Comparing Constraints for Taxonomic Organization

Anne Cocos[*], **Marianna Apidianaki**[*†], and **Chris Callison-Burch**[*]

[*] Department of Computer and Information Science, University of Pennsylvania
[†] LIMSI, CNRS, Université Paris-Saclay, 91403 Orsay
{acocos,marapi,ccb}@seas.upenn.edu

Abstract

Building a taxonomy from the ground up involves several sub-tasks: selecting terms to include, predicting semantic relations between terms, and selecting a subset of relational instances to keep, given constraints on the taxonomy graph. Methods for this final step – taxonomic organization – vary both in terms of the constraints they impose, and whether they enable discovery of synonymous terms. It is hard to isolate the impact of these factors on the quality of the resulting taxonomy because organization methods are rarely compared directly. In this paper, we present a head-to-head comparison of six taxonomic organization algorithms that vary with respect to their structural and transitivity constraints, and treatment of synonymy. We find that while transitive algorithms out-perform their non-transitive counterparts, the top-performing transitive algorithm is prohibitively slow for taxonomies with as few as 50 entities. We propose a simple modification to a non-transitive optimum branching algorithm to explicitly incorporate synonymy, resulting in a method that is substantially faster than the best transitive algorithm while giving complementary performance.

1 Introduction

Many words and phrases fit within a natural semantic hierarchy: a *mobile* is a type of *telephone*, which in turn is a *communications device* and an *object*. Taxonomies, which encode this knowledge, are important resources for natural language understanding systems.

There is ongoing interest in developing methods to build taxonomic resources automatically (Bordea et al., 2015, 2016). Although several widely-used general ontologies (e.g. WordNet (Miller, 1995)) and domain-specific ontologies (e.g. Unified Medical Language System (UMLS) (Bodenreider, 2004)) exist, these resources are handcrafted and therefore expensive to update or expand. Automatic taxonomy induction enables the construction of taxonomic resources at scale in new languages and domains. Further, there is evidence that it is useful to build dynamic or context-specific taxonomies extemporaneously for some applications (Do and Roth, 2010).

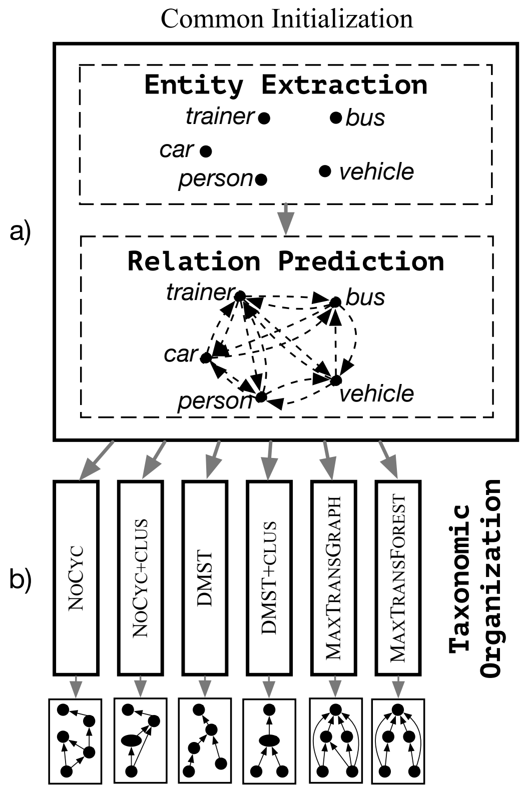


Figure 1: In this study we compare algorithms for taxonomic organization. We first (a) run entity extraction and pairwise relation prediction as a common initialization; we then (b) feed the resulting graphs as identical input to six taxonomic organization algorithms. We evaluate the impact of varied structural constraints between algorithms.

Taxonomy induction involves three sub-tasks: *entity extraction*, *relation prediction*, and *taxonomic organization*. In many cases these subtasks are undertaken sequentially to build a taxonomy from the ground up. While many works directly compare methods for relation prediction (e.g. Turney and Mohammad (2015), Shwartz et al. (2017) and others), none directly compare methods for the final taxonomic organization step with varying constraints. Each paper that proposes a taxonomic organization method starts with its own set of predicted relations, making it impossible to determine – even with benchmark datasets – the extent to which improvements in identifying ground-truth relations are due to (a) better relation prediction, or (b) better taxonomic organization.

In this work, we present an empirical apples-to-apples comparison of six algorithms for unsupervised taxonomic organization. The algorithms vary along three axes: whether they impose transitivity constraints on the taxonomic graph, whether they specify that the final graph structure be a directed acyclic graph (DAG) or tree/forest, and whether they identify 'clusters' of synonymous terms. In each case we begin with the same sets of terms and predicted relations (see Figure 1). This makes it possible to address several research questions. First, which combination of these factors produces a taxonomy that most closely mirrors a set of ground-truth taxonomic relations? Second, which algorithms are efficient enough in practice to run on large term sets? And third, how robust is each algorithm to noise in the predicted relations used as input?

