A Graph-based Algorithm for Inducing Lexical Taxonomies from Scratch

Roberto Navigli, Paola Velardi and Stefano Faralli
Dipartimento di Informatica
Sapienza Università di Roma
{navigli,velardi,faralli}@di.uniroma1.it

Abstract

In this paper we present a graph-based approach aimed at learning a lexical taxonomy automatically starting from a domain corpus and the Web. 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 possibly disconnected hypernym graph. The algorithm then induces a taxonomy from the graph. Our experiments show that we obtain high-quality results, both when building brand-new taxonomies and when reconstructing WordNet sub-hierarchies.

1 Introduction

It is widely accepted that ontologies can facilitate text understanding and automatic processing of textual resources. Moving from words to concepts is important for solving data sparseness issues and promises appealing solutions to polysemy and homonymy by finding unambiguous concepts within a domain [Biemann, 2005]. Indeed, ontologies have been proven to be useful for many different applications, such as question answering, information search and retrieval, etc.

A quite recent challenge is to automatically or semi-automatically create an ontology using textual data, thus reducing the time and effort needed for manual construction. Surveys on ontology learning from text and other sources (such as the Web) can be found in, among others, [Biemann, 2005; Perez and Mancho, 2003; Maedche and Staab, 2009]. In ontology learning from text, two main approaches are used. Rule-based approaches use pre-defined rules or heuristic patterns in order to extract terms and relations. These approaches are based on lexico-syntactic patterns, first introduced by Hearst [1992]. Lexical patterns for expressing a certain type of relation are used to discover instances of relations from text. Patterns can be chosen manually [Berland and Charniak, 1999; Kozareva *et al.*, 2008] or via automatic bootstrapping [Widdows and Dorow, 2002; Girju *et al.*, 2003]. 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] or clustering of formalized statements [Poon and Domingos, 2010]. Such approaches are based on the assumption that similar concepts[1] appear in similar contexts and their main advantage is that they are able to discover relations which do not explicitly appear in the text. However, they are less accurate and the selection of feature types, notion of context and similarity metrics vary considerably depending on which specific approach is used.

In this paper we are concerned with the problem of learning a taxonomy – the backbone of an ontology – entirely from scratch. Very few systems in the literature address this task. Among the most promising ones we mention Yang and Callan [2009], who present a semi-supervised taxonomy induction framework which integrates various features to learn an ontology metric, calculating a 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.

Snow *et al.* [2006] propose the incremental construction of taxonomies using a probabilistic model. In their work, they combine evidence from multiple classifiers using constraints from hyponymy and cousin relations. Given the body of evidence obtained from all the relevant word pairs, the taxonomy learning task is defined probabistically 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, this approach aims at attaching new concepts under the appropriate nodes of an existing taxonomy (i.e., WordNet [Fellbaum, 1998]).

A method which is closer to our research is presented in [Kozareva and Hovy, 2010]. From an initial given set of root concepts and basic level terms, the authors use Hearst-like lexico-syntactic patterns recursively to harvest new terms from the Web. The result of the first part of the algorithm is a set of hyponym-hypernym relations. To induce taxonomic relations between intermediate concepts they then search the Web 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

[1] Because we are concerned with lexical taxonomies, in this paper we use the words "concepts" and "terms" interchangeably.

the longest path in the case of multiple paths between concept pairs. Kozareva and Hovy's method has some limitations: first, patterns used to harvest hypernymy relations are very simple, thus they are inherently incapable of extracting relations for specialized domains, as shown by Navigli and Velardi [2010]; second, the pruning method does not produce a taxonomy, but an acyclic graph; furthermore, in evaluating their methodology, the authors select only nodes belonging to a WordNet sub-hierarchy (they experiment on plants, vehicles and animals), thus limiting themselves to Yang and Callan's target of finding relations between an assigned set of nodes.

In practice, none of the algorithms described in the literature actually creates a new, usable taxonomy from scratch, instead each measures 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[2]).

