Taxonomy Induction Using Hypernym Subsequences

Amit Gupta
EPFL, Lausanne, Switzerland
amit.gupta@epfl.ch

Rémi Lebret
EPFL, Lausanne, Switzerland
remi.lebret@epfl.ch

Hamza Harkous
EPFL, Lausanne, Switzerland
hamza.harkous@epfl.ch

Karl Aberer
EPFL, Lausanne, Switzerland
karl.aberer@epfl.ch

ABSTRACT

We propose a novel, semi-supervised approach towards domain taxonomy induction from an input vocabulary of seed terms. Unlike all previous approaches, which typically extract direct hypernym *edges* for terms, our approach utilizes a novel probabilistic framework to extract hypernym *subsequences*. Taxonomy induction from extracted subsequences is cast as an instance of the minimum-cost flow problem on a carefully designed directed graph. Through experiments, we demonstrate that our approach outperforms state-of-the-art taxonomy induction approaches across four languages. Importantly, we also show that our approach is robust to the presence of noise in the input vocabulary. To the best of our knowledge, this robustness has not been empirically proven in any previous approach.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**; **Information extraction**; **Ontology engineering**; *Semantic networks*;

KEYWORDS

Knowledge acquisition; taxonomy induction; term taxonomies; algorithms; flow networks; minimum-cost flow optimization;

1 INTRODUCTION

Motivation. Lexical semantic knowledge in the form of term taxonomies has been beneficial in a variety of NLP tasks, including inference, textual entailment, question answering and information extraction [3]. This widespread utility of taxonomies has led to multiple large-scale manual efforts towards taxonomy induction, such as WordNet [22] and Cyc [21]. However, such manually constructed taxonomies suffer from low coverage [15] and are unavailable for specific domains or languages. Therefore, in recent years, there has been substantial interest in extending existing taxonomies automatically or building new ones [4, 5, 19, 34, 38, 40].

Approaches towards automated taxonomy induction consist of two main stages:

1. **extraction of hypernymy relations** (i.e., “is-a" relations between a term and its hypernym such as *apple→fruit*)
2. **structured organization of terms into a taxonomy**, i.e., a coherent tree-like hierarchy.

Extraction of hypernymy relations has been relatively well-studied in previous works. Its approaches can be classified into two main categories: *Distributional* and *Pattern-based* approaches.

Distributional approaches use clustering to extract hypernymy relations from structured or unstructured text. Such approaches draw primarily on the distributional hypothesis [12], which states that semantically similar terms appear in similar contexts. The main advantage of distributional approaches is that they can discover relations not directly expressed in the text. In contrast, *Pattern-based* approaches utilize pre-defined rules or lexico-syntactic patterns to extract terms and hypernymy relations from text [13, 26]. Patterns are either chosen manually [13, 20] or learnt automatically via bootstrapping [35]. Pattern-based approaches usually result in higher accuracies. However, unlike the distributional approaches, which are fully unsupervised, they require a set of seed surface patterns to initiate the extraction process.

Early work on the second stage of taxonomy induction, namely the structured organization of terms into a taxonomy, focused on extending existing partial taxonomies such as WordNet by inserting missing terms at appropriate positions [34, 39, 40]. Another line of work focused on taxonomy induction from Wikipedia by exploiting the semi-structured nature of the Wikipedia category network [7, 10, 24, 30, 31, 36]. Subsequent approaches to taxonomy induction focused on building lexical taxonomies entirely *from scratch*, i.e., from a domain corpus or the Web [1, 2, 19, 25, 28, 38].

Automated taxonomy induction from scratch is preferred because it can be used over arbitrary domains, including highly specific or technical domains, such as Finance or Artificial Intelligence [25]. Such domains are usually under-represented in existing taxonomic resources. For example, WordNet is limited to the most frequent and the most important nouns, adjectives, verbs, and adverbs [11, 23]. Similarly, Wikipedia is limited to popular entities [18], and its utility is further diminished by slowed growth [37].