We find that while transitive algorithms perform better than non-transitive algorithms given the same constraints on graph structure, the best-performing transitive algorithm is prohibitively slow to use on input with as few as 50 nodes. By modifying a commonly-used optimum branching algorithm to consolidate clusters of predicted synonyms into a single graph node, we show that it is possible to achieve complementary performance levels with an average runtime that is faster by orders of magnitude.

2 General Framework for Taxonomy Induction

The problem of taxonomy induction can be summarized via three core sub-tasks. While all systems that build taxonomies automatically must address each of these tasks, the sequence and manner in which they are addressed varies. In the most straightforward case, the core tasks are viewed as orthogonal and carried out sequentially. They are:

1. **Entity Extraction:** Identify a set of entities E (i.e. word types, synsets, etc) that will become nodes in the eventual taxonomy graph.

2. **Relation Prediction:** Predict the presence or absence of a directed semantic relation (hypernymy or entailment) between each pair of nodes, $(e_i, e_j) \in E \times E$. The outputs are (a) a set of potential edges $R \subseteq E \times E$, where we use the notation $r_{ij} \in R$ to signify the relational instance, or edge, (e_i, e_j), and (b) relation scores $s(r_{ij})$ for each edge derived from the classifier's predicted likelihood that the relational instance exists.

3. **Taxonomic Organization:** Select a subset of the predicted edges, $\hat{R} \subseteq R$, that produces a high sum of scores, $\sum_{r \in \hat{R}} s(r_{ij})$, subject to structural constraints. The final output is the graph $\hat{G}(E, \hat{R})$.

Structural constraints dictate what can be considered a *valid* or *invalid* combination of edges in a taxonomic graph (Do and Roth, 2010). Two structural constraints frequently imposed are that the final graph be a DAG, or that the final graph be a tree/forest.[1] Examples of algorithms that produce DAG structures are the longest-path algorithm of Kozareva and Hovy (2010), the *ContrastMedium* approach of Faralli et al. (2017), and the random cycle-breaking method used in (Panchenko et al., 2016) and Faralli et al. (2015). We experiment with a variation of the last one here, which we call NoCyc. To produce tree-structured taxonomies, most researchers (including us) use algorithms for finding the maximally-weighted rooted tree spanning a directed graph (DMST). Examples of prior work following this approach are Navigli et al. (2011) and Bansal et al. (2014).

Another dimension along which taxonomy organization approaches differ is whether they explicitly require the set of chosen relational instances $\hat{R}$ to be fully transitive. The transitivity constraint dictates that if (*beetle* IS-A *insect*) is selected as part of $\hat{R}$, and (*insect* IS-A *organism*) is

[1] WLOG, the tree and forest constraints are identical, as a dummy root node can be attached to the root of each component in a forest to produce a tree.

selected as part of $\hat{R}$, then (*beetle* IS-A *organism*) must also be selected. Two methods that impose such transitivity constraints are the MaxTransGraph and MaxTransForest methods of Berant et al. (2015), both of which we experiment with here.

A final consideration when choosing a taxonomy organization algorithm is whether the method should enable the consolidation of synonyms into a single taxonomic entity. Synonym sets, or *synsets*, are present as nodes in the WordNet graph (Miller, 1995). Potential advantages to using synonym sets, rather than individual terms, as nodes include the ability to model polysemy (*horse* means one thing when grouped with its synonym *cavalry* and another entirely when grouped with *sawhorse*), and the ability to be more precise in defining relations. A few early taxonomy induction approaches incorporated synonym clustering (e.g. Lin and Pantel (2002) and Pantel and Ravichandran (2004)). The two transitive algorithms that we analyze here, MaxTransGraph and MaxTransForest, also consolidate equivalent terms into a single node.

3 Taxonomic Organization Algorithms

The six algorithms that we compare differ along the three dimensions just described, namely, the structural constraints imposed (DAG or tree), whether transitivity is required, and whether synonyms are combined into a single taxonomy node (Figure 2). Here we provide a short description of each.

	No Transitivity	**Transitivity**
DAG	NoCyc NoCyc+clus	MaxTransGraph
Tree / Forest	DMST* DMST+clus	MaxTransForest

Figure 2: Classification of the algorithms compared in our study. The starred DMST algorithm is the only one that does not support consolidation of synonyms into clusters.

3.1 NoCyc: Organizing a DAG

The no-cycles method, which we abbreviate as NoCyc, is a simple method for constructing a **DAG** with high score from a set of predicted relational edges. It is **not transitive**.

The algorithm works as follows. From the set R of all predicted hypernym relations, we first filter out of the graph $G(E, R)$ any edges with score $s(r_{ij})$ less than a tunable threshold τ. Next, we break any cycles by finding strongly connected components (SCC) in the graph (i.e. a subset of nodes such that each node in the subset has a path to every other node in the subset), and iteratively removing the lowest-scoring edge from each SCC until all cycles are broken. This implementation is slightly different from that of Faralli et al. (2015) and Panchenko et al. (2016), where cycles were broken by removing cycle edges randomly. The search for SCCs in each iteration is linear using Tarjan's algorithm (Tarjan, 1972).