In this paper, we present a considerable advancement over the state of the art in taxonomy learning:

- First, except for the selection of just a few root nodes, this is the first algorithm to build a new taxonomy truly from scratch.
- 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 new algorithm to extract an optimal taxonomy from the resulting hypernymy graph. Taxonomy induction is based on the topological structure of the graph and some general properties of taxonomies.
- Fourth, our evaluation is not limited, as it is in most papers, to the number of retrieved hypernymy relations that are found in a reference taxonomy, because we also analyse the extracted taxonomy in its entirety. Furthermore, we also acquire a "brand new" taxonomy in the domain of Artificial Intelligence.

In Section 2 we describe our taxonomy induction algorithm. In Section 3 we present our experiments, and the performance results. Evaluation is both qualitative (on a new Artificial Intelligence taxonomy) and quantitative (on the reconstruction of WordNet sub-hierarchies). Section 4 is dedicated to concluding remarks.

2 Graph-based Taxonomy Induction

Our objective is to produce a domain taxonomy in the form of a directed graph. We start from an initially-empty directed graph $G = (V, E)$, where $V = E = \emptyset$. Our approach to graph-based taxonomy induction consists of four steps, detailed hereafter.

2.1 Terminology Extraction

Domain terms are the building blocks of a taxonomy. While relevant terms for the domain could be selected manually, 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. To this end, we used our term extraction tool, TermExtractor[3] [Sclano and Velardi, 2007]. Note that any equally valid term extraction tool can be applied in this step. As a result, a domain terminology $T^{(0)}$ is produced which includes both single-word and multi-word expressions. We add to our graph G one node for each term in $T^{(0)}$, i.e., $V := V \cup T^{(0)}$.

The aim of our taxonomy induction algorithm is to learn a hypernymy graph by means of several iterations, starting from $T^{(0)}$ and stopping at very general concepts. We define the latter as a small set of upper terms U (e.g., *object*, *abstraction*, etc.), that we consider as the end point of our algorithm.

2.2 Definition and Hypernym Extraction

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 further 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], that is, domain-independent machine-learned classifiers that identify definition sentences for the given term t, together with the corresponding hypernym – i.e., lexical generalization – in each sentence. An example of a lattice classification model is shown in Figure 1. The following sentences are examples of definitional patterns that can be retrieved using the lattice in Figure 1 (we use bold for the terms being defined and italics for the extracted hypernyms):

- **computer science** is a branch of *engineering science*.
- **artificial intelligence** is a prominent branch of *computer science* that...

For each term in our set $T^{(i)}$, we extract definition candidates from the domain corpus and from Web documents by harvesting all the sentences that contain t. We also add definitions from the Web obtained using the Google `define` keyword. Finally, we apply WCLs and collect all those sentences that are classified as definitional.

2.3 Domain Filtering

Given a term t, the common case is that several definintions are found for this term. However, many of these will not pertain to the domain of interest, especially if they are obtained from the Web or if they define ambiguous terms. To eliminate these sentences, we weigh each definition candidate $d(t)$ according to the domain terms that are contained therein using the following formula:

$$DomainWeight(d(t)) = \frac{|B_{d(t)} \cap D|}{|B_{d(t)}|} \qquad (1)$$

where $B_{d(t)}$ is the bag of content words contained 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*, *spanning tree* $\}$, our domain terminology $D =$

[2] http://www.dmoz.org/

[3] http://lcl.uniroma1.it/termextractor

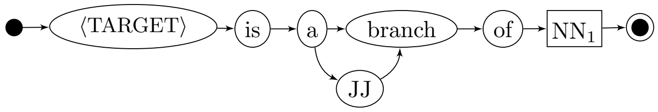

Figure 1: Lattice for the pattern "⟨TARGET⟩ is a * branch of ⟨HYPER⟩".