Past approaches to taxonomy induction from scratch either assume the availability of a clean input vocabulary [28] or employ a time-consuming manual cleaning step over a noisy input vocabulary [38]. For example, Figure 1 shows the pipeline of a typical taxonomy induction approach from a domain corpus [38]. An initial noisy vocabulary is automatically extracted from the domain

---

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3133041

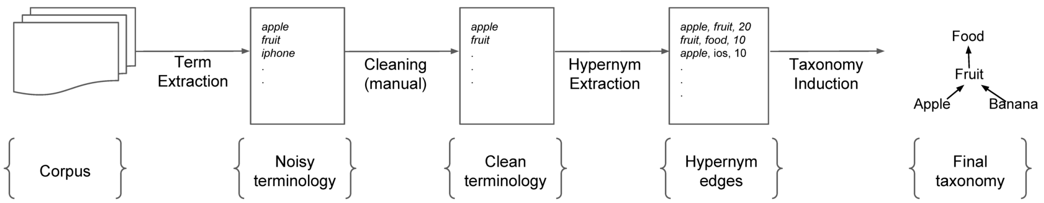


Figure 1: Traditional process for taxonomy induction from a domain-specific corpus [38].

corpus using a term extraction tool, such as *TermExtractor* [32], and is further cleaned manually to produce the final vocabulary. This requirement severely limits the applicability of such approaches in an automated setting because clean vocabularies are usually unavailable for specific domains.

To handle these limitations, we designed our approach to induce a taxonomy directly from a noisy input vocabulary. Consequently, it is the first work to fully automate the taxonomy induction process for arbitrary domains.

Contributions. In this paper, we present a novel, semi-supervised approach for building lexical taxonomies given an input vocabulary of (potentially noisy) seed terms. We leverage the existing work on hypernymy relations extraction and focus on the second stage, i.e. the organization of terms into a taxonomy. Our main contributions are as follows:

- We propose a novel probabilistic framework for extracting longer hypernym subsequences from hypernymy relations, as well as a novel minimum-cost flow based optimization framework for inducing a tree-like taxonomy from a noisy hypernym graph.
- We empirically show that our approach outperforms state-of-the-art taxonomy induction approaches across four different languages, while achieving >32% relative improvement in F1-measure over the Food domain.
- We demonstrate that our subsequence-based model is robust to the presence of noisy terms in the input vocabulary, and achieves a 65% relative improvement in precision over an edge-based model while maintaining similar coverage. To the best of our knowledge, this is the first approach towards taxonomy induction from a noisy input vocabulary.

The rest of the paper is organized as follows. In Section 2, we describe our taxonomy induction approach. In Section 3, we discuss our experiments and performance results. In Section 4, we discuss related work. We conclude in Section 5.

2 TAXONOMY INDUCTION

Given a potentially-noisy vocabulary[1] of seed terms as an input, we define our goal as inducing a taxonomy consisting of these seed terms (and possibly other terms). This taxonomy is a directed acyclic graph with terms as the nodes and the edges indicating a hypernymy relationship between the terms. For our task, we assume the availability of a database of *candidate* hypernymy relations.

Candidate hypernym	Frequency
company	5536
fruit	3898
apple	2119
vegetable	928
orange	797
tech company	619
brand	463
hardware company	460
technology company	427
food	370

Table 1: Candidate hypernyms for the term *apple*.

Multiple such resources have been compiled and made available publicly over the years. A prominent example of such a resource is WebIsA [33], a collection of more than 400 million hypernymy relations for English, extracted from the CommonCrawl web corpus using lexico-syntactic patterns. However, such resources come with a considerable number of noisy candidate hypernyms, typically containing a mixture of relations such as hyponymy, meronymy, synonymy and co-hyponymy. For example, WebIsA has more than 12,000 hypernyms for the term *apple*, including noisy hypernyms such as *orange*, *everyone* and *smartphone*. A sample set of candidate hypernyms and their occurrence frequencies for the term *apple* taken from WebIsA is shown in Table 1.