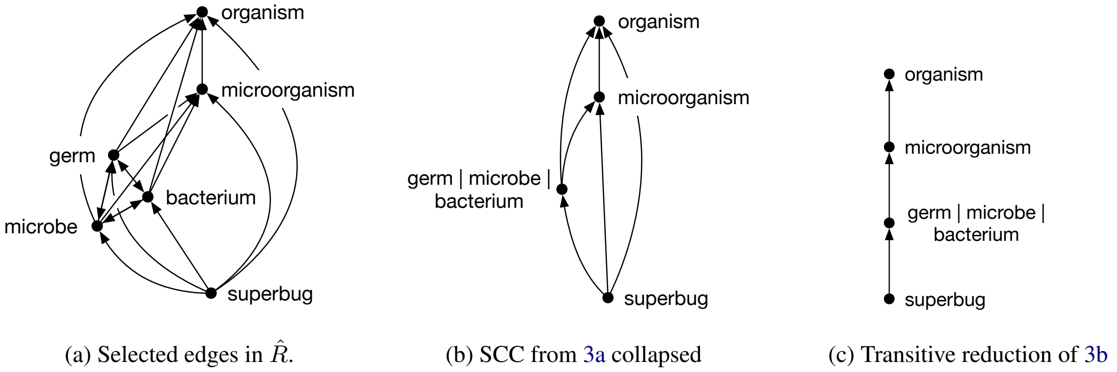
The NoCyc algorithm does not explicitly cluster synonyms, but we can find synonyms in the resulting graph implicitly as follows. If we assume all synonymous terms share the same direct hypernyms and direct hyponyms, we can find such pairs by taking the transitive reduction[2] of the resulting graph $\hat{G} = (E, \hat{R})$, and grouping all pairs of terms that have identical sets of direct hypernyms and hyponyms in the transitive reduction.

While NoCyc itself does support finding synonyms within the graph implicitly, we also experiment with an explicit synonym-clustering version, NoCyc+clus. We modify NoCyc by collapsing into a single node all subsets of nodes predicted to be synonym clusters, using a method described in Section 4.2.2, prior to executing the cycle breaking algorithm.

3.2 DMST: Organizing a Tree

Our second method selects hypernym edges for the taxonomy by using the Chu-Liu-Edmonds optimum branching algorithm (Chu and Liu, 1965; Edmonds, 1967) to solve the directed analog of the maximum spanning tree problem (DMST). It constrains the final graph to be a **tree** and is **not transitive**.

[2] In the *transitive closure* of a graph, each node e_i is directly connected by a single edge to every node e_j to which it has a path. The *transitive reduction* can be obtained for a graph G by removing all edges from G that do not change its transitive closure. The transitive reduction of a DAG is unique (Aho et al., 1972).



(a) Selected edges in $\hat{R}$.

(b) SCC from 3a collapsed

(c) Transitive reduction of 3b

Figure 3: An example transitive taxonomy graph as output by MAXTRANSGRAPH or MAXTRANSFOREST is given in 3a. Its strongly-connected component (SCC) is collapsed into a single node of synonyms to create the DAG in 3b. Finally, its transitive reduction is in 3c. Because flipping edges in the result produces a tree rooted at *organism*, the graph in 3a is called forest-reducible.

The algorithm works by adding a dummy root node e_{ROOT} to E, and an edge from e_{ROOT} to every other node e_i in the graph. We then use Chu-Liu-Edmonds to find the directed tree rooted at e_{ROOT} that spans all nodes in E and has the maximal sum of scores. Note that until now we have considered edges in taxonomy graphs to point from hyponyms to hypernyms; in this case we must switch the order, so that the spanning tree starts at the most general level of the hierarchy and reaches down to the leaves. Chu-Liu-Edmonds finds the DMST efficiently in polynomial time (Tarjan, 1977).

Because DMST requires the final graph to be a tree, there is no implicit way to find synonyms within the taxonomy graphs it generates. As with NOCYC, we test a modification called DMST+CLUS that collapses predicted synonym clusters into a single graph node prior to running the DMST algorithm (see Section 4.2.2).

3.3 MAXTRANSGRAPH: Organizing a Transitive DAG

The first **transitive** algorithm we evaluate is MAXTRANSGRAPH (Berant et al., 2012, 2015), which constrains the graph structure to be a **DAG**. MAXTRANSGRAPH was originally designed for building taxonomies of entailment relations (which can be subclassified as either synonyms or hypernyms) and is solved using integer linear programming (ILP). Rather than using classifier scores directly as input, MAXTRANSGRAPH first computes a weight between each term pair (e_i, e_j) that is equal to the classifier score minus a tunable parameter: $w_{ij} = s(r_{ij}) - \lambda$. The purpose of modifying scores this way is efficiency; MAXTRANSGRAPH solves its optimization on each connected component of the graph independently, where components are constructed by considering only positively-weighted edges in the graph. Increasing λ increases sparsity and decreases runtime.