$T^{(0)} \cup \{$ *algorithm*, *information*, *retrieval*, *tree* $\}$. According to the above formula, the domain weight of a definition d 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 a threshold empirically set to 0.38, tuned on a dataset of 200 manually-annotated definitions. We note that 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, i.e., $V := V \cup H_t$. For each t, we also add a directed edge (h, t)[4] for each hypernym $h \in H_t$. As a result of this step, the graph contains our domain terms and their hypernyms obtained from domain-filtered definitions. We now set:

$$T^{(i+1)} := \bigcup_{t \in T^{(i)}} H_t \setminus \bigcup_{j=0}^{i} T^{(j)} \tag{2}$$

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, i.e., we perform definition/hypernym extraction and domain filtering on $T^{(i+1)}$. As a result of subsequent iterations, the initially-empty graph G is increasingly populated with new nodes (i.e., terms) and edges (i.e., hypernymy relations). After a maximum number of iterations K, we obtain a dense hypernym graph, that most likely 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 final step of graph pruning, as described in the next Section.

2.4 Taxonomy Induction

Taxonomy induction is the core of our work. As previously remarked, the graph obtained at the end of the previous step is particularly complex and large (see Section 3 for statistics concerning the experiments that we performed). Wrong nodes and edges might originate 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 algorithm that exploits the topological graph properties to produce a full-fledged taxonomy. Our algorithm consists of four steps, described hereafter with the help of the noisy graph in Figure 2(a), whose grey nodes belong to the initial terminology $T^{(0)}$ and the bold node is the only upper term.

[4]In what follows, (h, t) or $h \to t$ reads "t is-a h".

Graph trimming. We first perform two trimming steps:

i) **Eliminate "false" roots:** recursively delete each edge (v, v') such that $v \notin U$ and v has no incoming edges.

ii) **Eliminate "false" leaf nodes:** Recursively delete each edge (v, v') such that $v' \notin T^{(0)}$ and v' has no outgoing edges.

These steps disconnect the false root *service* and the false leaf *band* (see Figure 2(b)).

Edge weighting. The most important aspect of our algorithm is the edge weighting step. 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 induction[5]. Consider again the graph of Figure 2: in choosing the best hypernym for *biplane*, a connectivity-based measure would select *aircraft* rather than *airplane*, since the former reaches more nodes. However, in taxonomy learning, longer hypernymy paths should be preferred, e.g., *craft* → *aircraft* → *airplane* → *biplane* is better than *craft* → *aircraft* → *biplane*.

Our novel weighting policy is aimed at finding the best trade-off between path length and the connectivity of traversed nodes. It consists of 3 steps:

i) weight each node v by the number of nodes belonging to $T^{(0)}$ that can be reached from v (possibly including v itself).[6] Let $w(v)$ denote the weight of v (e.g., in Figure 2(b), node *aircraft* reaches *airplane* and *biplane*, thus $w(\textit{aircraft}) = 2$, while $w(\textit{watercraft}) = 3$). All weights are shown in the corresponding nodes in Figure 2(b).

ii) for each node v, consider all the paths from a root $r \in U$ 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, i.e., $\omega(p) = \sum_{v' \in p} w(v')$.

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

$$w(h, v) = \max_{r \in U} \max_{p \in \Gamma(r,h)} \omega(p) \tag{3}$$

This formula assigns to edge (h, v) the value $\omega(p)$ of the highest-weighting path p from h to any root in U. For example, in Figure 2(b), $w(\textit{airplane}) = 2$, $w(\textit{aircraft}) = 2$, $w(\textit{craft}) = 5$. Therefore, the set of paths $\Gamma(\textit{craft}, \textit{airplane}) = \{$ *craft* → *airplane*, *craft* → *aircraft* → *airplane* $\}$, whose weights are 7 ($w(\textit{craft}) + w(\textit{airplane})$) and 9 ($w(\textit{craft}) + w(\textit{aircraft}) + w(\textit{airplane})$), respectively. Hence, according to Formula 3, $w(\textit{airplane}, \textit{biplane}) = 9$. We show all edge weights in Figure 2(b).

[5]As also remarked by Kozareva and Hovy [2010], who experimented with in-degree.

[6]Nodes in a cycle are visited only once.

