Graph-Based Ontology Construction from Heterogenous Evidences

Christoph Böhm[1], Philip Groth[2], and Ulf Leser[2]

[1] Hasso-Plattner-Institut, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
christoph.boehm@hpi.uni-potsdam.de
[2] Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany
{groth,leser}@informatik.hu-berlin.de

Abstract. Ontologies are tools for describing and structuring knowledge, with many applications in searching and analyzing complex knowledge bases. Since building them manually is a costly process, there are various approaches for bootstrapping ontologies automatically through the analysis of appropriate documents. Such an analysis needs to find the concepts and the relationships that should form the ontology. However, since relationship extraction methods are imprecise and cannot homogeneously cover all concepts, the initial set of relationships is usually inconsistent and rather imbalanced - a problem which, to the best of our knowledge, was mostly ignored so far. In this paper, we define the problem of extracting a consistent as well as properly structured ontology from a set of inconsistent and heterogeneous relationships. Moreover, we propose and compare three graph-based methods for solving the ontology extraction problem. We extract relationships from a large-scale data set of more than 325K documents and evaluate our methods against a gold standard ontology comprising more than 12K relationships. Our study shows that an algorithm based on a modified formulation of the dominating set problem outperforms greedy methods.

1 Introduction

A primary use of ontologies in information systems is the description and structuring of shared knowledge [11]. For instance, ontologies are used extensively in Life Science databases to describe properties of biomedical objects, such as the function of a gene [6] or properties of a biological sequence [4]. The most prominent such ontology, the Gene Ontology [6], consists of more than 28,000 terms describing the molecular function, biological processes, and cellular locations of genes. It is used by dozens of databases throughout the world, is constantly extended and revised, and has established itself as a major research tool in functional genomics.

Building and maintaining a high-quality ontology is costly. Creating the first version of the GO has taken more than two years and involved several people from various research groups. Its maintenance and further development since then requires a constant funding of 10-20 researchers. Due to these high costs, there have been various proposals to automatically learn ontologies from a set of domain-specific documents (ontology induction, e.g., [15]). Results of such an endeavor may either be used directly, e.g., to support semantic search in documents [12], or may serve as a basis for a subsequent manual verification and extension process (ontology bootstrapping). However, ontology induction is complex and involves a series of subproblems. First, the set of terms (concepts) that are to be included in the ontology must be determined. Second, all occurrences of those concepts in the available corpus must be found. This is not trivial, because concepts may consist of several tokens, which may or may not appear in different orders. Furthermore, morphological variations, abbreviations, and usage of synonymous words are common. Next, evidences for the semantic relationships between concepts must be found (relationship extraction), for instance by searching for specific grammatical patterns involving two concepts. Finally, the ontology itself must be constructed from the set of all extracted relationships, a step we call *ontology extraction*.

Our work focusses on ontology extraction. This is an important step in ontology induction, because in large-scale projects extracted relationships are often redundant, contradicting, or circular. Such cases destroy the semantic consistency of an ontology and need to be resolved. More precisely, we study the following problem: Given a set S of ISA-relationships between concepts, find a subset $S' \subset S$ such that S' is cycle-free and also fulfills certain other criteria. A natural first choice for additional criteria might be that S' has maximal size, i.e., that it is computed from S by removing the least number of edges. But this is not necessarily the best choice in ontology extraction. It is common-sense that an ontology should exhibit certain topological properties to make it better accessible to humans (see, for instance, [1, 23, 24]). These properties include:

1. the topology should roughly resemble a tree, i.e., the number of nodes with more than one parent should be low (trees are easier to grasp),
2. nodes should not also be parents of their siblings (this is often perceived as semantic nonsense),
3. nodes with only one child should be avoided (merging them would be appropriate),
4. leaves should have a comparable depth (to avoid imbalanced subparts and varying semantic granularity of leaves),
5. grossly imbalanced numbers of children should be avoided (to prevent varying semantic granularity in inner nodes),
6. nodes should not have (additional) parents that are more than one level away (which goes beyond property 2 to avoid links completely inconsistent with the tree-like ontology backbone), and
7. the ontology should have a single root node.

Consider Fig. 1 as an example. The left part shows a set of ISA-relationships between concepts a, b, c, d, and e as it could emerge from a relationship extraction phase (edges are not weighted to keep the example simple). The set contains cycles and is thus inconsistent. There are various ways to break the cycles by removing one or more edges. Some resulting (consistent) ontologies are shown in the second to fifth part of the figure. Option 1 requires only one edge to be removed, while Options 2, 3, and 4 require two edge removals. Most of the options have some property that might be considered unfortunate for a 'clean' ontology. For instance, in Option 1, node b is sibling and parent of node d, in Option 3, the depth of the leaves varies greatly, and Options 3 and 4 contain many nodes with only one child.

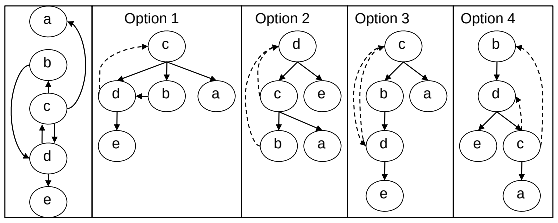

Fig. 1. Extracting a consistent ontology from a set of inconsistent relationships. The left part shows concepts and extracted (cyclic) relationships. The second to fifth part depict different ontologies, which can be derived. Option 1: removal of edge $d \rightarrow c$; 2: $c \rightarrow d, b \rightarrow d$; 3: $c \rightarrow d, d \rightarrow c$; 4: $c \rightarrow d, c \rightarrow b$.