The objective of the ILP is to maximize the weights of selected relations, while requiring that the graph respects transitivity. Berant et al. (2012) proved this problem is NP-hard and provided an ILP formulation for it as follows. Let x_{ij} be a binary variable that indicates whether edge (e_i, e_j) is in the subset of selected edges, $\hat{R}$.

$$\begin{aligned} \max_{x} \quad & \sum_{i \neq j} w_{ij} x_{ij} \\ \text{s.t.} \quad & \forall e_i, e_j, e_k \in E,\ x_{ij} + x_{jk} - x_{ik} \leq 1 \\ & \forall e_i, e_j \in E,\ x_{ij} \in \{0, 1\} \end{aligned} \tag{1}$$

The objective maximizes the sum of edge weights where the edge is 'turned on' (i.e. $x_{ij} = 1$). The first constraint enforces transitivity, i.e. for every triple of nodes (e_i, e_j, e_k), if edge $(e_i, e_j) \in \hat{R}$ and edge $(e_j, e_k) \in \hat{R}$, then edge $(e_i, e_k) \in \hat{R}$. The second constraint specifies that all x_{ij} are binary. The number of variables is $O(|E|^2)$ and number of constraints is $O(|E|^3)$.

MAXTRANSGRAPH assumes that cycles of entailment relations in the resulting graph $G(E, \hat{R})$ comprise cycles of synonyms, and that the remaining edges which are not part of a cycle are hypernym edges. Because the resulting graph must be transitive, all cycles of three or more nodes are cliques, in which each node is directly connected to every other. Once every SCC is collapsed into a single synonym cluster node, the transitive reduction of the resulting graph is a DAG (Figure 3b).

3.4 MaxTransForest: Organizing a Transitive Forest

The final algorithm we evaluate is MaxTransForest (Berant et al., 2012, 2015), which like MaxTransGraph is **transitive**, but produces a **forest/tree** structure.

MaxTransForest is nearly identical to MaxTransGraph, with the addition of one constraint that imposes its forest structure. More specifically, the graphs produced by MaxTransForest are *forest reducible*. A forest reducible graph is one where, after collapsing every SCC into a single node, the transitive reduction of the result is a forest (see Figure 3).

In practice, the forest reducibility constraint is enforced by applying one additional constraint to the ILP in Equation 1:

$$\forall e_i, e_j, e_k \in E \;\; x_{ij} + x_{ik} - x_{jk} - x_{kj} \leq 1 \quad (2)$$

This constraint says that each node e_i can have only a single parent. If relations r_{ij} and r_{ik} are in $\hat{R}$, then either e_j is the parent of e_k or vice versa; e_i may not have two parents that are not related via a hypernym relationship. Like MaxTransGraph, the number of variables is $O(|E|^2)$ and number of constraints is $O(|E|^3)$. Also like MaxTransGraph, cycles in the resulting graph are assumed to constitute clusters of synonymous terms.

4 Experimental Setup

In order to directly compare the organization algorithms described, we organize our experiments as follows. We first run entity extraction (Section 4.1) and relation prediction (Section 4.2) as a common initialization for all algorithms. Then, we take the edge scores output by the relation prediction step and feed them to each taxonomic organization algorithm (Section 4.3). Finally, we compare the output from each algorithm. Here we describe the initialization steps in more detail.

4.1 Entity Extraction

We extract sets of entities from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015). Our goal is to construct *local taxonomies*, where each entity set E consists of terms sharing a common target paraphrase. For example, a local taxonomy centered around the target *coach* might contain entities *bus*, *vehicle*, *trainer*, *person*, *car*, and *railcar*. The local taxonomy for a target word does not contain the target word itself.

We build a dataset for constructing local taxonomies centered around 50 target nouns drawn from the 2010 SemEval word sense induction dataset (Manandhar et al., 2010). For each target noun, we extract as taxonomy terms the set of PPDB paraphrases having a PPDB2.0Score of at least 2.0 with the target.[3] The number of entities in each local taxonomy ranges from 13 to 126, with a median of 40 entities per set. We hold out 5 local taxonomies to tune parameters for NoCyc, MaxTransGraph, and MaxTransForest, and use the remaining 45 as our test set.

Because they consist of related terms centered around a common paraphrase, there are several semantic relations present among these entity sets in addition to hypernymy and synonymy. We analyze the overlap between all pairs of terms appearing in our local taxonomies and in WordNet, and find that the distribution of relation types among the overlapping pairs is 6.0% hypernym/hyponym, 1.3% synonym, 0.1% meronym/holonym, 3.1% coordinate terms (sharing a common direct hypernym), and 89.5% none of these.

4.2 Relation Prediction

Having extracted a set of entities, the next step in our initialization process is to make pairwise relation predictions for each pair of terms (e_i, e_j) that exist within an entity set E. The different organization algorithms we compare take predicted synonym and/or hypernym edge scores as input. Here we describe the methods we use to generate these scores.

