresponding WCL is built as follows: The first part-of-speech tagged sentence, "In/IN arts/NN , a/DT ⟨TARGET⟩/NN is/VBZ a/DT monochrome/JJ *picture*/NN," is considered. The corresponding generalized sentence is "In NN$_1$, a ⟨TARGET⟩ is a JJ *NN*$_2$." The initially empty graph is thus populated with one node for each word class and one edge for each pair of consecutive tokens, as shown in Figure 2a. Note that we use a rectangle to denote the hypernym token $\boxed{\text{NN}_2}$. We also add to the graph a start node• and an end node ⦿, and connect them to the corresponding initial and final sentence tokens. Next, the second sentence, "In mathematics, a graph is a data structure that consists of...," is aligned to the first sentence. The alignment is perfect, apart from the $\boxed{\text{NN}_3}$ node corresponding to "data." The node is added to the graph together with the edges "a"$\rightarrow\boxed{\text{NN}_3}$ and $\boxed{\text{NN}_3}\rightarrow\boxed{\text{NN}_2}$ (Figure 2b, node and edges in bold). Finally, the third sentence in Table 4, "In computer science, a pixel is a dot that is part of a computer image," is generalized as "In NN$_4$ NN$_1$, a ⟨TARGET⟩ is a *NN*$_2$." Thus, a new node NN$_4$ is added, corresponding to "computer" and new

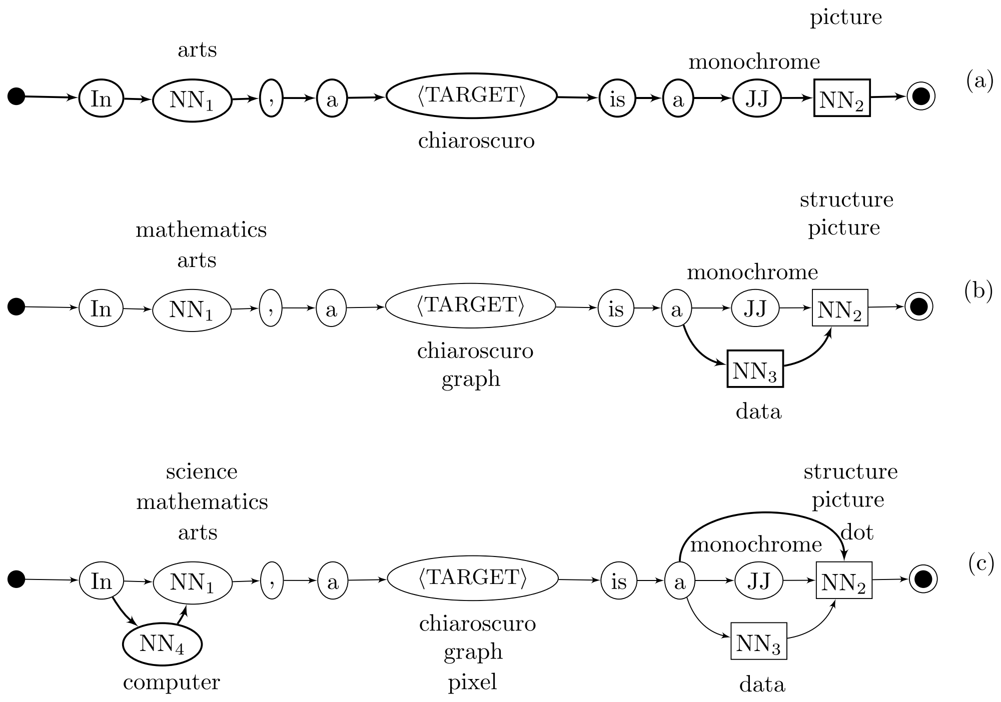

Figure 2
The Word-Class Lattice construction steps on the first three sentences in Table 4. We show in bold the nodes and edges added to the lattice graph as a result of each sentence alignment step. The support of each word class is reported beside the corresponding node.

edges are added that connect node "In" to NN_4 and NN_4 to NN_1. Figure 2c shows the resulting lattice.

Variants of the WCL model. So far we have assumed that our WCL model learns lattices from the training sentences in their entirety (we call this model WCL-1). We also considered a second model that, given a star pattern, learns three separate WCLs, one for each of the three main fields of the definition, namely: definiendum (DF), definitor (VF), and definiens (GF). We refer to this latter model as WCL-3. Note that our model does not take into account the REST field, so this fragment of the training sentences is discarded. The reason for introducing the WCL-3 model is that, whereas definitional patterns are highly variable, DF, VF, and GF individually exhibit a lower variability, thus WCL-3 improves the generalization power.

Once the learning process is over, a set of WCLs is produced. Given a test sentence s, the classification phase for the WCL-1 model consists of determining whether there exists a lattice that matches s. In the case of WCL-3, we consider any combination of definiendum, definitor, and definiens lattices. Given that different combinations might match, for each combination of three WCLs we calculate a confidence score as follows:

$$score(s, l_{\mathrm{DF}}, l_{\mathrm{VF}}, l_{\mathrm{GF}}) = coverage \cdot log_2(support + 1) \quad (2)$$

where s is the candidate sentence, l_{DF}, l_{VF}, and l_{GF} are three lattices (one for each definition field), *coverage* is the fraction of sentence tokens covered by the third lattice, and *support* is the total number of sentences in the corresponding star pattern.

WCL-3 selects, if any, the combination of the three WCLs that best fits the sentence in terms of coverage and support from the training set. In fact, choosing the most appropriate combination of lattices impacts the performance of hypernym extraction. Given its higher performance (Navigli and Velardi 2010), in OntoLearn Reloaded we use WCL-3 for definition classification and hypernym extraction.

3.3 Domain Filtering and Creation of the Hypernym Graph

The WCLs described in the previous section are used to identify definitional sentences and harvest hypernyms for the terms obtained as a result of the terminology extraction phase. In this section we describe how to filter out non-domain definitions and create a dense hypernym graph for the domain of interest.

Given a term t, the common case is that several definitions are found for it (e.g., the *flow network* example provided at the beginning of this section). Many of these will not pertain to the domain of interest, however, especially if they are obtained from the Web or if they define ambiguous terms. For instance, in the COMPUTER SCIENCE domain, the *cash flow* definition of *flow network* shown in Table 3 was not pertinent. To discard these non-domain sentences, we weight each definition candidate $d(t)$ according to the domain terms that are contained therein using the following formula:

$$DomainWeight(d(t)) = \frac{|B_{d(t)} \cap D|}{|B_{d(t)}|} \tag{3}$$

where $B_{d(t)}$ is the bag of content words in the definition candidate $d(t)$ and D is given by the union of the initial terminology $T^{(0)}$ and the set of single words of the terms in $T^{(0)}$ that can be found as nouns in WordNet. For example, given $T^{(0)} = \{$ *greedy algorithm*, *information retrieval*, *minimum spanning tree* $\}$, our domain terminology $D = T^{(0)} \cup \{$ *algorithm*, *information*, *retrieval*, *tree* $\}$. According to Equation (3), the domain weight of a definition is normalized by the total number of content words in the definition, so as to penalize longer definitions. Domain filtering is performed by keeping only those definitions $d(t)$ whose $DomainWeight(d(t)) \geq \theta$, where θ is an empirically tuned threshold.[9] In Table 3 (third column), we show some values calculated for the corresponding definitions (the fourth column reports a check mark ✓if the domain weight is above the threshold, an $\times$ otherwise). Domain filtering performs some implicit form of Word Sense Disambiguation (Navigli 2009), as it aims at discarding senses of hypernyms which do not pertain to the domain.

Let H_t be the set of hypernyms extracted with WCLs from the definitions of term t which survived this filtering phase. For each $t \in T^{(i)}$, we add H_t to our graph $G_{noisy} = (V_{noisy}, E_{noisy})$, that is, we set $V_{noisy} := V_{noisy} \cup H_t$. For each t, we also add a directed edge (h, t)[10] for each hypernym $h \in H_t$, that is, we set $E_{noisy} := E_{noisy} \cup \{(h, t)\}$. As a result

9 Empirically set to 0.38, as a result of tuning on several data sets of manually annotated definitions in different domains.

10 In what follows, (h, t) or $h \rightarrow t$ reads "t is-a h."

of this step, the graph contains our domain terms and their hypernyms obtained from domain-filtered definitions. We now set:

$$T^{(i+1)} := \bigcup_{t \in T^{(i)}} H_t \setminus \bigcup_{j=1}^{i} T^{(j)} \tag{4}$$

that is, the new set of terms $T^{(i+1)}$ is given by the hypernyms of the current set of terms $T^{(i)}$ excluding those terms that were already processed during previous iterations of the algorithm. Next, we move to iteration $i+1$ and repeat the last two steps, namely, we perform definition/hypernym extraction and domain filtering on $T^{(i+1)}$. As a result of subsequent iterations, the initially empty graph is increasingly populated with new nodes (i.e., domain terms) and edges (i.e., hypernymy relations).

After a given number of iterations K, we obtain a dense hypernym graph G_{noisy} that potentially contains more than one connected component. Finally, we connect all the upper term nodes in G_{noisy} to a single top node $\top$. As a result of this connecting step, only one connected component of the noisy hypernym graph—which we call the **backbone component**—will contain an upper taxonomy consisting of upper terms in U.

The resulting graph G_{noisy} potentially contains cycles and multiple hypernyms for the vast majority of nodes. In order to eliminate noise and obtain a full-fledged taxonomy, we perform a step of graph pruning, as described in the next section.

3.4 Graph Pruning

At the end of the iterative hypernym harvesting phase, described in Sections 3.2 and 3.3, the result is a highly dense, potentially disconnected, hypernymy graph (see Section 4 for statistics concerning the experiments that we performed). Wrong nodes and edges might stem from errors in any of the definition/hypernym extraction and domain filtering steps. Furthermore, for each node, multiple "good" hypernyms can be harvested. Rather than using heuristic rules, we devised a novel graph pruning algorithm, based on the Chu-Liu/Edmonds optimal branching algorithm (Chu and Liu 1965; Edmonds 1967), that exploits the topological graph properties to produce a full-fledged taxonomy. The algorithm consists of four phases (i.e., graph trimming, edge weighting, optimal branching, and pruning recovery) that we describe hereafter with the help of the noisy graph in Figure 3a, whose grey nodes belong to the initial terminology $T^{(0)}$ and whose bold node is the only upper term.

3.4.1 Graph Trimming. We first perform two trimming steps. First, we disconnect "false" roots, i.e., nodes which are not in the set of upper terms and with no incoming edges (e.g., *image* in Figure 3a). Second, we disconnect "false" leaves, namely, leaf nodes which are not in the initial terminology and with no outgoing edges (e.g., *output* in Figure 3a). We show the disconnected components in Figure 3b.

3.4.2 Edge Weighting. Next, we weight the edges in our noisy graph G_{noisy}. A policy based only on graph connectivity (e.g., in-degree or betweenness, see Newman [2010] for a complete survey) is not sufficient for taxonomy learning.[11] Consider again the graph in

11 As also remarked by Kozareva and Hovy (2010), who experimented with in-degree.

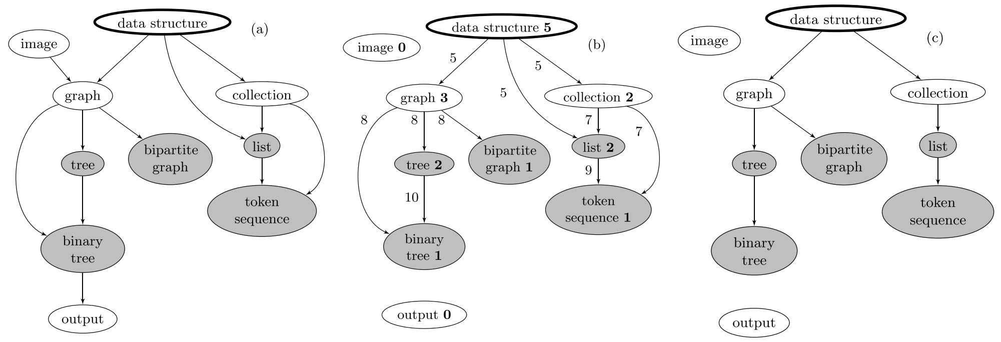

Figure 3
A noisy graph excerpt (a), its trimmed version (b), and the final taxonomy resulting from pruning (c).

Figure 3: In choosing the best hypernym for the term *token sequence*, a connectivity-based measure might select *collection* rather than *list*, because the former reaches more nodes. In taxonomy learning, however, longer hypernymy paths should be preferred (e.g., *data structure* $\rightarrow$ *collection* $\rightarrow$ *list* $\rightarrow$ *token sequence* is better than *data structure* $\rightarrow$ *collection* $\rightarrow$ *token sequence*).