Certainly one should take into account that extracted relationships often are simply false. The probably most important, yet also most difficult to evaluate, criterium for S' is whether it contains a maximal number of true relationships and a minimal number of false relationships. Finally, it is important that relationship evidences usually carry method-specific confidence values. These values must be considered when choosing the relationships to form the ontology.

In this work, we study the problem of constructing a semantically consistent, correct and well-formed ontology from a given set of heterogeneous, weighted, and possibly inconsistent evidences. For this, we use two different methods for extracting ISA-relationships among concepts from a domain-specific corpus resulting in a set of $> 29K$ relationships. We propose three graph-based algorithms for selecting a semantically consistent subset of relationships from this set. We evaluate our methods by trying to (re-)construct a phenotype ontology (a phenotype is, broadly speaking, an observable property of an organism attributed to the (mal-)function of a gene [8]), using as corpus a set of 327,200 phenotype descriptions downloaded from PhenomicDB [7] as of 02/2008. We converted the texts to lower case, removed URIs, punctuation, and tokens consisting only of numbers. For evaluation we leverage as our gold standard the Mammalian Phenotype Ontology (MPO) [22], consisting of 11,700 concepts (plus synonyms) and 6,830 direct ISA-relationships (03/2008). For the evaluation we use all transitive relationships from the MPO; we consider two concepts a and b to be in transitive relationship if there is a path from a to b or a and b are synonyms. MPO contains 172.134 transitive relationships.

2 Related Work

Ontology induction in general is well studied (e.g. [25]). Systems covering the entire process are, e.g., OntoEdit [15] or OntoLearn [16]. However, to the best of our knowledge, the problem of ontology extraction with inconsistent evidences has achieved little attention in the scientific literature so far, probably due to the fact that most studies in ontology construction are rather small-scale and therefore do not face this problem on a notable scale (e.g., [3, 9]).

We are aware of only few papers that explicitly deal with this problem. Schmitz reported on a study for creating term hierarchies from *flickr* tags [21]. He used subsumption for relationship extraction (see below) and suggested filtering techniques for the resulting relationships, such as a required minimum occurrence of tags. The paper also proposed a pruning strategy that eliminates all relationships that would form relations within the same hierarchy level. This approach is not comparable to ours, because no global consistency of the resulting ontology is targeted or guaranteed and structural properties are not considered. Krishna and Krishnapuram presented a clustering-based approach to hierarchy construction in [13]. However, this algorithm is used to cluster documents for improved browsing and works on entire documents, not extracted relations. This paper also mentions desirable properties for concept hierarchies, which are more usage-oriented than ours. We refrained from reusing these properties, because their formulations is vague and an evaluation is only possible through usage observation. [14] described a method for constructing a concept hierarchy based on a probabilistic language model derived from term co-occurrences. The authors acknowledge the existence of inconsistent relationships and describe a pruning strategy based on the Dominating Set Problem (DSP). This paper largely influenced our work; however, we go beyond their proposal by refining the DSP approach and by comparing it to two other strategies; furthermore, we provide a thorough, large-scale evaluation of both methods, which is lacking in [14] where only 500 texts were used as input (compared to 327,200 in this work).

In sharp contrast to ontology extraction, relationship extraction has been researched extensively over the last decades. In this work, we use Hearst-style POS pattern [10] and subsumption [20], because of their simplicity and expected coverage. Furthermore, we wanted to evaluate how our methods for ontology extraction deal with heterogeneous sources of evidences. Using multiple sources of evidence for relationship extraction is not a new idea; for instance, [3] combined Hearst-style patterns, WordNet relations, head-modifier properties, and term co-occurrence. Note that this work has a much smaller scale than our study and does not cover ontology extraction.

3 Concept Occurrences and Relationship Extraction

In this work, we consider the set of concepts predefined by the gold standard ontology, i.e., MPO, to allow a comparative evaluation. However, we still need to spot occurrences of such concepts in the corpus, which we discuss first. We then show how we extract weighted relationships between those concepts.

3.1 Concept Occurrences

We need to locate all occurrences of all concepts in our corpus. Note that this step is non-trivial for biomedical ontologies, especially because concepts often consist of multiple tokens, which only rarely appear as such in a text. For instance, in the MPO the average number of tokens per concept is 3.5 and only 5.5% of the concepts consist of only one token. Thus, exact matching of concepts would yield a low recall.

We apply approximative concept matching using various meanings of 'approximative'. We investigated the following options to determine whether a set of tokens in a text should be considered as a match with a concept: (1) tokens of a concept appear in a window of w consecutive tokens (integer parameter $w > 0$) in the text; (2) tokens of the concept must or need not appear in the same order in the text (boolean parameter *order*); (3) concepts and corpus are stemmed before matching (boolean parameter *stemming*); (4) stop words are removed from the concepts before matching (boolean parameter *stopwords*). Stemming is accomplished using Porter Stemmer [17]. The list of stop words was taken from [18] (320 terms). In the following, we describe the actual setting of parameters using a vector $\boldsymbol{p} = (order, stemming, stopwords)$. We searched all occurrences of 11,700 MPO concepts (incl. synonyms) using different parameter settings over a range of values for w. We do evaluate our methods only by counting the number of distinct concepts matched, because checking whether matches are actually true would be extremely time-consuming and is not in the focus of our work. We are not aware of any MPO-annotated corpus, which we could use to compute classical IR metrics.