4.2.1 Hypernym Prediction

For hypernym prediction, we adopt the state-of-the-art HypeNET method of Shwartz et al. (2016). HypeNET integrates distributional (Lin and Pantel, 2002; Roller et al., 2014; Levy et al., 2015; Benotto, 2015) and path-based (Hearst, 1992; Snow et al., 2004; Nakashole et al., 2012) ap-

[3] The PPDB2.0Score is a supervised metric designed to correlate with human judgements of paraphrase quality (Pavlick et al., 2015).

proaches to hypernym prediction. It uses a recurrent neural network to represent the set of observed dependency paths connecting an input word pair, and concatenates this representation with distributional word embeddings to produce a set of features for predicting hypernymy.

We create a dataset of noun pairs for training and evaluating the HypeNET model. It combines noun pairs from four benchmark relation prediction datasets (BLESS (Baroni and Lenci, 2011), ROOT09 (Santus et al., 2016), EVALution (Santus et al., 2015), and K&H+N (Necsulescu et al., 2015)) with a set of related and unrelated noun pairs extracted from PPDB. Since each of these is a multi-class dataset, we binarize the data by labeling noun pairs with a hypernym relation as positive instances, and all others as negative. The combined benchmark+PPDB training set contains 76,152 noun pairs with a 1:4 hypernym:non-hypernym ratio, and the evaluation set contains 29,051 pairs. We ensure lexical separation from our taxonomy induction dataset; no terms in the classifier training set appear in any of the local taxonomies. We train HypeNET using our 76K-pair test set, and provide the results of evaluation on the 29K-pair test set in Table 1. The trained model achieves an overall average F1-score of 0.93 on the entire benchmark+PPDB test set. The full details of our dataset creation and classifier training are provided in the supplementary material.

Finally, we use the trained model to predict hypernym likelihoods for each potential edge r_{ij} in one of our local taxonomies, corresponding to an ordered pair of terms (e_i, e_j) that appear together in one of the 50 entity sets. We assign a hypernym score $s_h(r_{ij})$ to each potential directed edge that equals the HypeNET predicted likelihood for that pair of terms.

4.2.2 Synonym Prediction and Cluster Formation

We predict synonymy between noun pairs using distributional similarity, operationalized as the cosine similarity of PARAGRAM (Wieting et al., 2015) word embeddings.[4] We use PARAGRAM vectors because they perform well in semantic similarity tasks, and because they were originally extracted from PPDB and thus have 100% coverage of our entity sets. The synonym score $s_s(r_{ij})$ for a potential edge r_{ij} between entities (e_i, e_j) is simply the cosine similarity of their PARAGRAM embeddings.

Dataset	# Test Inst.	% Hyp.	Avg F1 (hyp.)	% Syn.	Avg F1 (syn.)
PPDB	3,000	20.1	.777	24.7	.707
BLESS	6,637	5.3	.978	0	–
EVALution	1,846	24.5	.763	15.1	.797
K&H+N	14,377	7.3	.988	0	–
ROOT09	3,191	26.3	.808	0	–

Table 1: Evaluation of the HypeNET hypernym classifier and the PARAGRAM synonym classifier on the PPDB test set and four benchmark test sets. We report micro-averaged F1-scores for positive and negative instances in the test sets.

We also tune a synonymy threshold for the purpose of consolidating clusters of synonymous terms into a single node for DMST+CLUS and NoCyc+CLUS (see Section 4.3). We tune threshold $\tau = 0.76$ over the benchmark+PPDB training set (binarized for synonymy) such that we predict a term pair (e_i, e_j) to be synonymous if $s_s(r_{ij}) \geq \tau$. When evaluated over the test sets, this method achieves weighted average F1-scores of 0.707 and 0.797 for predicting synonyms in the PPDB and EVALution test sets respectively (Table 1).

Target word	Clustered Entities
field	[(topic, issue, subject matter), (respect, regard), (battlefield, battleground), (outside, exterior), (territory, land), (domain, purview, sphere, ambit, realm, area, fields)]
address	[(directorate, direction), (administration, management), (answer, response), (discourse, speech), (treat, handling), (domicile, residence)]
innovation	[(novelty, imagination, creativity, newness), (modernization, modernisation), (regeneration, renewal, renovation, rejuvenation)]

Table 2: Examples of clustered entities produced using the PARAGRAM vector cosine similarity threshold of 0.76.

4.3 Input to organization algorithms

Finally, we use the calculated hypernym and synonym scores $s_h(r_{ij})$ and $s_s(r_{ij})$ to initialize each organization algorithm as follows.

NoCyc and DMST: We use the hypernym

[4] We also tried using HypeNET to predict synonym relations, but results were significantly worse.