Our approach to taxonomy induction consists of three main steps:

1. extracting hypernym subsequences for the given seed terms (Section 2.1),
2. aggregating the extracted subsequences into an initial hypernym graph (Section 2.2),
3. pruning the hypernym graph using a minimum-cost flow approach to induce the final taxonomy (Section 2.3).

2.1 Hypernym Subsequences Extraction

Unsupervised or semi-supervised approaches to taxonomy induction typically aim to extract **single hypernym edges** among terms from noisy candidate hypernyms [19, 28]. In contrast, our approach consists of extracting **hypernym subsequences** (where a subsequence is a series of one or more individual hypernym edges).

To motivate this, we first note that Table 1 includes hypernyms of *apple* at different levels of generality, such as *fruit* and *food*. In fact, we observe this pattern in the candidate hypernyms of most terms. This suggests that we can leverage such information to not only

---

[1] In this work, we use terminology and vocabulary interchangeably.

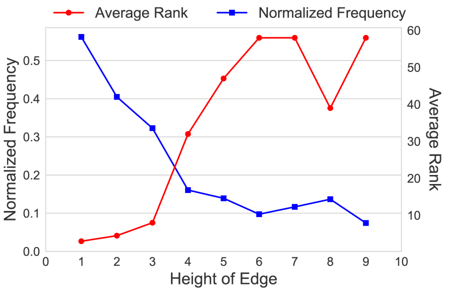


Figure 2: Average rank and normalized frequency of WordNet edges vs. height of edge.

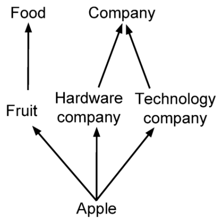


Figure 3: An example DAG built using generalizations of term *apple*.

extract the direct hypernyms of *apple*, but to also extract longer hypernym subsequences, such as *apple→fruit→food*.

This becomes even more important given the result by Velardi et al. [38], who demonstrated that hypernym extraction becomes increasingly erroneous as the generality of terms increases, mainly due to the increase in term ambiguity. To further support this hypothesis, we perform an experiment where we first randomly sample 100 paths from Wordnet. For each edge $a \rightarrow b$ in a sampled path, we plot the normalized frequency[2] of "b as a candidate hypernym for a" against the height of the edge, where frequencies are computed using lexico-syntactic patterns (cf. Table 1). We also plot the average rank of b among candidate hypernyms of a, where candidate hypernyms are ranked by their normalized frequencies in a decreasing order. Results of this experiment are shown in Figure 2. Since edges in WordNet are assumed to be ground truth, it is desired that they have a higher normalized frequency and lower ranks. This small-scale experiment demonstrates that as the height of the edge increases, the normalized frequencies decrease whereas the average ranks increase. Therefore, the accuracy of patterns-based hypernymy detection decreases for more general terms that appear higher in generalization paths. Hence, for such terms, it makes sense to not solely base the hypernym selection on a noisy set of candidate hypernyms. We can potentially improve the accuracy of selected hypernyms for general terms (such as *fruit*) by relying on extracted subsequences starting from more specific terms (such as *apple*). Those subsequences would be evidenced by the less-noisy candidate hypernyms of the specific terms.

In sum, extracting hypernym subsequences is both *possible* and potentially *beneficial*. The remainder of this section describes our model that realizes this intuition.

Model. We now describe our model for extracting hypernym subsequences for a given term. We begin with a general formulation using directed acyclic graphs (referred to as DAG), and we make simplifying assumptions to derive a model for hypernym subsequences. We use the following notations:

- t_0: a given seed term, e.g., *apple*;
- l_t: lexical head of any term t, e.g., l_t=*soup* for t=*chicken soup*;
- E: Hypernym $\underline{E}$vidence, i.e., the set of all the candidate hypernymy relations, in the form of 3-tuples (*hyponym, hypernym, frequency*);
- $E_k(t)$: Hypernym $\underline{E}$vidence for term t, i.e., the set of top-$\underline{k}$ candidate hypernyms for term t, having the highest frequency counts (Table 1 shows a sample from $E_k(t)$ for t=*apple*);
- $E_k(t, m)$: m^{th} ranked candidate hypernym from $E_k(t)$, where $m \leq k$, and ranks are computed by sorting candidate hypernyms in decreasing order of frequency counts;
- $\text{sim}(t_i, t_j)$: A similarity measure between terms t_i and t_j estimated using evidence E;
- G_t: a DAG consisting of generalizations for a term t (Figure 3 shows an example of a possible DAG for t=*apple*).