For $w = 2...6$ we find 4,500-6,500 concepts, which is ≈50% of all MPO concepts. The highest number of different concepts is matched when token order is ignored, *stemming* is used, and no stop words are removed ($\boldsymbol{p} = (0, 1, 0)$). Although, we expect $\boldsymbol{p} = (1, 0, 0)$ to produce the least number of false positives, manual reviews have shown that $\boldsymbol{p} = (0, 1, 0)$ performs comparably. Considering the number of transitive MPO relations, which only use matched concepts, we find 35,000-61,500 MPO relations, which amounts to at most 35.5% of all transitive MPO relationships. Note that these numbers form an upper bound to all subsequent steps, since a relationship including a concept that is never found in the corpus cannot be inferred by any method.

3.2 Relationship Extraction

We use two methods for extracting relationships between concepts spotted in the previous step: *subsumption* [20] and Hearst-style *POS-patterns* [10]. Both are explained and evaluated on our corpus in the following subsections. Evaluation is carried out in terms of precision and coverage. We omit figures for recall for the following reason: Imagine a gold standard like $a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_{10}$. These nine direct relationships induce 45 transitive relationships. If a method would recover all direct relationships but $a_5 \rightarrow a_6$, it would miss 25 transitive relationships and its recall would already drop down to 45%. Therefore, we consider recall values under our evaluation scheme as somewhat misleading and instead give absolute numbers of correctly extracted relationships (coverage).

Subsumption. The underlying hypothesis for subsumption is that if a concept a always occurs close to a concept b, then a is related to b. If furthermore a occurs more often than b, then a is considered to be more general than b. This intuition is formalized as follows: Let a and b be two concepts and let $p(a|b)$ denote the relative frequency with which a occurs in a window of tokens of length v that also contains b. We say that b ISA a if $p(a|b) \geq t$ and $p(b|a) < 1$, where t is a threshold. Such a relationship is assigned the score $s_{Sub}(a, b) = p(a|b)$.

For evaluating subsumption, we took six parameters into account: v, t and the four parameters from the concept matching (w, *order*, *stemming*, *stopwords*). The impact of varying t is generally as expected: the higher t, the higher precision and the lower coverage. We found $t = 0.9$ to be a good compromise; results for varying t are omitted for brevity. Also, changing w between $2 \ldots 6$ has no significant influence on precision and coverage; we show data for $w = 5$.

Results for various other parameter settings are shown in Fig. 2. The main tendency is that increasing v causes a decrease in precision but an increase in coverage. $\boldsymbol{p} = (1, 0, 0)$ reaches the best precision $(0.75 - 0.6)$ at average coverage, while $\boldsymbol{p} = (0, 1, 0)$ yields a precision which is only 0.05 less but has a much better coverage. To keep precision high, we fixed this setting ($\boldsymbol{p} = (0, 1, 0)$, $t = 0.9$, $w = 5$) at $v = 10$ for all subsequent experiments.

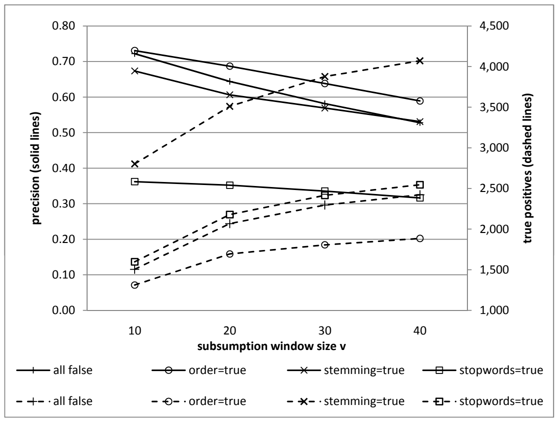

Fig. 2. Performance of subsumption for different parameter settings and $w = 5, t = 0.9$: precision (left axis, solid lines) and coverage (right axis, dashed lines).

POS-patterns. Hearst-style patterns are fixed patterns of word forms or word types that hint to a certain relationship. Consider the following sentence: "*The occurrence of cleft lip and palate in association with skeletal changes such as absent radius suggests Roberts syndrome.*" From the key phrase "*such as*" one can conclude that *absent radius* is a *skeletal change*. In this work, we used the following patterns: *[a such as b]*, *[a includes b]*, *[a especially b]*, *[a like b]* and *[a for example b]*. However, we have to pay special attention to determining meaningful borders of those a and b. We therefore perform a series of preprocessing steps on the corpus before searching the patterns. First, we run a sentence splitter [2] to exclude matches across sentences boundaries. We discard sentences that do not contain a key phrase. Finally, we used a chunker [2] to divide each sentence into syntactically correlated phrases, especially noun phrases. Chunking is important, because the concepts connected in a pattern are not always the ones that are immediately before or after the key phrase.

We considered x noun phrases before and x noun phrases after a key phrase as being connected. The precision obviously decreases for increasing x since a relation discovered by x_j is also discovered by any $x_k > x_j$ but not vice versa (of course, recall increases). We use these precision values to score extracted relationships in the following way: Let $p_1...p_i$ be the precision values (in descending order) the process achieves for $x_1...x_i$ ($x_1 \leq ... \leq x_i$). We assign scores $s_P(a,b) = \frac{p_j}{p_1}$ if $a \rightarrow b$ was discovered using x_j ($1 \leq j \leq i$) to all generated relationships.