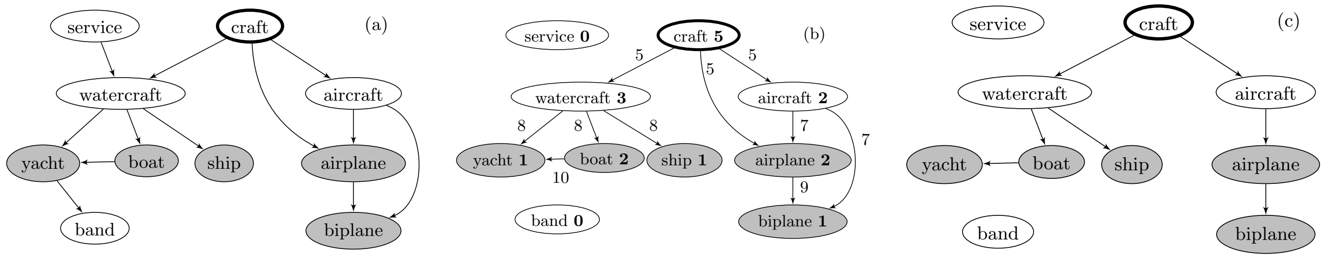

Figure 2: A noisy graph excerpt (a), its trimmed version (b), and the final taxonomy resulting from pruning (c).

Finding the optimal branching. This step aims at producing a taxonomy by pruning the noisy graph on the basis of our edge weighting strategy. A maximum spanning tree algorithm cannot be applied, 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 apply Chu-Liu/Edmonds's algorithm [Edmonds, 1967] to our directed weighted graph G to find an optimal branching. The initial step of the algorithm is to select, for each edge, the maximum-weight incoming edge. Next, it recursively breaks cycles with the following idea: nodes in a cycle are collapsed into a pseudo-node and the maximum-weight edge entering the pseudo-node is selected to replace the other incoming edge in the cycle. During backtracking, pseudo-nodes are expanded into an acyclic directed graph, that is, our final taxonomy. The resulting taxonomy for our example is shown in Figure 2(c).

Pruning recovery. The weighted directed graph from which we induce our taxonomy might contain many weakly connected components. In this case, an optimal branching is found for each component. Moreover, the number of taxonomy components could be increased as a result of Edmond's breaking-cycle strategy, thus losing relevant taxonomic relations. Most of these components are actually noise, but some of them could be included in the final taxonomy. Let r be the root of one of these disconnected components. To recover from excessive pruning, we apply a simple heuristic: first, if there existed at least one edge pointing to r in the original noisy graph, we select the best-ranking edge (v, r) according to the domain score of the definition from which this taxonomic relation was extracted; else, if in the final taxonomy there exists a node v which is the maximal left substring of r (e.g., r=*combat ship* and v=*ship*), then we add edge (v, r).

3 Evaluation

Taxonomy evaluation is a hard task which is difficult even for humans. One reason for this is that different taxonomies might model the domain of interest equally well. Nonetheless, various different evaluation methods have been proposed in the literature to assess the quality of a taxonomy. These include: i) manual evaluation performed by domain experts, ii) structural evaluation of the taxonomy, iii) automatic evaluation against a gold standard, iv) application-driven evaluation, in which a taxonomy is assessed on the basis of the improvement its use generates within an application. In our experiments, we take advantage of the first 3 evaluation strategies. To this end, we performed two different experiments: the first aimed at inducing a brand-new taxonomy of Artificial Intelligence (AI), the second at making a gold-standard comparison with WordNet sub-hierarchies. We describe these experiments in the following subsections.

3.1 Experiment 1: Inducing an AI Taxonomy

Setup. Our first experiment was aimed at inducing a full-fledged taxonomy of AI. To extract the domain terminology, we created a corpus consisting of the entire IJCAI 2009 proceedings (334 papers, overall). The same corpus was used to extract definitions for the domain terms. We collected additional definitions by querying Google with the `define` keyword. Finally, we manually selected a set of 13 upper terms U (such as *process*, *abstraction*, *algorithm*) used as a stopping criterion for our iterative definition/hypernym extraction and filtering procedure (cf. Sections 2.1 and 2.2).