We thus developed a novel weighting policy aimed at finding the best trade-off between path length and the connectivity of traversed nodes. It consists of three steps:

i) Weight each node v by the number of nodes belonging to the initial terminology that can be reached from v (potentially including v itself).[12] Let $w(v)$ denote the weight of v (e.g., in Figure 3b, node *collection* reaches *list* and *token sequence*, thus $w(\textit{collection}) = 2$, whereas $w(\textit{graph}) = 3$). All weights are shown in the corresponding nodes in Figure 3b.

ii) For each node v, consider all the paths from an upper root r to v. Let $\Gamma(r, v)$ be the set of such paths. Each path $p \in \Gamma(r, v)$ is weighted by the cumulative weight of the nodes in the path, namely:

$$\omega(p) = \sum_{v' \in p} w(v') \tag{5}$$

iii) Assign the following weight to each incoming edge (h, v) of v (i.e., h is one of the direct hypernyms of v):

$$w(h, v) = \max_{r \in U} \max_{p \in \Gamma(r,h)} \omega(p) \tag{6}$$

This formula assigns to edge (h, v) the value $\omega(p)$ of the highest-weighting path p from h to any upper root $\in U$. For example, in Figure 3b, $w(\textit{list}) = 2$, $w(\textit{collection}) = 2$, $w(\textit{data structure}) = 5$. Therefore, the set of paths Γ(*data structure*, *list*) = { *data structure* $\rightarrow$ *list*, *data structure* $\rightarrow$ *collection* $\rightarrow$ *list* }, whose weights are 7 (w(*data structure*) + w(*list*)) and 9 (w(*data structure*) + w(*collection*) + w(*list*)), respectively. Hence, according to Formula 6, w(*list*, *token sequence*) = 9. We show all edge weights in Figure 3b.

12 Nodes in a cycle are visited only once.

3.4.3 Optimal Branching. Next, our goal is to move from a noisy graph to a tree-like taxonomy on the basis of our edge weighting strategy. A maximum spanning tree algorithm cannot be applied, however, because our graph is directed. Instead, we need to find an optimal branching, that is, a rooted tree with an orientation such that every node but the root has in-degree 1, and whose overall weight is maximum. To this end, we first apply a pre-processing step: For each (weakly) connected component in the noisy graph, we consider a number of cases, aimed at identifying a single "reasonable" root node to enable the optimal branching to be calculated. Let R be the set of candidate roots, that is, nodes with no incoming edges. We perform the following steps:

i) If $|R| = 1$ then we select the only candidate as root.

ii) Else if $|R| > 1$, if an upper term is in R, we select it as root, else we choose the root $r \in R$ with the highest weight w according to the weighting strategy described in Section 3.4.2. We also disconnect all the unselected roots, that is, those in $R \setminus \{r\}$.

iii) Else (i.e., if $|R| = 0$), we proceed as for step (ii), but we search candidates within the entire connected component and select the highest weighting node. In contrast to step (ii), we remove all the edges incoming to the selected node.

This procedure guarantees not only the selection but also the existence of a single root node for each component, from which the optimal branching algorithm can start. We then apply the Chu-Liu/Edmonds algorithm (Chu and Liu 1965; Edmonds 1967) to each component $G_i = (V_i, E_i)$ of our directed weighted graph G_{noisy} in order to find an optimal branching. The algorithm consists of two phases: a **contraction** phase and an **expansion** phase. The contraction phase is as follows:

1. For each node which is not a root, we select the entering edge with the highest weight. Let S be the set of such $|V_i| - 1$ edges;
2. If no cycles are formed in S, go to the expansion phase. Otherwise, continue;
3. Given a cycle in S, contract the nodes in the cycle into a pseudo-node k, and modify the weight of each edge entering any node v in the cycle from some node h outside the cycle, according to the following equation:

   $$w(h,k) = w(h,v) + (w(x(v),v) - min_v(w(x(v),v))) \tag{7}$$

   where $x(v)$ is the predecessor of v in the cycle and $w(x(v),v)$ is the weight of the edge in the cycle which enters v;
4. Select the edge entering the cycle which has the highest modified weight and replace the edge which enters the same real node in S by the new selected edge;
5. Go to step 2 with the contracted graph.

The expansion phase is applied if pseudo-nodes have been created during step 3. Otherwise, this phase is skipped and $T_i = (V_i, S)$ is the optimal branching of component

G_i (i.e., the i-th component of G_{noisy}). During the expansion phase, pseudo-nodes are replaced with the original cycles. To break the cycle, we select the real node v into which the edge selected in step 4 enters, and remove the edge entering v belonging to the cycle. Finally, the weights on the edges are restored. For example, consider the cycle in Figure 4a. Nodes *pagerank*, *map*, and *rank* are contracted into a pseudo-node, and the edges entering the cycle from outside are re-weighted according to Equation (7). According to the modified weights (Figure 4b), the selected edge, that is, $(table, map)$, is the one with weight $w = 13$. During the expansion phase, the edge $(pagerank, map)$ is eliminated, thus breaking the cycle (Figure 4c).

The tree-like taxonomy resulting from the application of the Chu-Liu/Edmonds algorithm to our example in Figure 3b is shown in Figure 3c.

3.4.4 Pruning Recovery. The weighted directed graph G_{noisy} input to the Chu-Liu/ Edmonds algorithm might contain many (weakly) connected components. In this case, an optimal branching is found for each component, resulting in a forest of taxonomy trees. Although some of these components are actually noisy, others provide an important contribution to the final tree-like taxonomy. The objective of this phase is to recover from excessive pruning, and re-attach some of the components that were disconnected during the optimal branching step. Recall from Section 3.3 that, by construction, we have only one backbone component, that is, a component which includes an upper taxonomy. Our aim is thus to re-attach meaningful components to the backbone taxonomy. To this end, we apply Algorithm 1. The algorithm iteratively merges non-backbone trees to the backbone taxonomy tree T_0 in three main steps:

- **Semantic reconnection step** (lines 7–9 in Algorithm 1): In this step we reuse a previously removed "noisy" edge, if one is available, to reattach a non-backbone component to the backbone. Given a root node r_{T_i} of a non-backbone tree T_i ($i > 0$), if an edge (v, r_{T_i}) existed in the noisy graph G_{noisy} (i.e., the one obtained before the optimal branching phase), with $v \in T_0$, then we connect the entire tree T_i to T_0 by means of this edge.

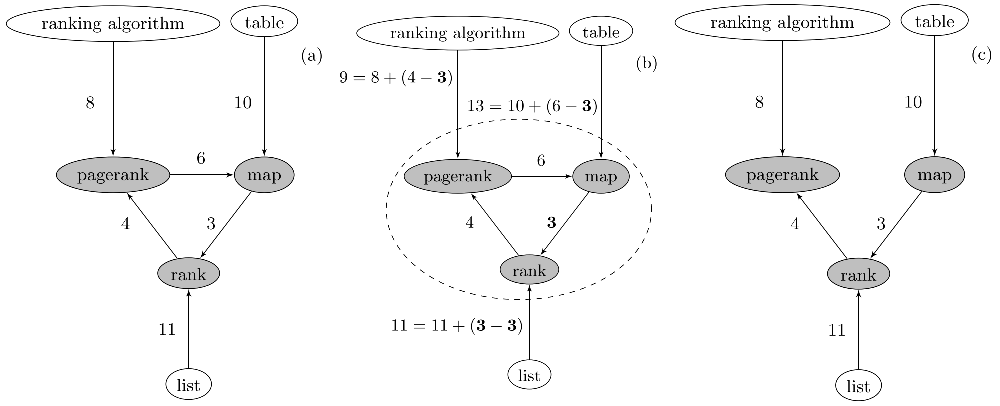

Figure 4
A graph excerpt containing a cycle (a); Edmonds' contraction phase: a pseudo-node enclosing the cycle with updated weights on incoming edges (b); and Edmonds' expansion phase: the cycle is broken and weights are restored (c).

Algorithm 1 *PruningRecovery*(G, G_{noisy})

```
Require: $G$ is a forest
1: repeat
2:    Let $F := \{T_0, T_1, \ldots, T_{|F|}\}$ be the forest of trees in $G = (V, E)$
3:    Let $T_0 \in F$ be the backbone taxonomy
4:    $E' \leftarrow E$
5:    for all $T$ in $F \setminus \{T_0\}$ do
6:       $r_T \leftarrow rootOf(T)$
7:       if $\exists v \in T_0$ s.t. $(v, r_T) \in G_{noisy}$ then
8:          $E \leftarrow E \cup \{(v, r_T)\}$
9:          break
10:      else
11:         if out-degree$(r_T) = 0$ then
12:            if $\exists v \in T_0$ s.t. $v$ is the longest right substring of $r_T$ then
13:               $E := E \cup \{(v, r_T)\}$
14:               break
15:         else
16:            $E \leftarrow E \setminus \{(r_T, v) : v \in V\}$
17:            break
18: until $E = E'$
```

- **Reconnection step by lexical inclusion** (lines 11–14): Otherwise, if T_i is a singleton (the out-degree of r_{T_i} is 0) and there exists a node $v \in T_0$ such that v is the longest right substring of r_{T_i} by lexical inclusion,[13] we connect T_i to the backbone tree T_0 by means of the edge (v, r_{T_i}).
- **Decomposition step** (lines 15–17): Otherwise, if the component T_i is not a singleton (i.e., if the out-degree of the root node r_{T_i} is > 0) we disconnect r_{T_i} from T_i. At first glance, it might seem counterintuitive to remove edges during pruning recovery. Reconnecting by lexical inclusion within a domain has already been shown to perform well in the literature (Vossen 2001; Navigli and Velardi 2004), but we want to prevent any cascading errors on the descendants of the root node, and at the same time free up other pre-existing "noisy" edges incident to the descendants.

These three steps are iterated on the newly created components, until no change is made to the graph (line 18). As a result of our pruning recovery phase we return the enriched backbone taxonomy. We show in Figure 5 an example of pruning recovery that starts from a forest of three components (including the backbone taxonomy tree on top, Figure 5a). The application of the algorithm leads to the disconnection of a tree root, that is, *ordered structure* (Figure 5a, lines 15–17 of Algorithm 1), the linking of the trees rooted at *token list* and *binary search tree* to nodes in the backbone taxonomy (Figures 5b and 5d, lines 7–9), and the linking of *balanced binary tree* to *binary tree* thanks to lexical inclusion (Figure 5c, lines 11–14 of the algorithm).

13 Similarly to our original OntoLearn approach (Navigli and Velardi 2004), we define a node's string $v = w_n w_{n-1} \ldots w_2 w_1$ to be lexically included in that of a node $v' = w'_m w'_{m-1} \ldots w'_2 w'_1$ if $m > n$ and $w_j = w'_j$ for each $j \in \{1, \ldots, n\}$.

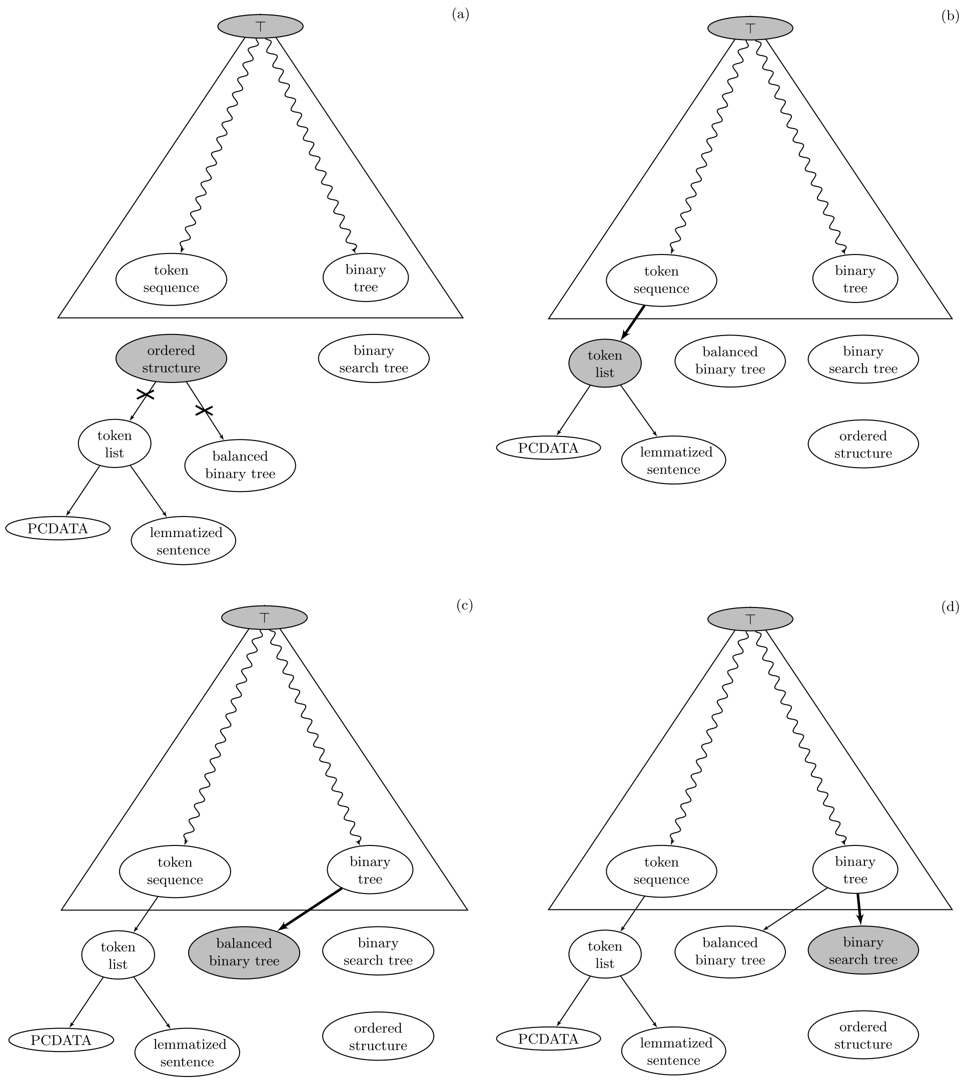

Figure 5
An example starting with three components, including the backbone taxonomy tree on the top and two other trees on the bottom (a). As a result of pruning recovery, we disconnect *ordered structure* (a); we connect *token sequence* to *token list* by means of a "noisy" edge (b); we connect *binary tree* to *balanced binary tree* by lexical inclusion (c); and finally *binary tree* to *binary search tree* by means of another "noisy" edge (d).