We ran experiments with different values for $\boldsymbol{p}$ and x (at $w = 5$); see Fig. 3. Generally, precision decreases and coverage increases with increasing x. $\boldsymbol{p} = (1,0,0)$ achieves the highest precision followed by $\boldsymbol{p} = (0,1,0)$. Coverage ranges greatly, from 30 ($\boldsymbol{p} = (1,0,0), x = 1$) to 1,900 ($\boldsymbol{p} = (0,1,0), x = 5$). $\boldsymbol{p} = (0,1,0)$ reaches the highest number of true positives at a precision that does not differ significantly from the other settings for values $x > 2$. Accordingly, we used $\boldsymbol{p} = (0,1,0)$ and $x = 1 \ldots 5$ for subsequent experiments.

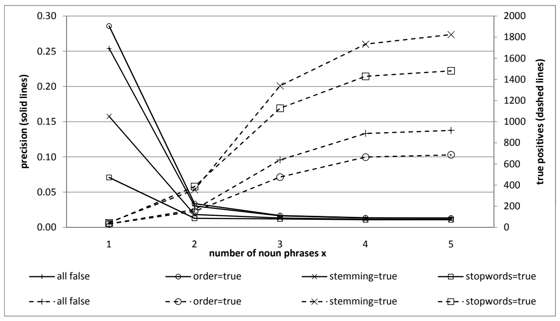

Fig. 3. Performance of POS pattern discovery for different parameter settings and $w = 5$: precision (left axis, solid lines) and coverage (right axis, dashed lines).

4 Graph-Based Ontology Extraction

The result of the previous phase are lists of relationships with a score, either resulting from subsumption or from Hearst-style patterns. This set of weighted relationships does not yet form a consistent and well-formed ontology. First, relationships often are found by both methods, which leads to redundant links; those should be merged in a way that the weights of the original links are properly considered. Second, relationships may be directly contradicting, even when we look at only one source of evidence. Third, relationships may form cycles. Both, cycles and contradicting relationships (essentially cycles of length two), destroy the semantic consistency of an ontology.

Removing these inconsistencies requires choices onto which relationships to exclude and which to include. These choices should be guided by the confidence in the respective relationships, their influence on topological properties of the resulting ontology (as discussed in the introduction), and their semantic correctness. In this section, we devise graph-based algorithms for solving this problem. All operate on a so-called concept graph, which unifies matched concepts and all extracted relationships into a single graph. Note that a nice property of this approach is that adding new sources of relationship evidence is trivial.

Definition 1. *A* concept graph *is a directed graph $G = (V, E)$, where V (nodes) is the set of concepts and E (edges) is constructed from the extracted, weighted evidences (see s_{Sub}, s_P from Sec. 3.2). Let $a, b \in V$ and $0 \leq w_S, w_P \leq 1$ be weighting factors. We define edges E,* edge weights *$s(a, b)$, and* node weights *$s(a)$ as follows:*

$$E = \{(a, b) | s_{Sub}(a, b) > 0 \vee s_P(a, b) > 0\}$$

$$s(a, b) = w_S * s_{Sub}(a, b) + w_P * s_P(a, b)$$

$$s(a) = \sum_{(a,b) \in E} s(a, b)$$

4.1 The Problem

As previously described, the concept graph itself cannot be used as an ontology (in most cases), because it is semantically inconsistent. However, any cycle-free subgraph of the concept graph could in principle be used as an ontology. Therefore, we define the problem of *ontology extraction* as follows.

Definition 2. *Let $G = (V, E)$ be a concept graph. We call a subgraph $G' = (V', E')$ of G with $V' = V$ and $E' \subseteq E$* consistent *if G' is cycle-free. The problem of* ontology extraction *is to choose a consistent subgraph of a given concept graph.*

Consider again Fig. 1 and assume this concept graph would have edge weights as given in Fig. 4. One way to further specify the ontology extraction problem is to require the extracted subgraph to be consistent and *maximal*, i.e., that the sum of the weights of extracted edges is maximal under all consistent subgraphs (by definition $V = V'$). This would result in option 4 with a total weight of 2.6. Using such a formulation would disregard many other criteria to judge good from bad ontologies. Furthermore, the corresponding optimization problem is too complex to be solved for concept graphs of non-trivial size. Actually it is already NP-hard to solve the f-DAG problem, which is to find, given a weighted digraph G and a parameter f, the heaviest DAG in G with out-degree $\leq f$ (by reduction to Minimum Feedback Arc Set [5]).

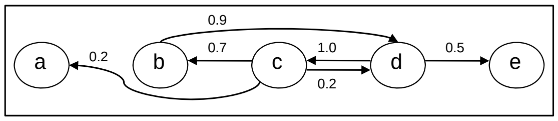

Fig. 4. Weighted Concept Graph with inconsistent relationships from Fig. 1.

In the following, we propose and compare three algorithms to solve the ontology extraction problem. The first, which we call *Greedy Edge Inclusion* (*GEI*), adds edges to an empty graph in the order of their weights. It is a heuristic for finding cycle-free graphs that disregards the specific nature of the ontology extraction problem. In contrast, the second and third solution, which we call *Hierarchical Greedy Expansion* (*HGE*) and *Weighted Dominating Set Problem* (*wDSP*), build balanced tree-like structures by iteratively adding *strong* nodes to an empty graph, i.e, nodes that have strong evidences to be a super-concept of many other nodes. They differ in the exact definition of *strong*. In the following, we explain each of these algorithms in more depth.