	Algorithm	Constraint	Hypernyms			Synonyms			Combined		
			P	R	F	P	R	F	P	R	F
Baseline Methods											
	Random	(none)	.036	.235	.061	.013	.034	.018	.033	.173	.054
	MaxTransForest**	Tree/Forest	.214	.758	.325	.707	.585	.586	.255	.708	.366
	DMST**	Tree/Forest	.411	.661	.470	0.	0.	0.	.411	.469	.418
Basic Methods											
Transitive	MaxTransGraph	DAG	.123	.529	.193	.727	.028	.040	.126	.375	.182
	MaxTransForest	Tree/Forest	.147	.473	.217	.353	.091	.121	.155	.365	.210
Non-Transitive	NoCyc	DAG	.104	.596	.172	.119	.013	.014	.101	.415	.158
	DMST	Tree/Forest	.192	.195	.178	0.0	0.0	0.0	.192	.131	.147
Clustering Variations											
Non-Transitive	NoCyc+clus	DAG	.081	.562	.138	.232	.368	.234	.091	.520	.149
	DMST+clus	Tree/Forest	.165	.204	.168	.304	.364	.266	.199	.265	.201

Table 3: Precision, recall, and F1 of hypernym, synonym, and all (relation-specific) edges for each method. Metrics are weighted averages over the 45 local taxonomies in the test set, where each taxonomy's result is weighted by its number of nodes. Starred methods indicate an oracle, where the weight of edges appearing in WordNet is set to 1 at input.

scores as input, setting $s(r_{ij}) = s_h(r_{ij})$ for all potential edges.

NoCyc+clus and DMST+clus: Initialization for these algorithms requires two steps. First, we collapse clusters of likely synonyms into a single entity as follows. For each local taxonomy, we create a graph with the extracted terms as nodes, and add an edge between every pair of terms having $s_s(r_{ij}) \geq \tau$ (the threshold tuned on our training set). We take the resulting connected components as the final entity set E. See examples of synonyms clustered by this method in Table 2.

Next, we calculate scores $s(r_{ij})$ for each pair of entities. When e_i and e_j are single-term entities (i.e. not synonym clusters), we simply set $s(r_{ij}) = s_h(r_{ij})$. To obtain an edge score when one or both nodes is a cluster, we simply calculate the average hypernym score over every pair of terms (t_m, t_n) such that $t_m \in e_i$ and $t_n \in e_j$:

$$s(r_{ij}) = \frac{\sum_{t_m \in e_i; t_n \in e_j} s_h(r_{mn})}{|e_i| + |e_j|}$$

MaxTransGraph and MaxTransForest: Since these algorithms are designed to use entailment relation predictions as input, we set the score of each edge to be the maximum of the synonym and hypernym scores: $s(r_{ij}) = \max(s_h(r_{ij}), s_s(r_{ij}))$. Intuitively, this reflects the idea that entailment can be sub-classified as synonymy *or* hypernymy.

5 Experiments

We conduct experiments aimed at addressing three primary research questions: (1) How does each taxonomic organization algorithm perform? In particular, how do DAG algorithms compare to tree-constrained ones, and how do transitive algorithms compare to their non-transitive counterparts? (2) Are any algorithms, particularly the ILP methods, too slow to use on large sets of terms? (3) Given that hypernym relation prediction is far from perfect, how robust is each algorithm to noise in the predicted relations?

5.1 Head-to-head Algorithm Comparison

In our first experiment, we predict PPDB local taxonomies for the 45 target nouns in our test set using each of the six algorithms after the initialization described in Section 4. In keeping with current work on this topic (Bordea et al., 2015, 2016), we evaluate the taxonomy organization algorithms' performance by calculating precision, recall, and F1-score of WordNet 3.0 hypernym and synonym edges for the 93% of PPDB taxonomy terms that are in WordNet. When evaluating hypernym edges we consider both direct and transitive hypernym edges. We report hypernym-specific scores – where the set of ground-truth edges considers just WordNet hypernyms – synonym-specific scores, and combined scores – where all WordNet hypernym and synonym edges are taken as ground truth, and a predicted edge must have the correct start node, end node, and relation type to be correct. Results are reported in Table 3. We compare the results of the six algorithms to two types of baselines. As a lower bound, we implement a random baseline where edges are selected randomly with likelihood tuned on the benchmark+PPDB training set. As an upper bound, we run 'oracle' versions of MAXTRANSFOREST and DMST where we set the score of any edge appearing in WordNet to 1.

The transitive, tree-constrained MAXTRANSFOREST algorithm achieves the best average combined F-score (0.21) over all the local taxonomies, followed closely by the non-transitive, tree-constrained clustering method DMST+CLUS (0.20). These two methods, which are the only two tree-constrained methods that incorporate synonymy, outperform all DAG-constrained methods on this dataset. While they perform similarly in terms of combined F-score, their results are complementary; MAXTRANSFOREST obtains a relatively high score on hypernym edges and lower score for synonym edges, while for DMST+CLUS the results are reversed.