3.5 Edge Recovery

The goal of the last phase was to recover from the excessive pruning of the optimal branching phase. Another issue of optimal branching is that we obtain a tree-like taxonomic structure, namely, one in which each node has only one hypernym. This is not fully appropriate in taxonomy learning, because systematic ambiguity and polysemy often require a concept to be paradigmatically related to more than one hypernym. In fact, a more appropriate structure for a conceptual hierarchy is a DAG, as in WordNet. For example, two equally valid hypernyms for *backpropagation* are *gradient descent search* *procedure* and *training algorithm*, so two hypernym edges should correctly be incident to the *backpropagation* node.

We start from our backbone taxonomy T_0 obtained after the pruning recovery phase described in Section 3.4.4. In order to obtain a DAG-like taxonomy we apply the following step: for each "noisy" edge $(v, v') \in E_{noisy}$ such that v, v' are nodes in T_0 but the edge (v, v') does not belong to the tree, we add (v, v') to T_0 if:

i) it does not create a cycle in T_0;

ii) the absolute difference between the length of the shortest path from v to the root r_{T_0} and that of the shortest path from v' to r_{T_0} is within an interval $[m, M]$. The aim of this constraint is to maintain a balance between the height of a concept in the tree-like taxonomy and that of the hypernym considered for addition. In other words, we want to avoid the connection of an overly abstract concept with an overly specific one.

In Section 4, we experiment with three versions of our OntoLearn Reloaded algorithm, namely: one version that does not perform edge recovery (i.e., which learns a tree-like taxonomy [TREE], and two versions that apply edge recovery (i.e., which learn a DAG) with different intervals for constraint (ii) above ($DAG[1, 3]$ and $DAG[0, 99]$; note that the latter version virtually removes constraint (ii)). Examples of recovered edges will be presented and discussed in the evaluation section.

3.6 Complexity

We now perform a complexity analysis of the main steps of OntoLearn Reloaded. Given the large number of steps and variables involved we provide a separate discussion of the main costs for each individual step, and we omit details about commonly used data structures for access and storage, unless otherwise specified. Let $G_{noisy} = (V_{noisy}, E_{noisy})$ be our noisy graph, and let $n = |V_{noisy}|$ and $m = |E_{noisy}|$.

1. **Terminology extraction:** Assuming a part-of-speech tagged corpus as input, the cost of extracting candidate terms by scanning the corpus with a maximum-size window is in the order of the word size of the input corpus. Thus, the application of statistical measures to our set of candidate terms has a computational cost that is on the order of the square of the number of term candidates (i.e., the cost of calculating statistics for each pair of terms).

2. **Definition and hypernym extraction:** In the second step, we first retrieve candidate definitions from the input corpus, which costs on the order of the corpus size.[14] Each application of a WCL classifier to an input candidate sentence s containing a term t costs on the order of the word length of the sentence, and we have a constant number of such classifiers. So the cost of this step is given by the sum of the lengths of the candidate sentences in the corpus, which is lower than the word size of the corpus.

14 Note that this corpus consists of both free text and Web glossaries (cf. Section 3.2).

3. **Domain filtering and creation of the graph:** The cost of domain filtering for a single definition is in the order of its word length, so the running time of domain filtering is in the order of the sum of the word size of the acquired definitions. As for the hypernym graph creation, using an adjacency-list representation of the graph G_{noisy}, the dynamic addition of a newly acquired hypernymy edge costs $O(n)$, an operation which has to be repeated for each (hypernymy, term) pair.
4. **Graph pruning**, consisting of the following steps:
   - **Graph trimming:** This step requires $O(n)$ time in order to identify false leaves and false roots by iterating over the entire set of nodes.
   - **Edge weighting:** i) We perform a DFS ($O(n+m)$) to weight all the nodes in the graph; ii) we collect all paths from upper roots to any given node, totalizing $O(n!)$ paths in the worst case (i.e., in a complete graph). In real domains, however, the computational cost of this step will be much lower. In fact, over our six domains, the average number of paths per node ranges from 4.3 ($n = 2107$, ANIMALS) to 3175.1 ($n = 2616$, FINANCE domain): In the latter, worst case, in practice, the number of paths is in the order of n, thus the cost of this step, performed for each node, can be estimated by $O(n^2)$ running time; iii) assigning maximum weights to edges costs $O(m)$ if in the previous step we keep track of the maximum value of paths ending in each node h (see Equation (6)).
   - **Optimal branching:** Identifying the connected components of our graph costs $O(n+m)$ time, identifying root candidates and selecting one root per component costs $O(n)$, and finally applying the Chu-Liu/Edmonds algorithm costs $O(m \cdot log_2 n)$ for sparse graphs, $O(n^2)$ for dense ones, using Tarjan's implementation (Tarjan 1977).
5. **Pruning recovery:** In the worst case, m iterations of Algorithm 1 will be performed, each costing $O(n)$ time, thus having a total worst-case cost of $O(mn)$.
6. **Edge recovery:** For each pair of nodes in T_0 we perform i) the identification of cycles ($O(n+m)$) and ii) the calculation of the shortest paths to the root ($O(n+m)$). By precomputing the shortest path for each node, the cost of this step is $O(n(n+m))$ time.

Therefore, in practice, the computational complexity of OntoLearn Reloaded is polynomial in the main variables of the problem, namely, the number of words in the corpus and nodes in the noisy graph.

4. Evaluation

Ontology evaluation is a hard task that is difficult even for humans, mainly because there is no unique way of modeling the domain of interest. Indeed several different taxonomies might model a particular domain of interest equally well. Despite this difficulty, various evaluation methods have been proposed in the literature for assessing the quality of a taxonomy. These include Brank, Mladenic, and Grobelnik (2006) and Maedche, Pekar, and Staab (2002):

a) automatic evaluation against a gold standard;

b) manual evaluation performed by domain experts;

c) structural evaluation of the taxonomy;

d) application-driven evaluation, in which a taxonomy is assessed on the basis of the improvement its use generates within an application.

Other quality indicators have been analyzed in the literature, such as accuracy, completeness, consistency (Völker et al. 2008), and more theoretical features (Guarino and Welty 2002) like essentiality, rigidity, and unity. Methods (a) and (b) are by far the most popular ones. In this section, we will discuss in some detail the pros and cons of these two approaches.

Gold standard evaluation. The most popular approach for the evaluation of lexicalized taxonomies (adopted, e.g., in Snow, Jurafsky, and Ng 2006; Yang and Callan 2009; and Kozareva and Hovy 2010) is to attempt to reconstruct an existing gold standard (Maedche, Pekar, and Staab 2002), such as WordNet or the Open Directory Project. This method is applicable when the set of taxonomy concepts are given, and the evaluation task is restricted to measuring the ability to reproduce hypernymy links between concept pairs. The evaluation is far more complex when learning a specialized taxonomy entirely from scratch, that is, when both terms and relations are unknown. In reference taxonomies, even in the same domain, the granularity and cotopy[15] of an abstract concept might vary according to the scope of the taxonomy and the expertise of the team who created it (Maedche, Pekar, and Staab 2002). For example, both the terms *chiaroscuro* and *collage* are classified under *picture, image, icon* in WordNet, but in the Art & Architecture Thesaurus (AA&T)[16] *chiaroscuro* is categorized under *perspective and shading techniques* whereas *collage* is classified under *image-making processes and techniques*. As long as common-sense, non-specialist knowledge is considered, it is still feasible for an automated system to replicate an existing classification, because the Web will provide abundant evidence for it. For example, Kozareva and Hovy (2010, K&H hereafter) are very successful at reproducing the WordNet sub-taxonomy for ANIMALS, because dozens of definitional patterns are found on the Web that classify, for example, *lion* as a *carnivorous feline mammal*, or *carnivorous*, or *feline*. As we show later in this section, however, and as also suggested by the previous AA&T example, finding hypernymy patterns in more specialized domains is far more complex. Even in simpler domains, however, it is not clear how to evaluate the concepts and relations not found in the reference taxonomy. Concerning this issue, Zornitsa Kozareva comments that: "When we gave sets of terms to annotators and asked them to produce a taxonomy, people struggled with the domain terminology and produced quite messy organization. Therefore, we decided to go with WordNet and use it as a gold truth" (personal communication). Accordingly, K&H do not provide an evaluation of the nodes and relations other than those for which the ground truth is known. This is further clarified in a personal communication: "Currently we do not have a full list of all is-a outside

15 The cotopy of a concept is the set of its hypernyms and hyponyms.
16 `http://www.getty.edu/vow/AATHierarchy`.

WordNet. [...] In the experiments, we work only with the terms present in WordNet [...] The evaluation is based only on the WordNet relations. However, the harvesting algorithm extracts much more. Currently, we do not know how to evaluate the Web taxonomization."

To conclude, gold standard evaluation has some evident drawbacks:

- When both concepts and relations are unknown, it is almost impossible to replicate a reference taxonomy accurately.
- In principle, concepts not in the reference taxonomy can be either wrong or correct; therefore the evaluation is in any case incomplete.

Another issue in gold standard evaluation is the definition of an adequate evaluation metric. The most common measure used in the literature to compare a learned with a gold-standard taxonomy is the *overlapping factor* (Maedche, Pekar, and Staab 2002). Given the set of *is-a* relations in the two taxonomies, the overlapping factor simply computes the ratio between the intersection and union of these sets. Therefore the overlapping factor gives a useful global measure of the similarity between the two taxonomies. It provides no structural comparison, however: Errors or differences in grouping concepts in progressively more general classes are not evidenced by this measure.

Comparison against a gold standard has been analyzed in a more systematic way by Zavitsanos, Paliouras, and Vouros (2011) and Brank, Mladenic, and Grobelnik (2006). They propose two different strategies for escaping the "naming" problem that we have outlined. Zavitsanos, Paliouras, and Vouros (2011) propose transforming the ontology concepts and their properties into distributions over the term space of the source data from which the ontology has been learned. These distributions are used to compute pairwise concept similarity between gold standard and learned ontologies.

Brank, Mladenic, and Grobelnik (2006) exploit the analogy between ontology learning and unsupervised clustering, and propose OntoRand, a modified version of the Rand Index (Rand 1971) for computing the similarity between ontologies. Morey and Agresti (1984) and Carpineto and Romano (2012), however, demonstrated a high dependency of the Rand Index (and consequently of OntoRand itself) upon the number of clusters, and Fowlkes and Mallows (1983) show that the Rand Index has the undesirable property of converging to 1 as the number of clusters increases, even in the unrealistic case of independent clusterings. These undesired outcomes have also been experienced by Brank, Mladenic, and Grobelnik (2006, page 5), who note that "the similarity of an ontology to the original one is still as high as 0.74 even if only the top three levels of the ontology have been kept." Another problem with the OntoRand formula, as also remarked in Zavitsanos, Paliouras, and Vouros (2011), is the requirement of comparing ontologies with the same set of instances.

Manual evaluation. Comparison against a gold standard is important because it represents a sort of objective evaluation of an automated taxonomy learning method. As we have already remarked, however, learning an existing taxonomy is not particularly interesting in itself. Taxonomies are mostly needed in novel, often highly technical domains for which there are no gold standards. For a system to claim to be able to acquire a taxonomy from the ground up, manual evaluation seems indispensable. Nevertheless, none of the taxonomy learning systems surveyed in Section 2 performs such evaluation. Furthermore, manual evaluation should not be limited to an assessment of the acquired hypernymy relations "in isolation," but must also provide a structural assessment aimed at identifying common phenomena and the overall quality of the taxonomic structure. Unfortunately, as already pointed out, manual evaluation is a hard task. Deciding whether or not a concept belongs to a given domain is more or less feasible for a domain expert, but assessing the quality of a hypernymy link is far more complex. On the other hand, asking a team of experts to blindly reconstruct a hierarchy, given a set of terms, may result in the "messy organization" reported by Zornitsa Kozareva. In contrast to previous approaches to taxonomy induction, OntoLearn Reloaded provides a natural solution to this problem, because *is-a* links in the taxonomy are supported by one or more definition sentences from which the hypernymy relation was extracted. As shown later in this section, definitions proved to be a very helpful feature in supporting manual analysis, both for hypernym evaluation and structural assessment.