4.2 Greedy Edge Inclusion (GEI)

The GEI algorithm works as follows: It first copies all nodes from the concept graph into the emerging ontology graph. It then adds edges from the concept graph to the ontology graph sorted by their weights. Every edge that introduces a cycle is omitted.

The main advantage of this method is its simplicity. However, it has many disadvantages. First, it completely disregards topological properties of the resulting structure, which leads to DAGs with many nodes with many parents, grossly imbalanced numbers of children, and many edges spanning large distances (see Sec. 5). Furthermore, due to its many cycle-checks it is rather costly to be computed. Finally, it does not produce a rooted DAG (and therefore has no distinct leaves). Rooting such a structure could be approached by a topological sort , which would open further choices since a topological sort is not unique. Therefore, we evaluate the GEI algorithm only in its unrooted form.

4.3 Strong Node Sets

The following two algorithms share a common idea captured by the notion of *strong* node sets. We first explain this idea in more depth before we discuss the *HGE* and *wDSP* algorithms.

Both algorithms iteratively build an ontology. They start from an artificial root node and add, level by level, *strong* nodes from the concept graph. Therein, they frequently need to solve the following problem: Given a node from the concept graph, choose a subset of its children such that (a) the resulting ontology is consistent, (b) that the overall sum of evidences added to the graph is high, and (c) that it obeys the requirements formulated in the introduction as much as possible. We approximate these requirements by the notion of a *strong node set*. Intuitively, a *strong node set* is a set of nodes that have strong evidences to be a super-concept for many other concepts. This idea leads to the algorithmic framework shown in Listing 1.1. We use the following notations (let $G = (V, E)$):

- $strong(G, f)$ computes a *strong node set* from G with maximally f members, i.e., a set $V' \subseteq V$, $|V'| \leq f$ and all $v \in V'$ have high evidence to be super-concepts of many other nodes. The concrete definition of *strong* is different in HGE (Sec. 4.4) and wDSP (Sec. 4.5).
- $subgraph(G, S)$ is the subgraph of G induced by $S \subseteq V$.
- (n, l, D) is a triple of a node n, a level l (an integer), and a (strong) set D of nodes.

Listing 1.1. Ontology extraction from a concept graph

```
 1  $last\_level = 0$
 2  $to\_be\_removed = \{\}$
 3  $V' = \{root\}$ // resulting nodes
 4  $E' = \{\}$ // resulting edges
 5  $strong\_set = strong(G, f)$
 6  $V' = V' \cup strong\_set$
 7  for each $child \in strong\_set$
 8      $E' = E' \cup \{(root, child)\}$
 9      add $(child, 1, strong\_set)$ to $queue$
10  while ( $queue$ not empty )
11      $(n, current\_level, S)$ = next from $queue$
12      if ( $current\_level > d$ )
13          break
14      if ( $last\_level < current\_level$ )
15          $last\_level = current\_level$
16          $G = subgraph(G, V - to\_be\_removed)$
17          $to\_be\_removed = \{\}$
18      $to\_be\_removed = to\_be\_removed \cup \{n\}$
19      $subgraph\_nodes = \{c|(n, c) \in E\} - S$ // $n$'s children without $S$
20      $strong\_set = strong(subgraph(G, subgraph\_nodes), f)$
21      $V' = V' \cup strong\_set$
22      for each $child \in strong\_set$
23          $E' = E' \cup \{(n, child)\}$
24          add $(child, l + 1, strong\_set)$ to $queue$
25  return $G' = (V', E')$
```

The first run of *strong* determines nodes for the first level of abstraction (line 6). To obtain a hierarchy-like structure with a root the algorithm creates edges from an artificial root node to the first-level-nodes (line 9). In a next step, it inserts the first-level-nodes, the current level (here 1), and the current *strong* set into a queue (line 10). The queue allows for a breadth-first processing of the nodes. The current *strong* set is stored to avoid that nodes in the set are part of following *strong* sets and thereby cause a cycle. The while loop (line 11) successively processes the nodes in the queue. The algorithm exits the loop if the maximum number d of levels is extracted from the concept graph (line 13-14). The following steps in the while loop determine the descendants (in the hierarchy) of the nodes in the queue. This is achieved by applying *strong* to the children of the nodes in the queue (line 20-21). There is a set of candidate nodes that may be direct descendants of a node n in the resulting hierarchy, i.e., $subgraph\_nodes = \{c|(n, c) \in E\} - S$ (line 20). The application of *strong* to $subgraph(G, subgraph\_nodes)$ chooses a subset to form a part of the next level of abstraction (line 21). A level l is complete when all nodes from level $l-1$ have been processed and are thus removed from the queue. When a level has been completely processed (line 15) the algorithm removes the nodes from the input graph (line 16-18). To connect levels the process creates only edges from a node n to nodes in the *strong* (sub)set of its children (line 24).