For a given term t_0, we define the goal of this step of our taxonomy induction approach as finding a DAG $\hat{G}_{t_0}$, which maximizes the conditional probability of G_{t_0}, given the evidence $E_k(t_0)$, for a fixed k:

$$\begin{aligned} \hat{G}_{t_0} &= \underset{G_{t_0}}{\text{argmax}} \Pr(G_{t_0} | E_k(t_0)) \\ &= \underset{G_{t_0}}{\text{argmax}} \Pr(E_k(t_0) | G_{t_0}) \times \Pr(G_{t_0}) \end{aligned} \quad (1)$$

Due to the combinatorial nature of the search space of G_{t_0}, finding an exact solution to the above equation is intractable, even for a small k. Therefore, we make the following simplifying assumptions, which facilitate an efficient search through the search space of G_{t_0}:

- G_{t_0} can be approximated as a set of independent hypernym subsequences with possibly repeated hypernyms. In other words, $G_{t_0} = \bigcup_{i=1}^{b} S_{t_0}^i$ where $S_{t_0}^i$ is the i^{th} subsequence and b is a fixed constant. For example, the DAG shown in Figure 3 can be approximated as a set of three subsequences: (i) *apple→fruit→food*, (ii) *apple→hardware company→company*, and (iii) *apple→technology company→company*. This assumption intuitively derives from the fact that any DAG can be represented by a finite number of subsequences.
- $\forall i$, the joint events $(E_k(t_0), S_{t_0}^i)$ are independent. Intuitively, this assumption implies that each subsequence independently contributes to the evidence $E_k(t_0)$.
- $\forall i$, the direct hypernyms of t_0 in $S_{t_0}^i$ are unique. In other words, for a candidate hypernym h_c of given term t_0, there is at most one subsequence with the first edge $t_0 \rightarrow h_c$. Intuitively, this assumption implies that a candidate hypernym h_c uniquely sense-disambiguates the term t_0, thus resulting in a only one possible generalization subsequence.

In conjunction, these assumptions imply that G_{t_0} is composed of b hypernym subsequences, where each subsequence independently attempts to generate $E_k(t_0)$. Given these assumptions, Equation 1

---

[2] Normalization is performed by dividing frequency counts by the maximum.

transforms into:

$$\hat{G}_{t_0} = \underset{\bigcup_{i=1}^{b} S_{t_0}^{i}}{\operatorname{argmax}} \prod_{i=1}^{b} \Pr(E_k(t_0)|S_{t_0}^{i}) \times \Pr(S_{t_0}^{i}) \tag{2}$$

Estimation. We now describe the estimation of $\Pr(E_k(t_0)|S_{t_0}^i)$ and $\Pr(S_{t_0}^i)$ for a hypernym subsequence $S_{t_0}^i$. In order to motivate the estimation of the conditional probability $\Pr(E_k(t_0)|S_{t_0}^i)$, we start with an example. Consider a valid hypernym subsequence *apple→fruit→food→substance→matter→entity* for the term *apple* (whose candidate hypernyms are in Table 1). At first sight, it might seem desirable for a candidate hypernym from $E_k(t_0)$ (e.g., *fruit*) to have a high similarity with as many terms in the subsequence as possible. However, since the similarity measure is based on the hypernym evidence E, it is plausible that terms such as *matter* and *entity* have a low similarity with the candidate hypernym *fruit*, simply because they are at a higher level of generality. To avoid penalizing such valid subsequences, we let the conditional probability $\Pr(E_k(t_0)|S_{t_0}^i)$ be proportional to the maximum similarity possible between the candidate hypernym and *any* term in the subsequence (e.g., for the candidate hypernym *fruit*, the similarity is 1 as *fruit* is in the subsequence). We aggregate those similarity values across the candidate hypernyms. More formally, assuming subsequence $S_{t_0}^i = t_0 \rightarrow h_{i1} \rightarrow h_{i2} \ldots h_{in}$, where n is the length of $S_{t_0}^i$, we compute the conditional probability as:

$$\Pr(E_k(t_0)|S_{t_0}^i) \propto \sum_{m=1}^{k} (\lambda_1)^m \max_{j \in [1,n]} \big(\mathrm{sim}(E_k(t_0, m), h_{ij})\big) \tag{3}$$

where λ_1 (a fixed parameter) serves as a rank-penalty to penalize candidate hypernyms with lower frequency counts.

We now proceed to compute $\Pr(S_{t_0}^i)$, the other constituent of Equation 2. Towards that, we assume that $S_{t_0}^i$ is a collection of independent hypernym edges. Thus, $\Pr(S_{t_0}^i)$ becomes the product of the individual edges' probabilities:

$$\Pr(S_{t_0}^i) \propto \Pr_e(t_0, h_{i1}) \times (\lambda_2)^n \prod_{j=1}^{n-1} \Pr_e(h_{ij}, h_{i(j+1)}) \tag{4}$$

where $\Pr_e(x_1, x_2)$ is the probability of an individual hypernym edge $x_1 \rightarrow x_2$ between terms x_1 and x_2; λ_2 is a length penalty parameter. Finally, we estimate $\Pr_e(x_1, x_2)$ as a log-linear model using a set of features $\mathbf{f}$, weighted by the learned weight vector $\mathbf{w}$:

$$\Pr_e(x_1, x_2) \propto \exp\big(\mathbf{w} \cdot \mathbf{f}(x_1, x_2)\big) \tag{5}$$

We also use this edge probability to compute the aforementioned similarity function (sim) as:

$$\mathrm{sim}(x_i, x_j) = \max\big(\Pr_e(x_i, x_j), \Pr_e(x_j, x_i)\big) \tag{6}$$

Intuitively, $\Pr(E_k(t_0)|S_{t_0}^i)$ promotes subsequences containing a larger number of candidate hypernyms from $E_k(t_0)$ whereas $\Pr(S_{t_0}^i)$ promotes subsequences consisting of individual edges with a larger probability of hypernymy.

Subsequence Extraction. After inserting Equations 3 and 4 into Equation 2 and taking logarithm, the objective function becomes:

$$\hat{G}_{t_0} = \underset{\bigcup_{i=1}^{b} S_{t_0}^{i}}{\operatorname{argmax}} \sum_{i=1}^{b} \Big[\log \sum_{m=1}^{k} (\lambda_1)^m \max_{j \in [1,n]} \big(\mathrm{sim}(E_k(t_0, m), h_{ij})\big) + \log \Pr_e(t_0, h_{i1}) + n\lambda_2 + \sum_{j=1}^{n-1} \log \Pr_e(h_{ij}, h_{i(j+1)}) \big) \Big]$$

This objective function leads to the following search algorithm for the extraction of subsequences:

1. For a given term t_0, iterate over all candidate hypernyms in $E_k(t_0)$.
2. For each $h_c \in E_k(t_0)$, perform a depth-limited beam search over the space of possible subsequences by recursively exploring the candidate hypernyms of h_c (i.e., $E_k(h_c)$).
3. For each $h_c \in E_k(t_0)$, choose the subsequence S with the highest score (i.e., $\log(\Pr(E_k(t_0)|S) \times \Pr(S))$).
4. Choose the top-b candidate hypernyms based on their corresponding subsequence scores.

While, in theory, we can iterate over all candidate hypernyms in $E_k(t_0)$, in practice, we employ an alternative two-stage execution that significantly improves the running time as well as produces more meaningful subsequences:

- *Search phase*: Proceed as in the aforementioned steps. However, in the special case where a candidate hypernym h_c is a compound term and its lexical head l_{h_c} is also present in $E_k(t_0)$, skip h_c in step (1) of the algorithm[3]. For example, for t_0 = *apple*, candidate hypernyms *tech company*, *software company* and *hardware company* are skipped in step (1) due to the presence of *company* in $E_k(t_0)$ (cf. Table 1).