Results. As a result of terminology extraction, we obtained 374 initial domain terms (our $T^{(0)}$, cf. Section 2.1). The resulting noisy graph included 715 nodes and 1025 edges. After applying Edmond's algorithm and pruning recovery (cf. Section 2.4), our taxonomy contained 427 nodes (of which 261 were initial terms, while for the remaining 113 initial terms no definition could be found[7]) and 426 edges.[8]

First, in order to study the structural effect of our taxonomy induction algorithm, we determined the compression ratio of the resulting graph against the unpruned graph: the node and edge compression ratios were 0.60 (427/715) and 0.41 (426/1025), respectively. In Figure 3 we show the compression effect of pruning (on the right) over the noisy graph (on the left). In Figure 4 we show an excerpt of the AI taxonomy rooted at *algorithm*. The maximum depth of the final taxonomy is 11. An example of a hypernymy path is: *abstraction* $\rightarrow$ *representation* $\rightarrow$ *model* $\rightarrow$ *model of a synchronous order machine* $\rightarrow$ *finite-state machine* $\rightarrow$ *Markov model*.

Second, we performed a manual evaluation of the whole set of edges (i.e., present in the final graph) and calculated a precision of 81.5%. Notice however that evaluating the correctness of individual edges in isolation, as we and virtually all the other works in the literature do, is not entirely appropriate. Despite the specificity of the domain, the average ambiguity of graph nodes before edge pruning was 1.4, and for

[7]This will be improved in future work by extending Web search and the domain corpus to the full archive of IJCAI proceedings.

[8]Any tree contains $|V| - 1$ edges, where V is the set of nodes.

Table 1: Results on the three WordNet sub-hierarchies: animals, plants, vehicles.

	animals		plants		vehicles	
node compression ratio	0.48	978/2015	0.40	744/1840	0.41	144/353
edge compression ratio	0.49	977/1975	0.35	743/2138	0.35	143/413
coverage of initial terminology	0.71	484/684	0.79	438/554	0.73	85/117
precision by hand of edges not in WN (100 randomly chosen)	0.76	76/100	0.82	82/100	0.92	92/100
precision by hand of nodes not in WN (all)	0.70	218/312	0.77	158/206	0.69	9/13

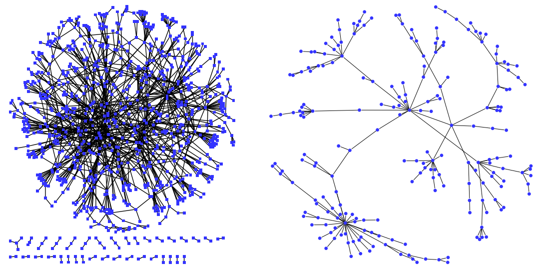

Figure 3: The effect of graph pruning (right) on the AI hypernym graph (left).

some node over 10 hypernyms were found. For example, we found for the term *discriminant analysis* the following hypernyms: *field*, *procedure*, *tool*, *technique*. The taxonomy induction algorithm selects *procedure*, based on the connectivity properties of the graph, and this is an appropriate choice, but, by looking at the overall structure of the taxonomy, and the collocation of other similar concepts (e.g., *logistic regression*, *constraint propagation*, etc.), *technique* would probably have been a better choice. However such structural evaluation should be performed by a community of experts. For the moment, the Artificial Intelligence taxonomy[9] has been released to the scientific community and will be further analysed and enriched in a future collaborative experiment.

3.2 Experiment 2: Evaluation against WordNet

Setup. In our second experiment we performed an automatic evaluation against a gold standard, that is, WordNet. This is a commonly adopted standard in the literature. Kozareva and Hovy [2010] show very high values of precision (0.98) and varying recall (0.60-0.38) in reproducing the animals, plants and vehicles WordNet sub-hierarchies. However, in computing the precision and recall of extracted relations, they consider only node pairs found by the system and in WordNet.