In general, these results suggest that consolidating synonyms into a single node helps tree-constrained methods by improving recall of both hypernym and synonym edges (DMST vs DMST+CLUS), but the same is not true for DAG-constrained methods. To understand why, we examine the output taxonomies. The average depth of the DAG taxonomies is greater than that of the tree taxonomies. When incorrect hyponym attachments are made in a deep taxonomy, the errors in transitive hypernym links can be magnified, which is evident in the low hypernym precision of NOCYC and NOCYC+CLUS. Synonym clustering prior to NOCYC+CLUS can magnify errors further, as synonyms are dragged into the incorrect hypernym relationships (see the NOCYC+CLUS example in Figure 4, where *telephone* is dragged along with *phone* into incorrect hypernym relations with *battery* and *pile*). For the shallower tree-constrained graph outputs, finding correct synonym relations helps the overall accuracy without inducing as many incorrect hypernym relations.

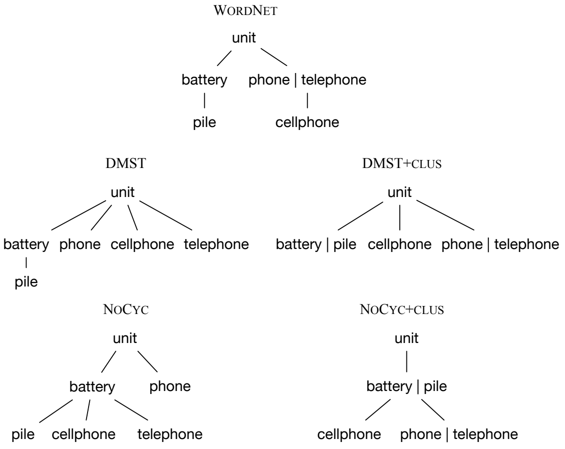


Figure 4: Examples from a portion of the local taxonomy for *cell*, with the WordNet gold standard compared to four of the algorithms' output. The bar notation denotes synonym clusters.

Finally, we note that transitive algorithms consistently out-perform their non-transitive counterparts. For the DAG-constrained algorithms, the transitive version, MAXTRANSGRAPH, improves precision of hypernym and synonym edges over its non-transitive counterparts NOCYC and NOCYC+CLUS. For the tree-constrained algorithms, the transitive MAXTRANSFOREST substantially improves recall of hypernym edges over its non-transitive counterparts DMST and DMST+CLUS.

5.2 Assessing Runtime

Next, we address the question of whether all algorithms are fast enough to be useful in practice. We record the runtime for each algorithm on each local taxonomy, and note the number of runs that timed out at 5 minutes. Results are in Table 4.

Method	Avg Runtime	% Timeout
MAXTRANSGRAPH	0.31	0.0
MAXTRANSFOREST	136.83	24.4
NOCYC	2.04	0.0
DMST	0.04	0.0
NOCYC+CLUS	6.41	0.0
DMST+CLUS	0.02	0.0

Table 4: Average runtime (seconds) over all 45 targets, and percent of targets for which runtime exceeded 5 minutes, by algorithm.

MAXTRANSFOREST, while most accurate on hypernyms and overall, is too slow to be useful on large inputs. The average runtime over all local taxonomies was over two minutes, and the runtime on local taxonomies with as few as 50 nodes reached the five minute limit. Meanwhile, DMST+clus, which performed best for synonyms and competitively for hypernyms, has a runtime that is over 6,000 times faster. In practice, this simpler algorithm may be preferable to use.

One surprising result is the speed of MaxTransGraph, which theoretically has a number of variables and constraints on the same order as that of MaxTransForest. In practice, we found that the average number of *active* constraints for MaxTransGraph – those violated at any point in the course of solving the ILP – was less than one percent of the average number of active constraints in MaxTransForest.

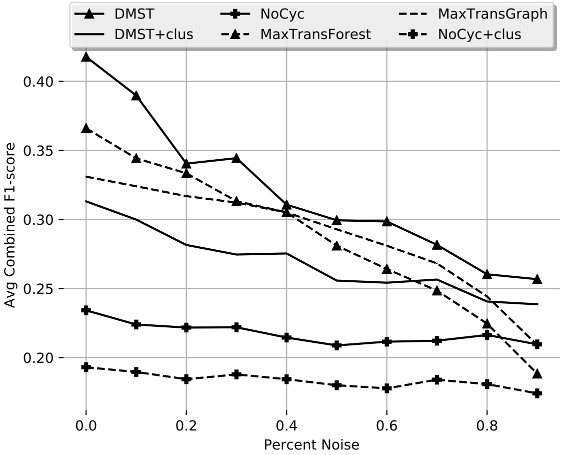


Figure 5: Weighted average combined F1-scores over the test set, where each algorithm is run in an oracle setting with noise percentage (p) settings in the range from 0% to 90%.

5.3 Assessing robustness to noise

Finally, given that hypernym prediction is still an open problem, we are interested in finding out how robust each algorithm is to noise in the input hypernym predictions. To test this, we re-run each taxonomy organization algorithm on the local taxonomies in an oracle setting, where the score of all potential edges that are present as direct or transitive edges in WordNet is set to 1. In each iteration, we set a noise probability p, and randomly perturb edge scores (according to a Gaussian distribution with 0 mean and 0.15 standard deviation) with probability p. We run this experiment with $p \in [0, 90]$. The combined F1-score is plotted against the noise level in Figure 5. We find that the performance of transitive algorithms MaxTransGraph and MaxTransForest degrades more quickly than the performance of other algorithms at higher noise levels. DMST performs best in the oracle setting at all levels of noise. The results are shown in Figure 5.