The rest of this section is organized as follows. We first describe the experimental set-up (Section 4.1): OntoLearn Reloaded is applied to the task of acquiring six taxonomies, four of which attempt to replicate already existing gold standard sub-hierarchies in WordNet[17] and in the MeSH medical ontology,[18] and the other two are new taxonomies acquired from scratch. Next, we present a large-scale multi-faceted evaluation of OntoLearn Reloaded focused on three of the previously described evaluation methods, namely: comparison against a gold standard, manual evaluation, and structural evaluation. In Section 4.2 we introduce a novel measure for comparing an induced taxonomy against a gold standard one. Finally, Section 4.3 is dedicated to a manual evaluation of the six taxonomies.

4.1 Experimental Set-up

We now provide details on the set-up of our experiments.

4.1.1 Domains. We applied OntoLearn Reloaded to the task of acquiring six taxonomies: ANIMALS, VEHICLES, PLANTS, VIRUSES, ARTIFICIAL INTELLIGENCE, and FINANCE. The first four taxonomies were used for comparison against three WordNet sub-hierarchies and the viruses sub-hierarchy of MeSH. The ANIMALS, VEHICLES, and PLANTS domains were selected to allow for comparison with K&H, who experimented on the same domains. The ARTIFICIAL INTELLIGENCE and FINANCE domains are examples of taxonomies truly built from the ground up, for which we provide a thorough manual evaluation. These domains were selected because they are large, interdisciplinary, and continuously evolving fields, thus representing complex and specialized use cases.

4.1.2 Definition Harvesting. For each domain, definitions were sought in Wikipedia and in Web glossaries automatically obtained by means of a Web glossary extraction system (Velardi, Navigli, and D'Amadio 2008). For the ARTIFICIAL INTELLIGENCE domain we also used a collection consisting of the entire IJCAI proceedings from 1969 to 2011 and the ACL archive from 1979 to 2010. In what follows we refer to this collection as the "AI corpus." For FINANCE we used a combined corpus from the freely available collection of *Journal of Financial Economics* from 1995 to 2012 and from *Review Of Finance* from 1997 to 2012 for a total of 1,575 papers.

17 `http://wordnet.princeton.edu`.
18 `http://www.nlm.nih.gov/mesh/`.

4.1.3 Terminology. For the ANIMALS, VEHICLES, PLANTS, and VIRUSES domains, the initial terminology was a fragment of the nodes of the reference taxonomies,[19] similarly to, and to provide a fair comparison with, K&H. For the AI domain instead, the initial terminology was selected using our TermExtractor tool[20] on the AI corpus. TermExtractor extracted over 5,000 terms from the AI corpus, ranked according to a combination of relevance indicators related to the (direct) document frequency, domain pertinence, lexical cohesion, and other indicators (Sclano and Velardi 2007). We manually selected 2,218 terms from the initial set, with the aim of eliminating compounds like *order of magnitude*, *empirical study*, *international journal*, that are frequent but not domain relevant. For similar reasons a manual selection of terms was also applied to the terminology automatically extracted for the FINANCE domain, obtaining 2,348 terms[21] from those extracted by TermExtractor. An excerpt of extracted terms was provided in Table 1.

4.1.4 Upper Terms. Concerning the selection of upper terms U (cf. Section 3.2), again similarly to K&H, we used just one concept for each of the four domains focused upon: ANIMALS, VEHICLES, PLANTS, and VIRUSES. For the AI and FINANCE domains, which are more general and complex, we selected from WordNet a core taxonomy of 32 upper concepts U (resulting in 52 terms) that we used as a stopping criterion for our iterative definition/hypernym extraction and filtering procedure (cf. Section 3.2). The complete list of upper concepts was given in Table 2. WordNet upper concepts are general enough to fit most domains, and in fact we used the same set U for AI and FINANCE. Nothing, however, would have prevented us from using a domain-specific core ontology, such as the CRM-CIDOC core ontology for the domain of ART AND ARCHITECTURE.[22]

4.1.5 Algorithm Versions and Structural Statistics. For each of the six domains we ran the three versions of our algorithm: without pruning recovery (TREE), with $[1,3]$ recovery ($DAG[1,3]$), and with $[0,99]$ recovery ($DAG[0,99]$), for a total of 18 experiments. We remind the reader that the purpose of the recovery process was to reattach some of the edges deleted during the optimal branching step (cf. Section 3.5).

Figure 6 shows an excerpt of the AI tree-like taxonomy under the node *data structure*. Notice that, even though the taxonomy looks good overall, there are still a few errors, such as "*neuron* is a *neural network*" and overspecializations like "*network* is a *digraph*." Figure 7 shows a sub-hierarchy of the FINANCE tree-like taxonomy under the concept *value*.

In Table 6 we give the structural details of the 18 taxonomies extracted for our six domains. In the table, edge and node compression refers to the number of surviving nodes and edges after the application of optimal branching and recovery steps to the noisy hypernymy graph. To clarify the table, consider the case of VIRUSES, $DAG[1,3]$: we started with 281 initial terms, obtaining a noisy graph with 1,174 nodes and 1,859 edges. These were reduced to 297 nodes (i.e., 1,174–877) and 339 edges (i.e., 1,859–1,520) after pruning and recovery. Out of the 297 surviving nodes, 222 belonged to the initial

19 For ANIMALS, VEHICLES, and PLANTS we used precisely the same seeds as K&H.
20 `http://lcl.uniroma1.it/termextractor`.
21 These dimensions are quite reasonable for large technical domains: as an example, *The Economist*'s glossary of economic terms includes on the order of 500 terms (`http://www.economist.com/economics-a-to-z/`).
22 `http://cidoc.mediahost.org/standard_crm(en)(E1).xml`.

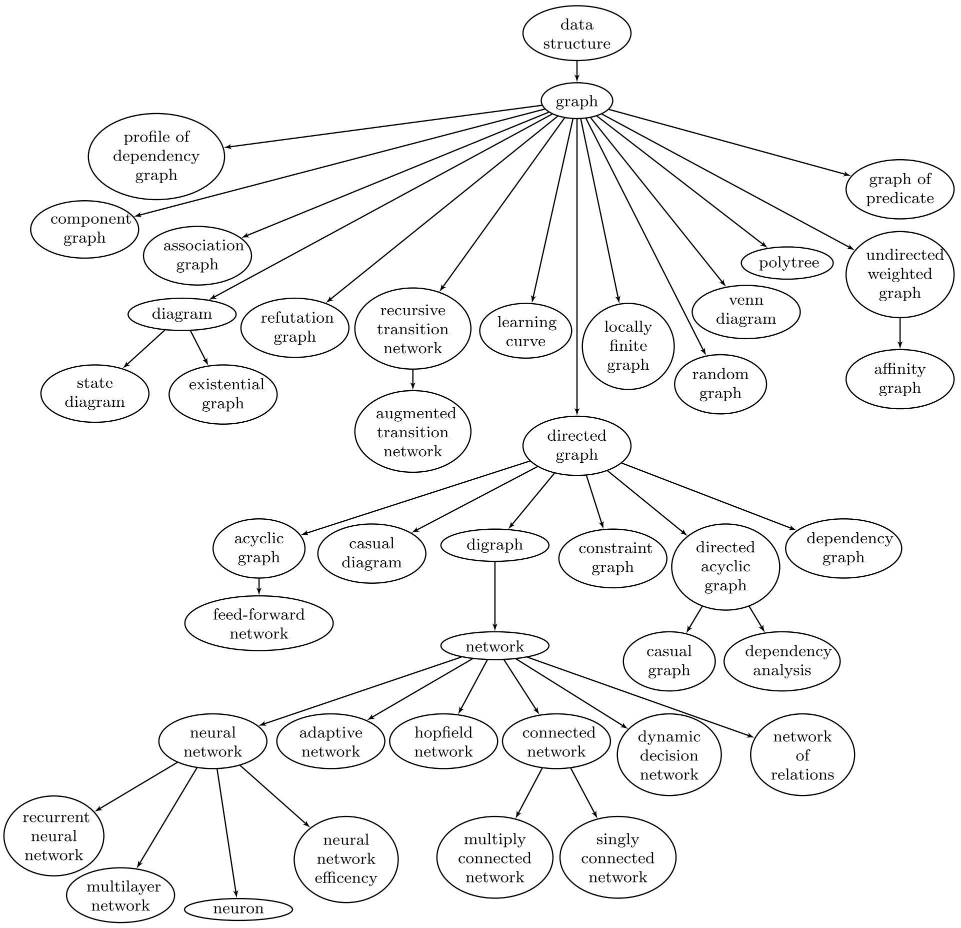

Figure 6
An excerpt of the ARTIFICIAL INTELLIGENCE taxonomy.

terminology; therefore the coverage over the initial terms is 0.79 (222/281). This means that, for some of the initial terms, either no definitions were found, or the definition was rejected in some of the processing steps. The table also shows, as expected, that the term coverage is much higher for "common-sense" domains like ANIMALS, VEHICLES, and PLANTS, is still over 0.75 for VIRUSES and AI, and is a bit lower for FINANCE (0.65). The maximum and average depth of the taxonomies appears to be quite variable, with VIRUSES and FINANCE at the two extremes. Finally, Table 6 reports in the last column the number of glosses (i.e., domain definitional sentences) obtained in each run. We would like to point out that providing textual glosses for the retrieved domain hypernyms is a novel feature that has been lacking in all previous approaches to ontology learning, and which can also provide key support to much-needed manual validation and enrichment of existing semantic networks (Navigli and Ponzetto 2012).

4.2 Evaluation Against a Gold Standard

In this section we propose a novel, general measure for the evaluation of a learned taxonomy against a gold standard. We borrow the Brank, Mladenic, and Grobelnik

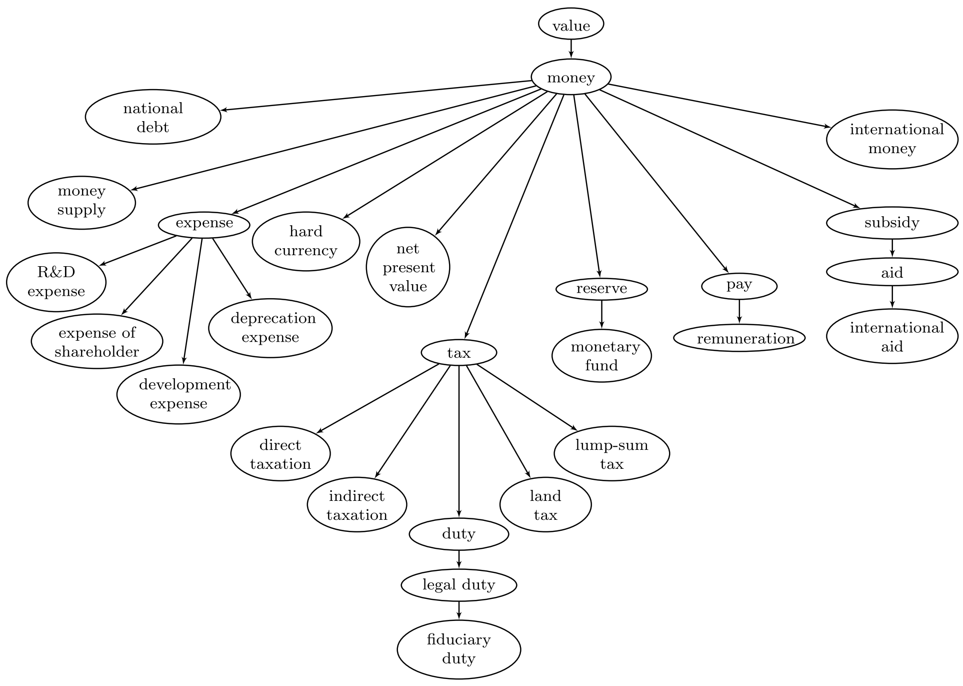

Figure 7
An excerpt of the FINANCE taxonomy.

(2006) idea of exploiting the analogy with unsupervised clustering but, rather than representing the two taxonomies as flat clusterings, we propose a measure that takes into account the hierarchical structure of the two analyzed taxonomies. Under this perspective, a taxonomy can be transformed into a hierarchical clustering by replacing each label of a non-leaf node (e.g., *perspective and shading techniques*) with the transitive closure of its hyponyms (e.g., *cangiatismo, chiaroscuro, foreshortening, hatching*).

4.2.1 Evaluation Model. Techniques for comparing clustering results have been surveyed in Wagner and Wagner (2007), although the only method for comparing hierarchical clusters, to the best of our knowledge, is that proposed by Fowlkes and Mallows (1983). Suppose that we have two hierarchical clusterings H_1 and H_2, with an identical set of n objects. Let k be the maximum depth of both H_1 and H_2, and H_j^i a cut of the hierarchy, where $i \in \{0, \ldots, k\}$ is the cut level and $j \in \{1, 2\}$ selects the clustering of interest. Then, for each cut i, the two hierarchies can be seen as two flat clusterings C_1^i and C_2^i of the n concepts. When $i = 0$ the cut is a single cluster incorporating all the objects, and when $i = k$ we obtain n singleton clusters. Now let:

- n_{11} be the number of object pairs that are in the same cluster in both C_1^i and C_2^i;
- n_{00} be the number of object pairs that are in different clusters in both C_1^i and C_2^i;
- n_{10} be the number of object pairs that are in the same cluster in C_1^i but not in C_2^i;

Table 6
Structural evaluation of three versions of our taxonomy-learning algorithm on six different domains.