Similarly, in [Yang and Callan, 2009] the F-measure varies from 0.82 (on WordNet "is-a" sub-hierarchies) to 0.61 (on WordNet "part-of" sub-hierarchies). But they only compute the number of correctly retrieved relations, given the set of concepts to be related by a given relation, since the authors are concerned with finding relations, not concepts. Snow *et al.* [2006] evaluate their algorithm in a task of sense-disambiguated hyponym acquisition, starting with the base of WordNet 2.1 and adding novel noun hyponyms. For the best-performing inferred taxonomy (with 30,000 new hyponyms), they report achieving a 0.58 precision and 0.21 recall.

[9]Available at: `http://lcl.uniroma1.it/taxolearn`

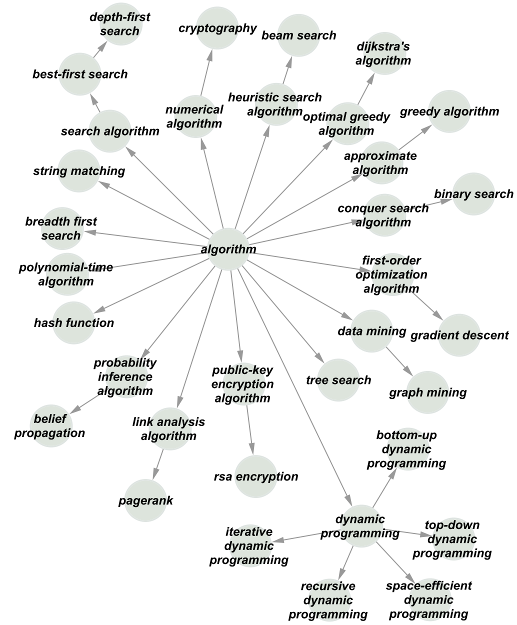

Figure 4: An excerpt of the AI taxonomy rooted at *algorithm*.

As is the case with the majority of experiments in this domain, the objective of our second experiment was to learn taxonomies for specific domains and compare them with existing sub-hierarchies rooted at specific synsets in WordNet. In order to enable a comparison with previous work, we selected the same domains as Kozareva and Hovy [2010], namely: animals, plants and vehicles.

Unlike the setup of our previous experiment, we did not perform any terminology extraction because we used the initial terminology provided by Kozareva and Hovy [2010] for the three domains. Concerning definition and hypernym extraction, we did not use any domain corpus, while we retained the use of Google `define`. Given that each of the three taxonomies is rooted at a specific synset in WordNet (animal#n#1, plant#n#2, vehicle#n#1, respectively), we used the synonyms of the corresponding root synset as the set of upper terms for that domain (e.g., { plant, flora, plant life } for the plants domain).

Results. We show the results for the three domains in Table 1. The first 3 rows provide some general statistics concerning the taxonomy induction experiment, such as node and edge compression and coverage. Coverage is defined as the number of terms in the initial terminology which survive the pruning phase. The animal, plant and vehicle domains are progres-

Table 2: Precision and recall compared with K&H.

	Our approach		K&H 2010	
Domain	P	R	P	R
Animals	97.0	43.7	97.3	38.0
Plants	97.0	38.3	97.2	39.4
Vehicles	90.9	48.7	98.8	60.0

sively less ambiguous as regards terms, and progressively less structured as regards the reference taxonomy (the animal taxonomy has an average depth of 6.23, while vehicles 3.91): this explains why we obtained a much larger hypernymy graph for animals. In Table 1 rows 4 and 5 provide a manual evaluation of edges and nodes appearing in the induced taxonomy but not in WordNet: precision is quite good, with the "caveat" of Section 3.1. To determine our ability to "reconstruct" WordNet, similarly to [Kozareva and Hovy, 2010] (K&H in what follows) we first removed from our taxonomy all the nodes not included in the WordNet sub-hierarchies and then computed precision and recall on the is-a relations from the initial terminology to the root. As shown in Table 2 our performance figures are higher than K&H on animals, similar on plants and lower on vehicles, even though 1) starting from a richer terminology than that of K&H would reinforce our algorithm's choices, 2) we induce a tree-like taxonomy while K&H obtain a graph with multiple paths from each term to the root. We replicated K&H's evaluation for the sake of comparison, however a better validation procedure should compare the actual result of "blind" taxonomy (or graph) learning with the ground truth, rather than mapping the latter onto the first.