Regarding the requirements stated above, this algorithm has the following properties. (a) It is consistent, because edges that cause cycles are removed actively. (b) It tries to maximize the global sum of edges by maximizing the local sum of edges emerging from each node. (c) The properties stated in Sec. 1 are considered as follows:

1. By repeating the strong node set computation in a breadth-first manner we inherently extract a hierarchy-like structure.
2. We actively remove edges between members of a chosen strong node set and neighboring strong node sets on each level.
3. Avoidance of nodes with only one child is not embedded in the algorithm; however, we shall see that both algorithms create such cases rather rarely (compared to the gold standard; see Sec. 5, Tab. 1).
4. We restrict the number of iterations and thus avoid leaves of grossly different depths.
5. We fix a global upper bound for the size of a strong node set, which leads to a relatively evenly distributed fanout.
6. We only add edges from nodes to strong (sub)set of its children and therefore remain consistent with the tree-like ontology backbone across multiple levels of abstraction.
7. The ontology has a single root by construction.

4.4 Hierarchical Greedy Expansion (HGE)

As stated above, *strong* node sets $V' \subseteq V$ ($|V'| \leq f$) contain nodes that have a strong evidence to be super-concepts for many other nodes. Therefore, a *strong* node set wants to maximize $\sum_{v \in V'} s(v)$ to gather as much evidence as possible; however, it also needs to consider the sheer number of children added to the graph, i.e., $|\bigcup_{v \in V'}\{w|v \rightarrow w \in E\}|$, because many children likely lead to many grand-children and so forth.

We implemented two ways of identifying such *strong* node sets from a concept (sub)graph. Our first approach is called Hierarchical Greedy Expansion and works as follows: For a given concept graph $G = (V, E)$ and an upper bound f it adds nodes n to a strong node set R in the order of their scores $s(n)$ (from

Def. 1) until $|R| = f$. After adding a node, it removes the added node and all its edges from G and adjusts the scores of the remaining nodes in V. Recall that the score $s(n)$ of a node n is the sum of the weights of its outgoing edges.

4.5 Weighted Dominating Set Problem (wDSP)

Our second approach models $strong(G, f)$ as a slightly modified instance of the Dominating Set Problem (DSP). Formally, the DSP is the following: Given a graph $G = (V, E)$, find a subset $D \subset V$ (the superconcepts), such that for each node $v \in V - D$ a node $d \in D$ exists, where $(d, v) \in E$, and $|D|$ is minimal. Thus, the DSP is quite similar to the problem of selecting a set of strong nodes in every iteration of our framework. This was first observed by Lawrie et al. in [14]. We reuse and refine their method in this section. Note that we need to modify the original proposal to consider weights of nodes.

Definition 3. *Let $G = (V, E)$ be a concept graph and f be an upper bound. Let $d(D) := \{v|v \in V - D \wedge \exists d \in D|(d, v) \in E\}$ denote the subset of $V - D$ dominated by $D \subset V$. The* ***weighted Dominating Set Problem (wDSP)*** *is to find a set of nodes $D \subset V$, which satisfies the following requirements:*

1. *$|D| \leq f$,*
2. *$\sum_{d \in D} s(d)$ is maximal,*
3. *$\nexists \hat{D}$ such that $\hat{D}$ satisfies (1) and (2) as well as $|d(\hat{D})| > |d(D)|$.*

However, solving DSP is NP-hard [5], and so is solving wDSP. Therefore, we use an approximation called $GreedyVote$ (adapted to wDSP), which was shown to be the best performing heuristic for solving DSP in [19]. $GreedyVote$ takes the neighborhood of a node n into account when it decides whether n should be added to a dominating set or not. It uses a score $gv(n)$ that captures for every node how 'difficult' or 'easy' it can be dominated; for details, we refer the reader to [19]. We modified the algorithm to take node weights into account. For each node, we compute a new score $score(n) = gv(n) + s(a)$ and thus include both maximization criteria, $|\bigcup_{v \in V'} \{w|v \rightarrow w \in E\}|$ as well as $\sum_{v \in V'} s(v)$.

5 Evaluation

We run all three algorithms on the set of relationships extracted from the phenotype corpus using the methods explained in Sec. 3.2. For the HGE and wDSP algorithm, we show results for varying values of f (max. size of strong node sets) and d (max. depth of ontology); all other parameters were fixed as described in Sec. 3.2. We compare all resulting ontologies to the MPO. In the following, we first concentrate on precision and coverage and then discuss differences in topological properties. Remember that we consider a relationship $a \rightarrow b$ to be true if concepts a and b are in transitive relationship in MPO, i.e., there is a path from a to b or a and b are synonyms.

We also ran experiments with different weights w_S, w_P for subsumption and pattern discovery (see Definition 1). For brevity, we do not show results here. In general, coverage and precision is better when both weights are set equally compared to using only one evidence, but the increase is small ($< 5\%$ on coverage). All further results were produced with $w_S = w_P = 0.5$. Another interesting information is the precision and coverage of the concept graph itself. Its coverage serves as an upper bound to all algorithms. We considered all relationships with $s_{Sub}(a, b) > 0.5$ or $s_P(a, b) > 0$. Then, precision is 0.47 and coverage is 5,070.

We first compare precision and coverage of HGE and wDSP for varying values of f and d. Fig. 5 gives coverage values for HGE and wDSP for increasing values of f. Additionally, it shows the fraction of true positives that are direct relations in MPO. The figure shows that $wDSP$ produces 10-25% more true positives than HGE, both, for transitive and direct relations. At the same time, precision (data not show) slightly decreases from 0.54 to 0.51 for wDSP and remains almost constant ≈ 0.5 for HGE. Though the increase of true positives slows down for increasing f, it is not bounded, i.e., the number of true positives grows monotonously. This monotony implies that better coverage values than reported here are possible, but only at the expense of less readable ontologies (many nodes with > 50 children).