Experiment		Term Coverage	Depth	$\|V\|$	$\|E\|$	V Compress.	E Compress.	Glosses
AI	TREE	75.51% (1,675/2,218)	12 max 6.00 avg	2,387	2,386	43.00% (1,801/4,188)	67.31% (4,915/7,301)	1,249
	DAG [1,3]	75.51% (1,675/2,218)	19 max 8.27 avg	2,387	3,554	43.00% (1,801/4,188)	51.32% (3,747/7,301)	2,081
	DAG [0,99]	75.51% (1,675/2,218)	20 max 8.74 avg	2,387	3,994	43.00% (1,801/4,188)	45.29% (3,307/7,301)	2,439
FINANCE	TREE	65.20% (1,533/2,348)	14 max 6.83 avg	2,038	2,037	22.09% (578/2,616)	47.99% (1,880/3,917)	1,064
	DAG [1,3]	65.20% (1,533/2,348)	38 max 18.82 avg	2,038	2,524	22.09% (578/2,616)	35.56% (1,393/3,917)	1,523
	DAG [0,99]	65.20% (1,533/2,348)	65 max 33.54 avg	2,038	2,690	22.09% (578/2,616)	31.32% (1,227/3,917)	1,677
VIRUSES	TREE	79.00% (222/281)	5 max 2.13 avg	297	296	74.70% (877/1,174)	84.07% (1,563/1,859)	172
	DAG [1,3]	79.00% (222/281)	5 max 2.20 avg	297	339	74.70% (877/1,174)	81.76% (1,520/1,859)	212
	DAG [0,99]	79.00% (222/281)	5 max 2.32 avg	297	360	74.70% (877/1,174)	80.63% (1,563/1,859)	233
ANIMALS	TREE	93.56% (640/684)	10 max 4.35 avg	900	899	57.28% (1,207/2,107)	66.96% (1,822/2,721)	724
	DAG [1,3]	93.56% (640/684)	16 max 5.21 avg	900	1,049	57.28% (1,207/2,107)	61.44% (1,672/2,721)	872
	DAG [0,99]	93.56% (640/684)	16 max 5.39 avg	900	1,116	57.28% (1,207/2,107)	58.98% (1,605/2,721)	939
PLANTS	TREE	96.57% (535/554)	19 max 5.85 avg	710	709	72.69% (1,890/2,600)	84.53% (3,877/4,586)	638
	DAG [1,3]	96.57% (535/554)	19 max 6.65 avg	710	922	72.69% (1,890/2,600)	79.89% (3,664/4,586)	851
	DAG [0,99]	96.57% (535/554)	19 max 6.54 avg	710	1,242	72.69% (1,890/2,600)	72.91% (3,344/4,586)	1,171
VEHICLES	TREE	95.72% (112/117)	8 max 3.44 avg	169	168	71.50% (424/593)	80.48% (693/861)	150
	DAG [1,3]	95.72% (112/117)	8 max 3.94 avg	169	200	71.50% (424/593)	76.77% (661/861)	182
	DAG [0,99]	95.72% (112/117)	10 max 4.48 avg	169	231	71.50% (424/593)	73.17% (630/861)	213

- n_{01} be the number of object pairs that are in the same cluster in C_2^i but not in C_1^i;

The generalized Fowlkes and Mallows (F&M) measure of cluster similarity for the cut i ($i \in \{0, \ldots, k\}$), as reformulated by Wagner and Wagner (2007), is defined as:

$$B_{1,2}^i = \frac{n_{11}^i}{\sqrt{(n_{11}^i + n_{10}^i) \cdot (n_{11}^i + n_{01}^i)}}. \tag{8}$$

Note that the formula can be interpreted as the geometric mean of precision and recall of an automated method in clustering the same concept pairs as in a gold-standard clustering. This formula has a few undesirable properties: first, the value of $B^i_{1,2}$ gets close to its maximum 1.0 as we approach the root of the hierarchy ($i = 0$); second, the two hierarchies need to have the same maximum depth k; third, the hierarchies need to have the same number of initial objects and a crisp classification.

In order to apply the F&M measure to the task of comparing a learned and a gold-standard taxonomy, we need to mitigate these problems. Equation (8) copes with the third problem without modifications. In fact, if the sets of objects in H_1 and H_2 are different, the integers n_{10} and n_{01} can be considered as also including objects that belong to one hierarchy and not to the other. In this case, the value of $B^0_{1,2}$ will provide a measure of the overlapping objects in the learned taxonomy and the gold standard one. In order to take into account multiple (rather than crisp) classifications, again, there is no need to change the formula, which is still meaningful if an object is allowed to belong to more than one cluster. As before, mismatches between H_1 and H_2 would result in higher values of n_{10} and n_{01} and lower $B^i_{1,2}$.

A more serious problem with Equation (8) is that the lower the value of i, the higher the value of the formula, whereas, ideally, we would like to reward similar clusterings when the clustering task is more difficult and fine-grained, that is, for cuts that are close to the leaf nodes. To assign a reward to "early" similarity values, we weight the values of $B^i_{1,2}$ with a coefficient $\frac{i+1}{k}$. We can then compute a cumulative measure of similarity with the following formula:

$$B_{1,2} = \frac{\sum_{i=0}^{k-1} \frac{i+1}{k} B^i_{1,2}}{\sum_{i=0}^{k-1} \frac{i+1}{k}} = \frac{\sum_{i=0}^{k-1} \frac{i+1}{k} B^i_{1,2}}{\frac{k+1}{2}}. \tag{9}$$

Finally, to solve the problem of different depths of the two hierarchies, we define a policy that penalizes a learned taxonomy that is less structured than the gold standard one, and rewards—or at least does not penalize—the opposite case.

As an example, consider Figure 8, which shows two taxonomies H_1 and H_2, with non-identical sets of objects $\{a, b, c, d, e, f\}$ and $\{a, b, c, d, e, g\}$. In the figure each edge is labeled by its distance from the root node (the value i in the F&M formula). Notice that H_1 and H_2 have multiple classifications (i.e., multiple hypernyms in our case) for the object e, thus modeling the common problem of lexical ambiguity and polysemy. Let us suppose that H_1 is the learned taxonomy, and H_2 the gold standard one. We start comparing the clusterings at cut 0 and stop at cut $k_r - 1$, where k_r is the depth of the

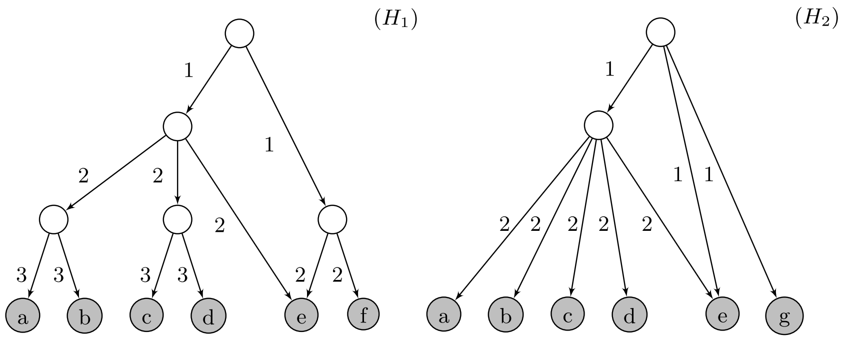

Figure 8
Two hierarchical clusters of n non-identical objects.

gold standard taxonomy. This means that if the learned taxonomy is less structured we replicate the cut $k_l - 1$ for $k_r - k_l$ times (where k_l is the maximum depth of the learned taxonomy), whereas if it is more structured we stop at cut $k_r - 1$. In contrast to previous evaluation models, our aim is to reward (instead of penalize) more structured taxonomies provided they still match the gold standard one.

Table 7 shows the cuts from 0 to 3 of H_1 and H_2 and the values of $B^i_{1,2}$. For $i = 2$ the B value is 0, if H_2 is the learned taxonomy, and is not defined, if H_2 is the gold standard. Therefore, when computing the cumulative Equation (9), we obtain the desired effect of penalizing less the structured learned taxonomies. Note that, when the two hierarchies have different depths, the value $k - 1$ in Equation (9) is replaced by $k_r - 1$.

Finally, we briefly compare our evaluation approach with the OntoRand index, introduced by Brank, Mladenic, and Grobelnik (2006). The Rand Index measures the similarity between two clusterings C_l and C_r by the formula:

$$R(C_l, C_r) = \frac{2(n_{11} + n_{00})}{n(n-1)} \tag{10}$$

where n_{11}, n_{00}, and n have the same meaning described earlier. In Brank, Mladenic, and Grobelnik (2006), a clustering is obtained from an ontology by associating each ontology instance to its concept. The set of clusters is hence represented by the set of leaf concepts in the hierarchy, namely, according to our notation, the clustering C_i^{k-1}. In order to take into account the hierarchical structure, they define the OntoRand formula. This measure, rather than summing up to 1 or 0, depending on whether or not two given instances i and j belong to the same cluster in the compared ontologies, returns a real number in $[0, 1]$ depending upon the distance between i and j in terms of common ancestors. In other terms, if i and j do not belong to the same concept but have a very close common ancestor, the OntoRand measure returns a value still close to 1.

Our measure has several advantages over the OntoRand index:

i) It allows for a comparison at different levels of depth of the hierarchy, and the cumulative similarity measure penalizes the contribution of the highest cuts of the hierarchy.

ii) It does not require that the two hierarchies have the same depth, nor that they have the same number of leaf nodes.

iii) The measure can be extended to lattices (e.g., it is not required that each object belongs precisely to one cluster).

Table 7
Application of the evaluation method to the hierarchies of Figure 8. The values of $B^i_{1,2}$ are shown both when H_1 and H_2 are the learned taxonomy (penultimate and last column, respectively).

i	C_1	C_2	n_{11}	n_{10}	n_{01}	H_1	H_2
						$B^i_{1,2}$	
0	{a,b,c,d,e,f}	{a,b,c,d,e,g}	10	5	5	0.67	0.67
1	{a,b,c,d,e},{e,f}	{a,b,c,d,e},{e},{g}	10	1	0	0.95	0.95
2	{a,b},{c,d},{e},{f}	{a},{b},{c},{d},{e},{g}	0	2	0	n.a.	0
3	{a},{b},{c},{d},{e},{f}	{a},{b},{c},{d},{e},{g}	0	0	0	n.a.	n.a.

iv) It is not dependent, as the Rand Index is, on the number n_{00}, the value of which has the undesirable effect of producing an ever higher similarity as the number of singleton clusters grows (Morey and Agresti 1984).

4.2.2 Results. This section presents the results of the F&M evaluation model for gold standard evaluation, therefore we focus on four domains and do not consider AI and FINANCE. The three WordNet sub-hierarchies are also compared with the taxonomies automatically created by Kozareva and Hovy (2010) in the same domains, kindly made available by the authors. It is once more to be noted that Kozareva and Hovy, during hypernym extraction, reject all the nodes not belonging to WordNet, whereas we assume no a-priori knowledge of the domain, apart from adopting the same set of seed terms used by K&H.

Figure 9 shows, for each domain (ANIMALS, PLANTS, VEHICLES, and VIRUSES), and for each cut level of the hierarchy, a plot of $B^i_{1,2}$ values multiplied by the penalty factor. As far as the comparison with K&H is concerned, we notice that, though K&H obtain better performance in general, OntoLearn has higher coverage over the domain, as is shown by the highest values for $i = 0$, and has a higher depth of the derived hierarchy, especially with $DAG[0, 99]$. Another recurrent phenomenon is that K&H curves gracefully degrade from the root to the leaf nodes, possibly with a peak in the intermediate levels, whereas OntoLearn has a hollow in the mid-high region (see the region 4–6 for ANIMALS and 1–2 for the other three hierarchies) and often a relative peak in the lowest

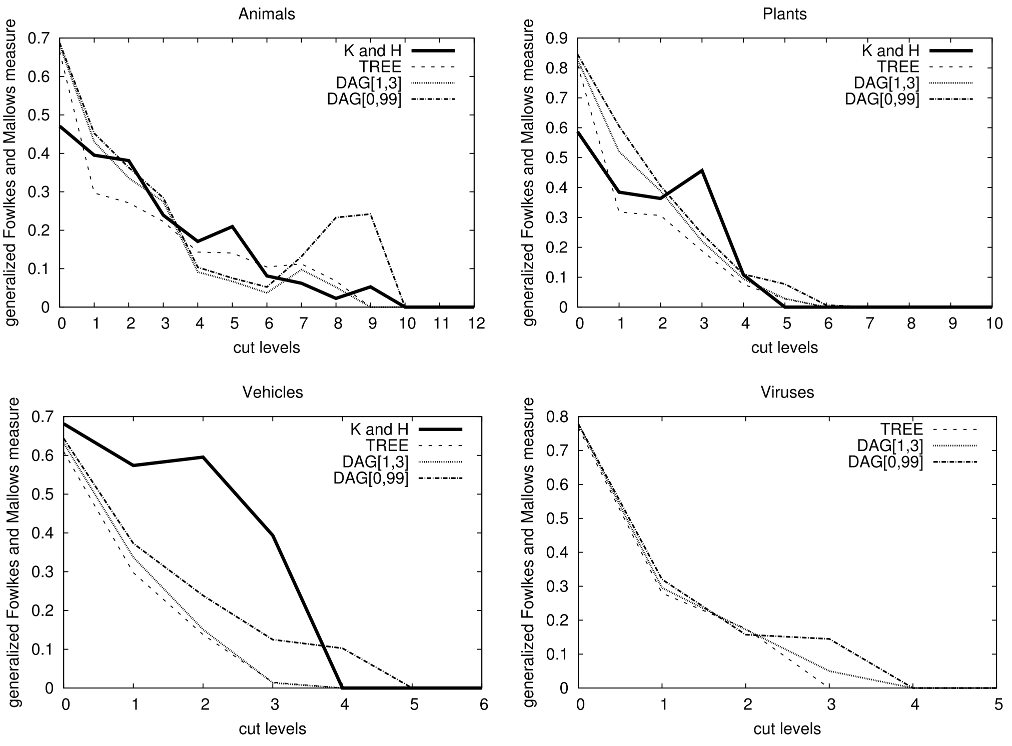

Figure 9
Gold standard evaluation of our three versions of OntoLearn Reloaded against WordNet (ANIMALS, PLANTS, and VEHICLES) and MeSH (VIRUSES). A comparison with K&H is also shown for the first three domains.

levels. In the manual evaluation section we explain this phenomenon, which also occurs in the ARTIFICIAL INTELLIGENCE taxonomy.