The performance of the top two performing algorithms, MaxTransForest and DMST+clus, in terms of combined F1-score degrades most with the introduction of noise. But even with up to 40% noise, these algorithms still out-perform all others.

6 Conclusion

In this paper we have conducted a direct comparison of six taxonomy organization algorithms that vary in terms of their transitivity and graph structure constraints, and their treatment of synonyms. Evaluating their performance over a dataset of local taxonomies drawn from PPDB, we find that transitive algorithms generally out-perform their non-transitive counterparts. While the best-performing algorithm – an ILP approach that constrains graphs to be transitive and tree-structured – is too slow to use on large inputs, a much simpler maximum spanning tree algorithm that consolidates synonyms into a single taxonomic node has complementary performance, with a small fraction of the runtime. Our results suggest that incorporating synonym detection into tree-constrained taxonomy organization algorithms is a promising area for future research.

Acknowledgments

This material is based in part on research sponsored by DARPA under grant number FA8750-13-2-0017 (the DEFT program) and HR0011-15-C-0115 (the LORELEI program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government. The work has also been supported by the French National Research Agency under project ANR-16-CE33-0013, and by the National Physical Science Consortium.

We are grateful to our anonymous reviewers for their thoughtful and constructive comments.

References

Alfred V. Aho, Michael R Garey, and Jeffrey D. Ullman. 1972. The transitive reduction of a directed graph. *SIAM Journal on Computing* 1(2):131–137.

Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. Baltimore, Maryland, USA, pages 1041–1051.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Edinburgh, Scotland, pages 1–10.

Giulia Benotto. 2015. *Distributional models for semantic relations: A sudy on hyponymy and antonymy*. Ph.D. thesis.

Jonathan Berant, Noga Alon, Ido Dagan, and Jacob Goldberger. 2015. Efficient global learning of entailment graphs. *Computational Linguistics* 41(2):249–291.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2012. Learning entailment relations by global graph structure optimization. *Computational Linguistics* 38(1):73–111.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research* 32(suppl_1):D267–D270.

Georgeta Bordea, Paul Buitelaar, Stefano Faralli, and Roberto Navigli. 2015. Semeval-2015 task 17: Taxonomy extraction evaluation (TExEval). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval-2015)*. Denver, Colorado, USA, pages 902–910.

Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (TExEval-2). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California, USA, pages 1081–1091.

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

Quang Xuan Do and Dan Roth. 2010. Constraints based taxonomic relation classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Cambridge, Massachusetts, USA, pages 1099–1109.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B* 71(4):233–240.

Stefano Faralli, Alexander Panchenko, Chris Biemann, and Simone Paolo Ponzetto. 2017. The ContrastMedium algorithm: taxonomy induction from noisy knowledge graphs with just a few links. In *Proceedings of the 15th Conference of the European Association for Computational Linguistics (EACL)*. Valencia, Spain, pages 590–600.

Stefano Faralli, Giovanni Stilo, and Paola Velardi. 2015. Large scale homophily analysis in twitter using a twixonomy. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*. Buenos Aires, Argentina, pages 2334–2340.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Atlanta, Georgia, USA, pages 758–764.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics (COLING)-Volume 2*. Nantes, France, pages 539–545.

Zornitsa Kozareva 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)*. Cambridge, Massachusetts, USA, pages 1110–1118.

Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Denver, Colorado, USA, pages 970–976.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)-Volume 1*. Taipei, Taiwan, pages 1–7.

Suresh Manandhar, Ioannis P Klapaftis, Dmitriy Dligach, and Sameer S Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval-2010)*. Uppsala, Sweden, pages 63–68.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods* *in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Jeju Island, Korea, pages 1135–1145.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*. Barcelona, Spain, volume 2:5.1, pages 1872–1877.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM)*. Denver, Colorado, USA, pages 182–192.

Alexander Panchenko, Stefano Faralli, Eugen Ruppert, Steffen Remus, Hubert Naets, Cédrick Fairon, Simone Paolo Ponzetto, and Chris Biemann. 2016. TAXI at SemEval-2016 Task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling. In *10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California, USA, pages 1211–1218.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*. Boston, Massachusetts, USA, pages 321–328.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevich, and Chris Callison-Burch Ben Van Durme. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*. Beijing, China, pages 425–430.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*. Dublin, Ireland, pages 1025–1036.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016. Nine features in a random forest to learn taxonomical semantic relations. Portoroz, Slovenia, pages 4557–4564.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications*. Beijing, China, pages 64–69.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 2389–2398.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Valencia, Spain, pages 65–75.

Rion Snow, Daniel Jurafsky, Andrew Y Ng, et al. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS)*. Vancouver, Canada, volume 17, pages 1297–1304.

Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing* 1(2):146–160.

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

Peter D. Turney and Saif Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering* 21:437–476.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics* 3:345–358.