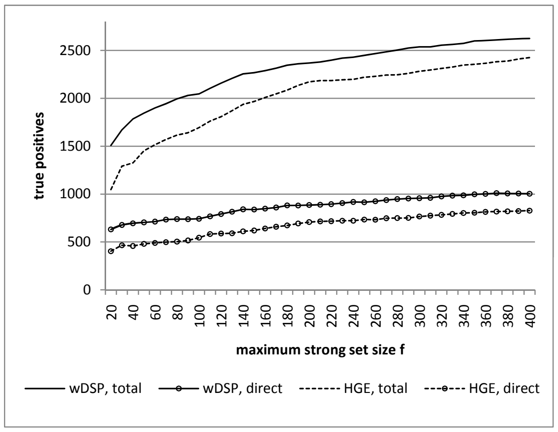

Fig. 5. Coverage of HGE and wDSP for different maximum *strong* set sizes f ($d = 10$).

Fig. 6 show coverage of HGE and wDSP for different values of d. wDSP has a steep increase until $d = 6$ and then roughly levels out, since the fluctuation for $d = 7...20$ probably can be contributed to the random factor in $GreedyVote$. In contrast, HGE's number of true positives is considerable lower but increases constantly. As in the previous experiment, precision of wDSP is better than for HGE (e.g., ≈ 0.54 versus ≈ 0.50 at $d = 2 \ldots 20$). The difference in coverage is mostly due to the fact that wDSP manages to include more relationships into the ontology than HGE without loosing precision.

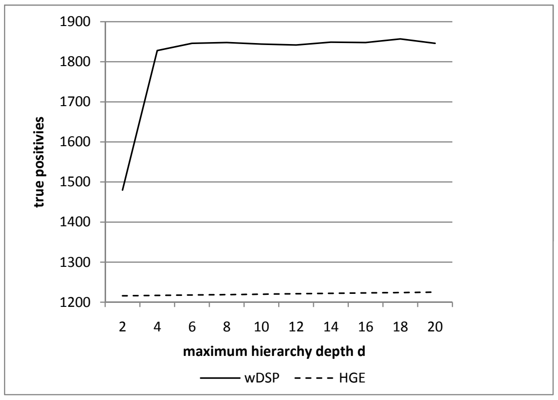

Fig. 6. Coverage of HGE and wDSP for different maximum hierarchy depths d ($f = 50$).

Compared to HGE and wDSP, the (parameter-free) GEI algorithm has a much higher coverage (4,376 true positive relationships). However, this is at the cost of a considerably lower precision (0.45) and a less well-formed topology (see below). Fig. 5 and 6 indicate that the coverage of HGE and wDSP likely could be increased by giving extremely high limits for f and d, but this would probably lead to equally twisted structures as in the case of the GEI algorithm.

Table 1 compares the performance of the GEI, HGE, and wDSP algorithm with regard to the topological properties stated in Sec. 1. We also computed those number for the gold standard itself, i.e., the MPO. GEI clearly performs the worst, since the standard deviation of both in- and out degree is much higher than for the other two algorithms (recall the GEI produces an unrooted DAG and therefore properties 4 and 6 are meaningless). The comparison between HGE and wDSP is less clear. wDSP has a much more homogeneous out-degree distribution, but a slightly worse in-degree distribution. Its ontology is deeper and has a much lower fan-out on average; furthermore, its values overall are considerable closer to the gold standard than those of the HGE algorithm. Interestingly, MPO does not quite adhere to Prop.3, i.e., it has 661 concepts that have only one child (which could therefore be merged). We attribute this to the still immature nature of the MPO where many concepts might still be placeholders for further expansion; however, this problem is also "reconstructed" by wDSP.

Overall, the evaluation shows that wDSP outperforms both, GEI and HGE. It has the highest precision of all methods and a better coverage than HGE. The topology of its computed ontology is closest to what one would consider as desirable for ontologies. However, it is also important to state that the simple GEI algorithm produces by far the best coverage at a small expense of precision. Possibly, this expense could be lowered when an evidence cut-off value for edges in the concept graph is applied. Therefore, we believe that further research on

Table 1. Topological properties for resulting ontologies compared to MPO.

		abs	min	max	avg	stdev
GEI	1) in degree	-	1	53	2.83	2.42
	2) sibling/parent nodes	11,271	-	-	-	-
	3) single-child-nodes	1101	-	-	-	-
	4) depth	-	-	-	-	-
	5) out-degree	-	1	1129	4.92	28.35
	6) multiple-level-parents	-	-	-	-	-
	7) rooted hierarchy	no	-	-	-	-
HGE	1) in degree	-	1	7	1.32	0.61
	2) sibling/parent nodes	0	-	-	-	-
	3) single-child-nodes	163	-	-	-	-
	4) depth	-	2	7	2.63	0.81
	5) out-degree	-	1	50	7.17	11.70
	6) multiple-level-parents	0	-	-	-	-
	7) rooted hierarchy	yes	-	-	-	-
wDSP	1) in degree	-	1	7	1.45	0.83
	2) sibling/parent nodes	0	-	-	-	-
	3) single-child-nodes	529	-	-	-	-
	4) depth	-	2	8	3.09	0.92
	5) out-degree	-	1	50	3.76	7.14
	6) multiple-level-parents	0	-	-	-	-
	7) rooted hierarchy	yes	-	-	-	-
MPO	1) in degree	-	1	4	1.18	0.41
	2) sibling/parent nodes	0	-	-	-	-
	3) single-child-nodes	661	-	-	-	-
	4) depth	-	1	13	5.53	1.41
	5) out-degree	-	1	38	3.15	3.18
	6) multiple-level-parents	0	-	-	-	-
	7) rooted hierarchy	yes	-	-	-	-

greedy methods is valuable, which would also need to include a method to root the ontology.