The generally decreasing values of $B^i_{1,2}$ in Figure 9 show that, as expected, mimicking the clustering criteria of a taxonomy created by a team of experts proves very difficult at the lowest levels, while performance grows at the highest levels. At the lowest taxonomy levels there are two opposing phenomena: overgeneralization and overspecialization. For example, *macaque* has *monkey* as a direct hypernym in WordNet, and we find *short-tailed monkey* as a direct hypernym of *macaque*. An opposite case is *ganoid*, which is a *taleostan* in WordNet and simply a *fish* in our taxonomy. The first case does not reward the learned taxonomy (though, unlike for the *overlapping factor* [Maedche, Pekar, and Staab 2002], it does not cause a penalty), whereas the second is quite penalizing. More of these examples will be provided in Section 4.3.

Finally, in Table 8 we show the cumulative $B_{1,2}$ values for the four domains, according to Equation (9). Here, except for the VEHICLES domain, the unconstrained $DAG[0, 99]$ performs best.

4.3 Manual Evaluation

This section is dedicated to the manual assessment of the learned ontologies. The section is divided in three parts: Section 4.3.1 is concerned with the human validation of hypernymy relations, Section 4.3.2 examines the global learned taxonomic structure in the search for common phenomena across the six domains, and finally Section 4.3.3 investigates the possibility of enhancing our hypernymy harvesting method with K&H's Hearst-like patterns, applying their method to the AI domain and manually evaluating the extracted hypernyms.

4.3.1 Hypernym Evaluation. To reduce subjectivity in taxonomy evaluation, we asked three annotators, only one of whom was a co-author, to validate, for each of the three experiments of each of the six domains, a random sample of hypernymy relations. For each relation the definition(s) supporting the relation were also provided. This was especially helpful for domains like VIRUSES, but also PLANTS and ANIMALS, in which the annotators were not expert. The size of each random sample was 300 for the (larger) AI and FINANCE domains and 100 for the others.

Each evaluator was asked to tag incorrect relations, regardless of whether the error was due to the selection of non-domain definitions (e.g., for VEHICLES: "a **driver** is a *golf club* with a near vertical face that is used for hitting long shots from the tee"), to a poor definition (e.g., for AI: "a **principle** is a fundamental essence, particularly one producing a given quality") or to a wrong selection of the hypernym. As an example of the latter, in the PLANTS domain, we extracted the hypernym *species* from the sentence: "**geranium** is a genus of 422 *species* of flowering annual, biennial, and perennial plants

Table 8
Values of $B_{1,2}$ for the domains of VIRUSES, ANIMALS, PLANTS, and VEHICLES.

Experiment	VIRUSES	ANIMALS	PLANTS	VEHICLES
TREE	0.093	0.064	0.059	0.065
DAG [1,3]	0.101	0.062	0.072	0.069
DAG [0,99]	**0.115**	**0.097**	**0.080**	0.103
K&H	n.a.	0.067	0.068	**0.158**

Table 9
Precision of hypernym edges on six domains (calculated on a majority basis) and inter-annotator agreement on the corresponding sample of relations.

Experiment		# of Sample	Precision		κ
AI	TREE	300	80.3%	[241/300]	0.45
	DAG [1,3]	300	73.6%	[221/300]	0.42
	DAG [0,99]	300	73.0%	[219/300]	0.41
FINANCE	TREE	300	93.6%	[281/300]	0.40
	DAG [1,3]	300	93.0%	[279/300]	0.43
	DAG [0,99]	300	92.6%	[278/300]	0.41
VIRUSES	TREE	100	99.0%	[99/100]	0.49
	DAG [1,3]	100	99.0%	[99/100]	0.39
	DAG [0,99]	100	99.0%	[99/100]	0.32
ANIMALS	TREE	100	92.0%	[92/100]	0.53
	DAG [1,3]	100	92.0%	[92/100]	0.36
	DAG [0,99]	100	90.0%	[90/100]	0.56
PLANTS	TREE	100	89.0%	[89/100]	0.49
	DAG [1,3]	100	85.0%	[85/100]	0.53
	DAG [0,99]	100	97.0%	[97/100]	0.26
VEHICLES	TREE	100	92.0%	[92/100]	0.64
	DAG [1,3]	100	92.0%	[92/100]	0.49
	DAG [0,99]	100	91.0%	[91/100]	0.44

κ	Interpretation
< 0	Poor agreement
0.01–0.20	Slight agreement
0.21–0.40	Fair agreement
0.41–0.60	Moderate agreement
0.61–0.80	Substantial agreement
0.81–1.00	Almost perfect agreement

that are commonly known as the cranesbills" since, in the WCL verb set, we have *"is a * species of"* and *"is a * genus of"*, but not the concatenation of these two patterns. Annotators could mark with *?* a hyponym–hypernym pair for which they felt uncertain. Though it would have been useful to distinguish between the different types of error, we found that regarding many error types there was, anyway, very low inter-annotator agreement. Indeed the annotation task would appear to be intrinsically complex and controversial. In any case, an assessment of the definition and hypernym extraction tasks in isolation was already provided by Navigli and Velardi (2010).

Table 9 summarizes the results. Precision of each classification was computed on a majority basis, and we used Fleiss' kappa statistics (Fleiss 1971) to measure the inter-annotator agreement. In general, the precision is rather good, though it is lower for the AI domain, probably due to its high "vitality" (many new terms continuously arise, and for some of them it is difficult to find good quality definitions). In general, precision is higher in focused domains (VIRUSES, ANIMALS, PLANTS, and VEHICLES) than in wide-range domains (AI and FINANCE). The former domains, however, have just one quite "narrow" upper concept (*virus* for VIRUSES, etc.), whereas AI and FINANCE have several upper concepts (e.g., *person* or *abstraction*), and furthermore they are less focused. This means that there is an inherently higher ambiguity and this may be seen as justifying the lower performance. In Table 9 we also note that TREE structures achieve in general a higher precision, except for PLANTS, whereas the DAG has the advantage of improving recall (see also Section 4.2.2).

Note that high precision here does not contradict the results shown in Section 4.2.2: In this case, each single relation is evaluated in isolation, therefore overgenerality or overspecificity do not imply a penalty, provided the relation is judged to be correct.

Furthermore, global consistency is not considered here: for example, *distance metric learning* ← *parametric technique*, and eventually ends up in *technique*, whereas *belief network learning* ← *machine learning algorithm* ends up in *algorithm* and then in *procedure*. In isolation, these hypernymy patterns are acceptable, but within a taxonomic structure one would like to see a category node grouping all terms denoting machine learning algorithms. This behavior should be favored by the node weighting strategy described in Section 3.4, aimed at attracting nodes with multiple hypernyms towards the most populated category nodes. As in the previous example, however, there are category nodes that are almost equally "attractive" (e.g., *algorithm* and *technique*), and, furthermore, the taxonomy induction algorithm can only select among the set of hypernyms extracted during the harvesting phase. Consequently, when no definition suggests that *distance metric learning* is a hyponym of *machine learning algorithm*, or of any other concept connected to *machine learning algorithm*, there is no way of grouping *distance metric learning* and *belief network learning* in the desired way. This task must be postponed to manual post-editing.

Concerning the kappa statistics, we note that the values range from moderate to substantial in most cases. These numbers are apparently low, but the task of evaluating hypernymy relations is quite a complex one. Similar kappa values were obtained in Yang and Callan (2008) in a human-guided ontology learning task.

4.3.2 Structural Assessment. In addition to the manual evaluation summarized in Table 9, a structural assessment was performed to identify the main sources of error. To this end, one of the authors analyzed the full AI and FINANCE taxonomies and a sample of the other four domains in search of recurring errors. In general, our optimal branching algorithm and weighting schema avoids many of the problems highlighted in well-known studies on taxonomy acquisition from dictionaries (Ide and Véronis 1993), like circularity, over-generalization, and so forth. There are new problems to be faced, however.

The main sources of error are the following:

- Ambiguity of terms, especially at the intermediate levels
- Low quality of definitions
- Hypernyms described by a clause rather than by a single- or multi-word expression
- Lack of an appropriate WCL to analyze the definition
- Difficulty of extracting the correct hypernym string from phrases with identical syntactic structure

We now provide examples for each of these cases.

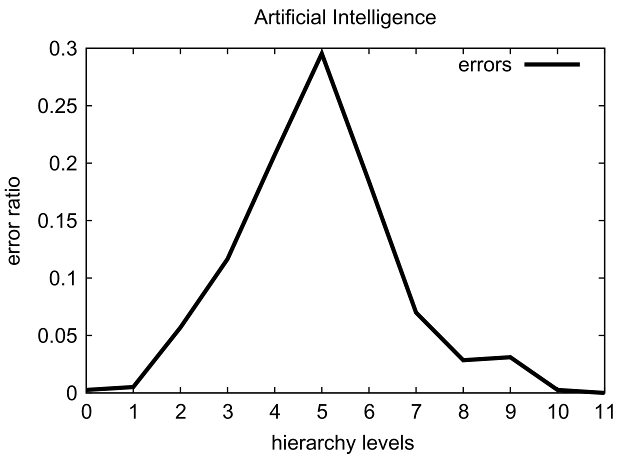

Figure 10
Error distribution of the TREE version of our algorithm on the ARTIFICIAL INTELLIGENCE domain.

Ambiguity. Concerning ambiguity of terms, consider Figures 10 and 11, which show the distribution of errors at the different levels of the learned AI and FINANCE taxonomies for the TREE experiment. The figures provide strong evidence that most errors are located in the intermediate levels of the taxonomy. As we move from leaf nodes to the upper ontology, the extracted terms become progressively more general and consequently more ambiguous. For these terms the domain heuristics may turn out to be inadequate, especially if the definition is a short sentence.

But why are these errors frequent at the intermediate levels and not at the highest levels? To understand this, consider the following example from the AI domain: For the term *classifier* the wrong hypernym is selected from the sentence "**classifier** is a *person* who creates classifications." In many cases, wrong hypernyms do not accumulate sufficient weight and create "dead-end" hypernymy chains, which are pruned during the optimal branching step. But, unfortunately, a domain appropriate definition is

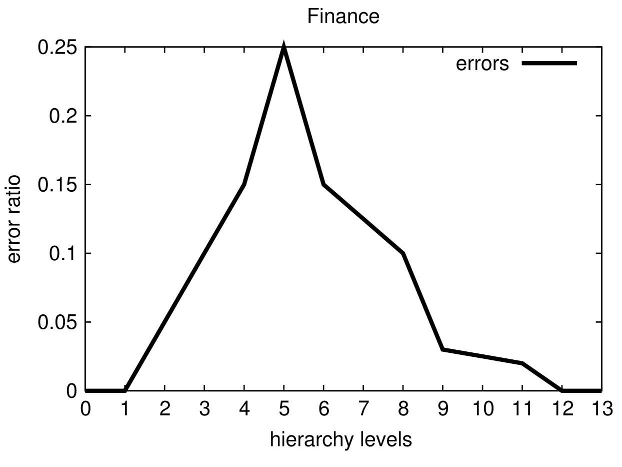

Figure 11
Error distribution of the TREE version of our algorithm on the FINANCE domain.

found for *person*: "**person** is the more general *category* of an individual," due to the presence of the domain word *category*. On the other hand, this new sentence produces an attachment that, in a sense, recovers the error, because *category* is a "good" domain concept that eventually ends up in subsequent iterations to the upper node *abstraction*. Therefore, what happens is that the upper taxonomy nodes, with the help of the domain heuristic, mitigate the "semantic drift" caused by out-of-domain ambiguity, recovering the ambiguity errors of the intermediate levels. This phenomenon is consistently found in all domains, as shown by the hollow that we noticed in the graphs of Section 4.2.2. An example in the ANIMALS domain is represented by the hypernymy sequence *fawn* ← *color* ← *race* ← *breed* ← *domestic animal*, where the wrong hypernym *color* was originated by the sentence "**fawn** is a light yellowish brown *color* that is usually used in reference to a dog's coat color." Only in VIRUSES is the phenomenon mitigated by the highly specific and very focused nature of the domain.