4 Conclusions

In this paper we presented the first algorithm to induce a lexical taxonomy truly from scratch using highly dense, possibly disconnected, hypernymy graphs. The algorithm performs the task of eliminating noise from the initial graph remarkably well, using a weighting scheme that accounts both for the topological properties of the graph and for some general principle of taxonomic structures. Taxonomy induction was applied to the task of creating a new Artificial Intelligence taxonomy and three plant, animal and vehicle taxonomies for gold-standard comparison against WordNet.

This paper was primarily concerned with the description of the algorithm, thus, for the sake of space, we could not present a detailed analysis of the extracted taxonomies, which we defer to a future publication. In summary, this analysis led us to conclude that errors and sub-optimal choices in graph pruning do not depend on the algorithm, but rather on the quality and amount of knowledge available in the source hypernymy graph. Future work, therefore, will be directed towards improving the hypernym harvesting phase.

Acknowledgments

The authors wish to thank Zornitsa Kozareva and Ed Hovy for providing all the necessary data for comparison, Senja Pollak and Jim McManus for their useful comments. The first author gratefully acknowledges the support of the ERC Starting Grant MultiJEDI No. 259234.

References

[Berland and Charniak, 1999] Matthew Berland and Eugene Charniak. Finding parts in very large corpora. In *Proceedings of ACL 1999*, pages 57–64, Stroudsburg, USA, 1999.

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

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

[Edmonds, 1967] J. Edmonds. Optimum branchings. *J. Res. Nat. Bur. Standard*, 71B:233–240, 1967.

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

[Girju *et al.*, 2003] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proc. of NAACL-HLT 2003*, pages 1–8, Canada, 2003.

[Hearst, 1992] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING 1992*, pages 539–545, 1992.

[Kozareva and Hovy, 2010] Zornitsa Kozareva and Eduard Hovy. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of EMNLP 2010*, pages 1110–1118, Cambridge, MA, October 2010.

[Kozareva *et al.*, 2008] Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL 2008*, pages 1048–1056, Columbus, Ohio, June 2008.

[Maedche and Staab, 2009] Alexander Maedche and Steffen Staab. Ontology learning. In *Handbook on Ontologies*, pages 245–268. Springer, 2009.

[Navigli and Velardi, 2010] Roberto Navigli and Paola Velardi. Learning Word-Class Lattices for definition and hypernym extraction. In *Proceedings of ACL 2010*, pages 1318–1327, Uppsala, Sweden, 2010.

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

[Newman, 2010] M. E. J. Newman, editor. *Networks: An Introduction*. Oxford University Pres, Oxford, UK, 2010.

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

[Perez and Mancho, 2003] Gomez A. Perez and Manzano D. Mancho. A Survey of Ontology Learning Methods and Techniques. OntoWeb Delieverable 1.5, May 2003.

[Poon and Domingos, 2010] Hoifung Poon and Pedro Domingos. Unsupervised ontology induction from text. In *Proceedings of ACL 2010*, pages 296–305, Stroudsburg, USA, 2010.

[Sclano and Velardi, 2007] F. Sclano and P. Velardi. Termextractor: a web application to learn the shared terminology of emergent web communities. In *Proc. of I-ESA 2007*, Portugal, 2007.

[Snow *et al.*, 2006] Rion Snow, Dan Jurafsky, and Andrew Ng. Semantic taxonomy induction from heterogeneous evidence. In *Proc. of COLING-ACL 2006*, pages 801–808, 2006.

[Widdows and Dorow, 2002] Dominic Widdows and Beate Dorow. A graph model for unsupervised lexical acquisition. In *Proceedings of COLING 2002*, pages 1–7, Stroudsburg, USA, 2002.

[Yang and Callan, 2009] Hui Yang and Jamie Callan. A metric-based framework for automatic taxonomy induction. In *Proc. of ACL 2009*, pages 271–279, Stroudsburg, USA, 2009.