6 Discussion

In this paper, we defined and studied the problem of constructing a consistent and well-formed ontology from a set of inconsistent and heterogenous concept relationships. We presented and analyzed three different strategies ranging from a simple greedy algorithm to a sophisticated reformulation of the DSP. We evaluated our approaches on a large-scale real-world scenario from the Life Sciences. Algorithms were judged based on the semantic correctness and certain topological properties of the created ontologies. Our results indicate that the wDSP approach outperforms greedy solutions in terms of precision and topology of the ontology, but it also has a a considerably lower coverage.

In the future, we will look into ways to directly incorporate topological requirements into the extraction algorithm. Additionally, we will work on increasing coverage without scarifying precision. This can be achieved by either improving the extraction algorithms or by refining the concept matching strategy or by simply adding further sources of relationship evidences. Certainly, the first of these options is the most interesting, but also most challenging one.

Bibliography

[1] J. Brank, M. Grobelnik, and D. Mladenić. A Survey of Ontology Evaluation Techniques. In *Proc. Conf. Data Mining and Data Warehouses*, 2005.

[2] E. Buyko, J. Wermter, M. Poprat, and U. Hahn. Automatically Adapting an NLP Core Engine to the Biology Domain. In *Proc. of the ISMB 2006: BioLink & Bio-Ontoligies SIG Meeting*, 2006.

[3] P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. *Learning Taxonomic Relations from Heterogeneous Sources of Evidence*, chapter II.4, pages 59–76. IOS Press Publication, 2003.

[4] K. Eilbeck, S. E. Lewis, C. J. Mungall, M. Yandell, L. Stein, R. Durbin, and M. Ashburner. The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biol.*, 6, 2005.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1990.

[6] Gene Ontology Consortium. Creating the gene ontology resource: design and implementation. *Genome Res*, 11:1425–1433, 2001.

[7] P. Groth, N. Pavlova, I. Kalev, S. Tonov, G. Georgiev, H.-D. Pohlenz, and B. Weiss. PhenomicDB: a new cross-species genotype/phenotype resource. *Nucl. Acids Res.*, 35:696–699, 2007.

[8] P. Groth and B. Weiss. Phenotype Data: A Neglected Resource in Biomedical Research? *Current Bioinformatics*, 1:347–358, 2006.

[9] J. A. Gulla and T. Brasethvik. A Hybrid Approach to Ontology Relationship Learning. In *Proc. of the 13th Int. Conf. on Natural Language and Information Systems*, pages 79–90. Springer-Verlag, 2008.

[10] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th Conf. on Computational Linguistics*, pages 539–545. Association for Computational Linguistics, 1992.

[11] I. Jurisica, J. Mylopoulos, and E. Yu. Ontologies for Knowledge Management: An Information Systems Perspective. *Knowl. Inf. Syst.*, 6:380–401, 2004.

[12] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *Proc. of the 24th Int. Conf. on Data Engineering*, pages 953–962, 2008.

[13] K. Krishna and R. Krishnapuram. A clustering algorithm for asymmetrically related data with applications to text mining. In *Proc. of the 10th Int. Conf. on Information and Knowledge Management*, pages 571–573. ACM, 2001.

[14] D. Lawrie, W. B. Croft, and A. Rosenberg. Finding topic words for hierarchical summarization. In *Proc. of the 24th Annu. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 349–357. ACM Press, 2001.

[15] A. Maedche and S. Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16:72–79, 2001.

[16] R. Navigli and P. Velardi. Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites. *Comput. Linguist.*, 30:151–179, 2004.

[17] M. F. Porter. An algorithm for suffix stripping. *Readings in Information Retrieval*, pages 313–316, 1997.

[18] C. J. v. Rijsbergen. *Information retrieval.* Butterworths, 1979.

[19] L. A. Sanchis. Exoerimental Analysis of Heuristic Algorithms for the Dominating Set Problem. *Algorithmica*, 33:3–18, 2002.

[20] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proc. of the 22nd Annu. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 206–213. ACM Press, 1999.

[21] P. Schmitz. Inducing Ontology from Flickr Tags. In *Proc. of the Collaborative Web Tagging Workshop (WWW '06).* IW3C2, 2006.

[22] C. L. Smith, C. A. Goldsmith, and J. T. Eppig. The Mammalian Phenotype Ontology as a tool for annotating, analyzing and comparing phenotypic information. *Genome Biol.*, 6, 2005.

[23] K. Supekar, C. Patel, and Y. Lee. Characterizing Quality of Knowledge on Semantic Web. In *Proc. of the Seventeenth Int. Florida Artificial Intelligence Research Society Conf.*, 2004.

[24] S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, and B. Aleman-Meza. OntoQA: Metric-Based Ontology Quality Analysis. In *Proc. of IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, 2005.

[25] L. Zhou. Ontology learning: state of the art and open issues. *Information Technology and Management*, 8:241–252, 2007.