In addition to out-of-domain ambiguity, we have two other phenomena: in-domain ambiguity and polysemy. In-domain ambiguity is rare, but not absent (Agirre et al. 2010; Faralli and Navigli 2012). Consider the example of Figure 12a, from the VEHICLES domain: *tractor* has two definitions corresponding to two meanings, which are both correct. The airplane meaning is "**tractor** is an *airplane* where the propeller is located in front of the fuselage," whereas the truck meaning is "**tractor** is a *truck* for pulling a semi-trailer or trailer." Here the three hyponyms of *tractor* (see the figure) all belong to the *truck* sense. We leave to future developments the task of splitting in-domain ambiguous nodes in the appropriate way.

Another case is systematic polysemy, which is shown in Figure 13. The graph in the figure, from the AI domain, captures the fact that a *semantic network*, as well as its hyponyms, are both a *methodology* and a *representation*. Another example is shown in Figure 12b for the PLANTS domain, where systematic polysemy can be observed for terms like *olive*, *orange*, and *breadfruit*, which are classified as *evergreen tree* and *fruit*. Polysemy, however, does not cause errors, as it does for in-domain ambiguity, because hyponyms of polysemous concepts inherit the polysemy: In the two graphs of Figures 13 and 12b, both *partitioned semantic network* and *tangerine* preserve the polysemy of their ancestors. Note that in-domain ambiguity and polysemy are only captured by the DAG structure; therefore this can be seen as a further advantage (in addition to higher recall) of the DAG model over and against the more precise TREE structure.

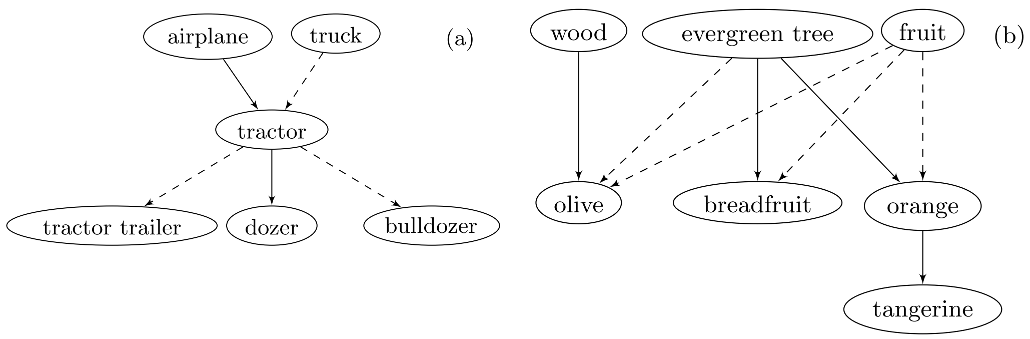

Figure 12
An example of in-domain ambiguity (a) and an example of systematic polysemy (b). Dashed edges were added to the graph as a result of the edge recovery phase (see Section 3.5).

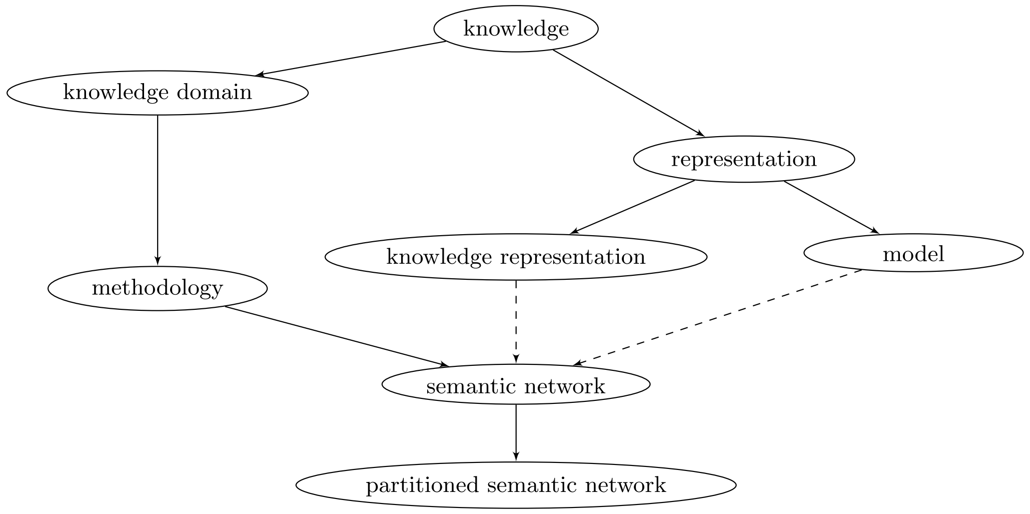

Figure 13
An example of systematic polysemy. Dashed edges were added to the graph as a result of the edge recovery phase (see Section 3.5).

Low quality of definitions. Often textual definitions, especially if extracted from the Web, do not have a high quality. Examples are: "**artificial intelligence** is the next big development in computing" or "**aspectual classification** is also a necessary prerequisite for interpreting certain adverbial adjuncts." These sentences are definitions on a syntactic ground, but not on a semantic ground. As will be shown in Section 4.3.3, this problem is much less pervasive than for Hearst-like lexico-syntactic patterns, although, neither domain heuristics nor the graph pruning could completely eliminate the problem. We can also include overgeneralization in this category of problems: Our algorithm prefers specific hypernyms to general hypernyms, but for certain terms no specific definitions are found. The elective way to solve this problem would be to assign a quality confidence score to the definition source (document or Web page), for example, by performing an accurate and stable classification of its genre (Petrenz and Webber 2011).

Hypernym is a clause. There are cases in which, although very descriptive and good quality definitions are found, it is not possible to summarize the hypernym with a term or multi-word expression. For example "**anaphora resolution** is the process of determining whether two expressions in natural language refer to the same real world entity." OntoLearn extracts *process of determining* which ends up in *procedure, process*. This is not completely wrong, however, and in some case is even fully acceptable, as for "**summarizing** is a process of condensing or expressing in short something you have read, watched or heard": here, *process of condensing* is an acceptable hypernym. An example for FINANCE is: "**market-to-book ratio** is book value of assets minus book value of equity plus market value of equity," where we extracted *book value*, rather than the complete formula. Another example is: "**roa** is defined as a ratio of operating income to book value of assets," from which we extracted *ratio*, which is, instead, acceptable.

Lack of an appropriate definitional pattern. Though we acquired hundreds of different definitional patterns, there are still definitions that are not correctly parsed. We already mentioned the *geranium* example in the PLANTS domain. An example in the AI domain is "**execution monitoring** is the robot's process of observing the world for discrepancies between the actual world and its internal representation of it," where the extracted hypernym is *robot* because we have no WCL with a Saxon genitive.

Wrong hypernym string. This is the case in which the hypernym is a superstring or substring of the correct one, like: "**latent semantic analysis** is a machine learning procedure." Here, the correct hypernym is *machine learning procedure*, but OntoLearn extracts *machine* because *learning* is POS tagged as a verb. In general, it is not possible to evaluate the extent of the hypernym phrase except case-by-case. The lattice learner acquired a variety of hypernymy patterns, but the precision of certain patterns might be quite low. For example, the hypernymy pattern "* of *" is acceptable for "In grammar, a **lexical category** is a *linguistic category of words*" or "**page rank** is a *measure of site popularity*" but not for "**page rank** is only a *factor of the amount* of incoming and outgoing links to your site" nor for "**pattern recognition** is an artificial intelligence *area of considerable importance*." The same applies to the hypernymy pattern *ADJ NN*: *important algorithm* is wrong, although *greedy algorithm* is correct.

4.3.3 Evaluation of Lexico-Syntactic Patterns. As previously remarked, Kozareva and Hovy (2010) do not actually apply their algorithm to the task of creating a new taxonomy, but rather they try to reproduce three WordNet taxonomies, under the assumption that the taxonomy nodes are known (cf. Section 4). Therefore, there is no evidence of the precision of their method on new domains, where the category nodes are unknown. On the other hand, if Hearst's patterns, which are at the basis of K&H's hypernymy harvesting algorithm, could show adequate precision, we would use them in combination with our definitional patterns. This section investigates the matter.

As briefly summarized in Section 2, K&H create a hypernym graph in three steps. Given a few root concepts (e.g., *animal*) and basic level concepts or instances (e.g., *lion*), they:

1) harvest new basic and intermediate concepts from the Web in an iterative fashion, using doubly anchored patterns (DAP) like '$\langle$*root*$\rangle$ *such as* $\langle$*seed*$\rangle$ *and* $*$' and inverse DAP (i.e., DAP^{-1}) like '$*$ *such as* $\langle$*term*1$\rangle$ *and* $\langle$*term*2$\rangle$'. The procedure is iterated until no new terms can be found;

2) rank the nodes extracted with DAP by out-degree and those extracted with inverse DAP by in-degree, so as to prune out less promising terms;

3) induce the final taxonomic structure by positioning the intermediate nodes between basic level and root terms using a concept positioning procedure based on a variety of Hearst-like surface patterns. Finally, they eliminate cycles, as well as nodes with no predecessor or no successor, and they select the longest path in the case of multiple paths between node pairs.

In this section we apply their method[23] to the domain of AI in order to manually analyze the quality of the extracted relations. To replicate the first two steps of K&H algorithm we fed the algorithm with a growing set of seed terms randomly selected from our validated terminology, together with their hierarchically related root terms

23 We followed the exact procedure described in Figure 2 of Kozareva & Hovy (2010).

Table 10
K&H performance on the AI domain.

number of root/seed pairs	1	10	100	1,000
# new concepts	131	163	227	247
# extracted *is-a* relations	114	146	217	237
correct and in-domain	21.05% (24/114)	24.65% (36/146)	18.89% (41/217)	18.56% (44/237)

in the upper taxonomy (e.g., *unsupervised learning* is a *method* or *maximum entropy* is a *measure*). We then performed the DAP and DAP^{-1} steps iteratively until no more terms could be retrieved, and we manually evaluated the quality of the harvested concepts and taxonomic relations using the same thresholding formula described in K&H.[24] We give the results in Table 10.

As we said earlier, our purpose here is mainly to evaluate the quality of Hearst patterns in more technical domains, and the efficacy of DAP and DAP^{-1} steps in retrieving domain concepts and relations. Therefore, replicating step (3) above is not useful in this case since, rather than adding new nodes, step (3) is aimed, as in our optimal branching and pruning recovery steps, at reorganizing and trimming the final graph.

Table 10 should be compared with the first three rows (AI) of Table 9: It shows that in the absence of a priori knowledge on the domain concepts the quantity and quality of the *is-a* links extracted by the K&H algorithm is much lower than those extracted by OntoLearn Reloaded. First, the number of new nodes found by the K&H algorithm is quite low: For the same domain of ARTIFICIAL INTELLIGENCE, our method, as shown in Table 9, is able to extract from scratch 2,387 – 52 = 2,335 nodes,[25] in comparison with the 247 new nodes of Table 10, obtained with 1,000 seeds. Second, many nodes extracted by the K&H algorithm, like *fan speed*, *guidelines*, *chemical engineering*, and so on, are out-of-domain and many hypernym relations are incorrect irrespective of their direction, like *computer program* is a *slow* and *data mining* is a *contemporary computing problem*. Third, the vast majority of the retrieved hypernyms are overgeneral, like *discipline, method, area, problem, technique, topic*, and so forth, resulting in an almost flat hypernymy structure. A high in-degree threshold and a very high number of seeds do not mitigate the problem, demonstrating that Hearst-like patterns are not very good at harvesting many valid hypernym relations in specialized domains.[26]

Following this evaluation, we can outline several advantages of our method over K&H's work (and, as a consequence, over Hearst's patterns):

i) We obtain higher precision and recall when no a priori knowledge is available on the taxonomy concepts, because hypernyms are extracted from expert knowledge on the Web (i.e., technical definitions rather than patterns reflecting everyday language).

24 The technique is based on the in-degree and out-degree of the graph nodes.
25 Remember that the 52 domain-independent upper terms are manually defined (cf. Section 4.1.4).
26 This result is in line with previous findings in a larger, domain-balanced experiment (Navigli and Velardi 2010) in which we have shown that WCLs outperform Hearst patterns and other methods in the task of hypernym extraction.

ii) We cope better with sense ambiguity via the domain filtering step.[27]

iii) We use a principled algorithmic approach to graph pruning and cycle removal.[28]

iv) Thanks to the support provided by textual definitions, we are able to cope with the problem of manually evaluating the retrieved concepts and relations, even in the absence of a reference taxonomy.

4.3.4 Summary of Findings. We here summarize the main findings of our manifold evaluation experiments:

i) With regard to the two versions of our graph pruning algorithm, we found that TREE structures are more precise, whereas DAGs have a higher recall.

ii) Errors are mostly concentrated in the mid-level of the hierarchy, where concepts are more ambiguous and the "attractive" power of top nodes is less influential. This was highlighted by our quantitative (F&M) model and justified by means of manual analysis.

iii) The quality and number of definitions is critical for high performance. Less-focused domains in which new terms continuously emerge are the most complex ones, because it is more difficult to retrieve high-quality definitions for them.

iv) Definitions, on the other hand, are a much more precise and high-coverage source of knowledge for hypernym extraction than (Hearst-like) patterns or contexts, because they explicitly represent expert knowledge on a given domain. Furthermore, they are a very useful support for manual validation and structural analysis.

5. Conclusions

In this paper we presented OntoLearn Reloaded, a graph-based algorithm for learning a taxonomy from scratch using highly dense, potentially disconnected, hypernymy graphs. The algorithm performs the task of eliminating noise from the initial graph remarkably well on arbitrary, possibly specialized, domains, using a weighting scheme that draws both on the topological properties of the graph and on some general principles of taxonomic structures. OntoLearn Reloaded provides a considerable advancement over the state of the art in taxonomy learning. First, it is the first algorithm that experimentally demonstrates its ability to build a new taxonomy from the ground up, without any a priori assumption on the domain except for a corpus and a set of (possibly general) upper terms. The majority of existing systems start from a set of concepts and induce hypernymy links between these concepts. Instead, we automatically learn both concepts and relations via term extraction and iterative definition and hypernym

27 In the authors' words (Kozareva and Hovy 2010, page 1,115): "we found that the learned terms in the middle ranking do not refer to the meaning of vehicle as a transportation device, but to the meaning of vehicle as media (i.e., *seminar*, *newspapers*), communication and marketing."

28 Again in the authors' words (Kozareva and Hovy 2010, page 1,115): "we found that in-degree is not sufficient by itself. For example, highly frequent but irrelevant hypernyms such as *meats* and *others* are ranked at the top of the list, while low frequent but relevant ones such as *protochordates*, *hooved-mammals*, *homeotherms* are discarded."

extraction. Second, we cope with issues such as term ambiguity, complexity, and multiplicity of hypernymy patterns. Third, we contribute a multi-faceted evaluation, which includes a comparison against gold standards, plus a structural and a manual evaluation. Taxonomy induction was applied to the task of creating new ARTIFICIAL INTELLIGENCE and FINANCE taxonomies and four taxonomies for gold-standard comparison against WordNet and MeSH.[29]

Our experimental analysis shows that OntoLearn Reloaded greatly simplifies the task of acquiring a taxonomy from scratch: Using a taxonomy validation tool,[30] a team of experts can correct the errors and create a much more acceptable taxonomy in a matter of hours, rather than man-months, also thanks to the automatic acquisition of textual definitions for our concepts. As with any automated and unsupervised learning tool, however, OntoLearn does make errors, as we discussed in Section 4. The accuracy of the resulting taxonomy is clearly related to the number and quality of discovered definitional patterns, which is in turn related to the maturity and generality of a domain. Even with good definitions, problems might arise due to in- and out-domain ambiguity, the latter being probably the major source of errors, together with complex definitional structures. Although we believe that there is still room for improvement to OntoLearn Reloaded, certain errors would appear unavoidable, especially for less focused and relatively dynamic domains like ARTIFICIAL INTELLIGENCE and FINANCE, in which new terms arise continuously and have very few, or no definitions on the Web.

Future work includes the addition of non-taxonomical relations along the lines of ReVerb (Etzioni et al. 2011) and WiSeNet (Moro and Navigli 2012), and a more sophisticated rank-based method for scoring textual definitions. Finally, we plan to tackle the issue of automatically discriminating between in-domain ambiguity and systematic polysemy (as discussed in Section 4.3.2).

Acknowledgments

Stefano Faralli and Roberto Navigli gratefully acknowledge the support of the ERC Starting Grant MultiJEDI No. 259234. The authors wish to thank Jim McManus for his valuable comments on the paper, and Zornitsa Kozareva and Eduard Hovy for making their data available.

References

Agirre, Eneko, Oier López de Lacalle, Christiane Fellbaum, Shu-Kai Hsieh, Maurizio Tesconi, Monica Monachini, Piek Vossen, and Roxanne Segers. 2010. SemEval-2010 Task 17: All-words Word Sense Disambiguation on a specific domain. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval-2010)*, pages 75–80, Uppsala.

Berland, Matthew and Eugene Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 57–64, College Park, MD.

Biemann, Chris. 2005. Ontology learning from text—A survey of methods. *LDV-Forum*, 20(2):75–93.

Brank, Janez, Dunja Mladenic, and Marko Grobelnik. 2006. Gold standard based ontology evaluation using instance assignment. In *Proceedings of 4th Workshop Evaluating Ontologies for the Web (EON)*, Edinburgh.

Carpineto, Claudio and Giovanni Romano. 2012. Consensus Clustering Based on a New Probabilistic Rand Index with Application to Subtopic Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2315–2326.

Chu, Yoeng-Jin and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Cimiano, Philipp, Andreas Hotho, and Steffen Staab. 2005. Learning concept

29 Data sets are available at: `http://lcl.uniroma1.it/ontolearn_reloaded`.
30 For example, `http://lcl.uniroma1.it/tav/`.

hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research*, 24(1):305–339.

Cohen, Trevor and Dominic Widdows. 2009. Empirical distributional semantics: Methods and biomedical applications. *Journal of Biomedical Informatics*, 42(2):390–405.

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. 1990. *Introduction to Algorithms*. MIT Electrical Engineering and Computer Science. MIT Press, Cambridge, MA.

De Benedictis, Flavio, Stefano Faralli, and Roberto Navigli. 2013. GlossBoot: Bootstrapping Multilingual Domain Glossaries from the Web. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (ACL), Sofia.

De Nicola, Antonio, Michele Missikoff, and Roberto Navigli. 2009. A software engineering approach to ontology building. *Information Systems*, 34(2):258–275.

Edmonds, Jack. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Etzioni, Oren, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3–10, Barcelona.

Fahmi, Ismail and Gosse Bouma. 2006. Learning to identify definitions using syntactic features. In *Proceedings of the EACL 2006 workshop on Learning Structured Information in Natural Language Applications*, pages 64–71, Trento.

Faralli, Stefano and Roberto Navigli. 2012. A new minimally supervised framework for domain Word Sense Disambiguation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1,411–1,422, Jeju.

Fellbaum, Christiane, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Fleiss, Joseph L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Fountain, Trevor and Mirella Lapata. 2012. Taxonomy induction using hierarchical random graphs. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*, pages 466–476, Montréal.

Fowlkes, Edward B. and Colin L. Mallows. 1983. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569.

Girju, Roxana, Adriana Badulescu, and Dan Moldovan. 2006. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135.

Gomez-Perez, Asunción and David Manzano-Mancho. 2003. A survey of ontology learning methods and techniques. OntoWeb Delieverable 1.5. Universidad Politécnica de Madrid.

Guarino, Nicola and Chris Welty. 2002. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65.

Hearst, Marti A. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the* 14^{th} *International Conference on Computational Linguistics (COLING)*, pages 539–545, Nantes.

Hovy, Eduard, Andrew Philpot, Judith Klavans, Ulrich Germann, and Peter T. Davis. 2003. Extending metadata definitions by automatically extracting and organizing glossary definitions. In *Proceedings of the 2003 Annual National Conference on Digital Government Research*, pages 1–6, Boston, MA.

Ide, Nancy and Jean Véronis. 1993. Extracting knowledge bases from machine-readable dictionaries: Have we wasted our time? In *Proceedings of the Workshop on Knowledge Bases and Knowledge Structures*, pages 257–266, Tokyo.

Kozareva, Zornitsa and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the Web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1,110–1,118, Cambridge, MA.

Kozareva, Zornitsa, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the Web with hyponym pattern linkage graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1,048–1,056, Columbus, OH.

Maedche, Alexander, Viktor Pekar, and Steffen Staab. 2002. Ontology learning part one—on discovering taxonomic relations from the Web. In N. Zhong, J. Liu, and Y. Y. Yao, editors, *Web Intelligence*. Springer Verlag, Berlin, pages 301–322.

Maedche, Alexander and Steffen Staab. 2009. Ontology learning. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*. Springer, Berlin, pages 245–268.

Miller, George A., R. T. Beckwith, Christiane D. Fellbaum, D. Gross, and K. Miller. 1990. WordNet: An online lexical database. *International Journal of Lexicography*, 3(4):235–244.

Morey, Leslie C. and Alan Agresti. 1984. The measurement of classification agreement: An adjustment to the Rand statistic for chance agreement. *Educational and Psychological Measurement*, 44:33–37.

Moro, Andrea and Roberto Navigli. 2012. WiSeNet: Building a Wikipedia-based semantic network with ontologized relations. In *Proceedings of the* 21*st ACM Conference on Information and Knowledge Management (CIKM 2012)*, pages 1,672–1,676, Maui, HI.

Navigli, Roberto. 2009. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.

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

Navigli, Roberto and Paola Velardi. 2004. Learning domain ontologies from document warehouses and dedicated websites. *Computational Linguistics*, 30(2):151–179.

Navigli, Roberto and Paola Velardi. 2005. Structural semantic interconnections: A knowledge-based approach to Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1075–1088.

Navigli, Roberto and Paola Velardi. 2010. Learning Word-Class Lattices for definition and hypernym extraction. In *Proceedings of the* 48*th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1,318–1,327, Uppsala.

Navigli, Roberto, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the* 22*nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1,872–1,877, Barcelona.

Newman, Mark E. J. 2010. *Networks: An Introduction*. Oxford University Press.

Pado, Sebastian and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Pantel, Patrick and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of* 44*th Annual Meeting of the Association for Computational Linguistics joint with* 21*st Conference on Computational Linguistics (COLING-ACL)*, pages 113–120, Sydney.

Pasca, Marius. 2004. Acquisition of categorized named entities for web search. In *Proceedings of the* 13*th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 137–145, Washington, DC.

Petasis, Georgios, Vangelis Karkaletsis, Georgios Paliouras, Anastasia Krithara, and Elias Zavitsanos. 2011. Ontology population and enrichment: State of the art. In Georgios Paliouras, Constantine Spyropoulos, and George Tsatsaronis, editors, *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution*, volume 6050 of *Lecture Notes in Computer Science*. Springer, Berlin / Heidelberg, pages 134–166.

Petrenz, Philipp and Bonnie L. Webber. 2011. Stable classification of text genres. *Computational Linguistics*, 37(2):385–393.

Ponzetto, Simone Paolo and Roberto Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In *Proceedings of the* 21*st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2,083–2,088, Pasadena, CA.

Ponzetto, Simone Paolo and Michael Strube. 2011. Taxonomy induction based on a collaboratively built knowledge repository. *Artificial Intelligence*, 175:1737–1756.

Poon, Hoifung and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 296–305, Uppsala.

Rand, William M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Schmid, Helmut. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*, pages 47–50, Dublin.

Sclano, Francesco and Paola Velardi. 2007. TermExtractor: A Web application to learn the shared terminology of emergent Web communities. In *Proceedings of the* 3th *International Conference on Interoperability for Enterprise Software and Applications (I-ESA)*, pages 287–290, Funchal.

Snow, Rion, Dan Jurafsky, and Andrew Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of* 44th *Annual Meeting of the Association for Computational Linguistics joint with* 21st *Conference on Computational Linguistics (COLING-ACL)*, pages 801–808, Sydney.

Sowa, John F. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA.

Storrer, Angelika and Sandra Wellinghoff. 2006. Automated detection and annotation of term definitions in German text corpora. In *Proceedings of the* 5th *International Conference on Language Resources and Evaluation (LREC)*, pages 2,373–2,376, Genova.

Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum. 2008. YAGO: A large ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6(3):203–217.

Tang, Jie, Ho Fung Leung, Qiong Luo, Dewei Chen, and Jibin Gong. 2009. Towards ontology learning from folksonomies. In *Proceedings of the* 21st *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2,089–2,094, Pasadena, CA.

Tarjan, Robert Endre. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

Velardi, Paola, Roberto Navigli, and Pierluigi D'Amadio. 2008. Mining the Web to create specialized glossaries. *IEEE Intelligent Systems*, 23(5):18–25.

Völker, Johanna, Denny Vrandečić, York Sure, and Andreas Hotho. 2008. AEON—An approach to the automatic evaluation of ontologies. *Journal of Applied Ontology*, 3(1-2):41–62.

Vossen, Piek. 2001. Extending, trimming and fusing WordNet for technical documents. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations (NAACL)*, pages 125–131, Pittsburgh, PA.

Wagner, Silke and Dorothea Wagner. 2007. Comparing clusterings: An overview. Technical Report 2006-04, Faculty of Informatics, Universität Karlsruhe (TH).

Westerhout, Eline. 2009. Definition extraction using linguistic and structural features. In *Proceedings of the RANLP Workshop on Definition Extraction*, pages 61–67, Borovets.

Yang, Hui and Jamie Callan. 2008. Human-guided ontology learning. In *Proceedings of Human-Computer Interaction and Information Retrieval (HCIR)*, pages 26–29, Redmond, WA.

Yang, Hui and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–279, Suntec.

Zavitsanos, Elias, Georgios Paliouras, and George A. Vouros. 2011. Gold standard evaluation of ontology learning methods through ontology transformation and alignment. *IEEE Transactions on Knowledge and Data Engineering*, 23(11):1635–1648.

Zhang, Chunxia and Peng Jiang. 2009. Automatic extraction of definitions. In *Proceedings of 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, pages 364–368, Beijing.