- *Expansion phase*: In this phase, we augment the subsequences extracted in the search phase to account for skipped compound terms. We focus on the case where the lexical head of the skipped compound terms occurs in a subsequence. In that case, we expand the incoming edge of the lexical head with zero or more of those compound terms. For example, in the subsequence *apple→company→organization*, a potential expansion of the edge *apple→company* is: *apple→American software company→software company→company*. However, special attention has to be taken while generating these potential expansions. For example, the expansion *apple→American software company→British software company→company* is invalid due to the co-hyponymy edge *American software company→British software company*. In contrast, the expansion *apple→American software company→software company→company* is a valid expansion. To avoid invalid expansions, we restrict the possible expansions to the case where the set of pre-modifiers of a compound term is a superset of its hypernym's pre-modifiers (e.g., {*American, software* }⊃{*software*}).

We generate all possible expansions for each edge and rank them by averaging a TF-IDF-style metric across the pre-modifiers of compound terms in each expansion. Our aim in the ranking is two-fold: i) promoting the pre-modifiers, which frequently appear in the evidence $E_k(t_0)$, and ii) penalizing the noisy pre-modifiers unrelated

---

[3] Lexical heads of terms have consistently played a special role in taxonomy induction [10, 31].

Initial subsequences
mortadella→sausage→meat→food *laksa→soup→dish→food*
Expanded subsequences
mortadella→large Italian sausage→sausage→process meat→meat→food *laksa→spicy noodle soup→noodle soup→soup→dish→food*

Table 2: Examples of hypernym subsequences found during the search phase, and their expanded versions.

to t_0 that frequently occur in compound terms (e.g., *several*, *other*, etc.). Hence, we compute the TF score of a pre-modifier as its average frequency of occurrence in the candidate hypernyms $E_k(t_0)$. We compute IDF as the average frequency of occurrences of the pre-modifier in $E_k(t)$ for a random term t. Finally, we choose the top ranked expansion per edge.

To illustrate the result of the previous steps, we show in Table 2 an example of extracted subsequences along with their expanded versions for the food domain. Intuitively, the two-stage execution serves to distinguish between two fundamentally different forms of generalization:

1. **type-based generalization**, which provides core types as generalizations (e.g., *apple→company→organization*).
2. **attribute-based generalization**, which enriches type-based generalization edges. For example, *apple→american software company→software company→company* enriches the individual type-based edge *apple→company*.

In our experiments, models that distinguished between these two different forms of generalizations consistently performed better than models, which attempted to unify them.

Features. We now describe the edge features that we employ for estimating the probability of a hypernymy relation between two terms (cf. Equation 5):

- *Normalized Frequency Diff (n_d)*: Similar to [28], this feature is an asymmetric hypernymy score based on frequency counts. We compute $n_d(x_i, x_j)$ by first normalizing the frequency counts obtained (i.e., the counts in $E_k(x_i)$) for term x_i as follows: $n_f(x_i, x_j) = \frac{\text{freq}(x_i, x_j)}{\max_m \text{freq}(x_i, x_m)}$, where $\text{freq}(x_i, x_j)$ is the frequency count of candidate hypernym x_j in $E_k(x_i)$. Further, we subtract the score in the opposite direction to downrank synonyms and co-hyponyms: $n_d(x_i, x_j) = n_f(x_i, x_j) - n_f(x_j, x_i)$.
- *Generality Diff (g_d)*: We introduce a novel feature for explicitly incorporating the term generality (or abstractness) in our model. To this end, we first define the generality $g(t)$ of a term t as the log of the number of distinct hyponyms present in all candidate hypernymy relations (E); i.e., $g(t) = \log(1 + |x \mid x \to t \in E|)$. We define the generality of an edge as the difference in generality between the hypernym and the hyponym: $g_e(x_i, x_j) = g(x_j) - g(x_i)$.

Intuitively, we aim to promote edges with the right level of generality and penalize edges, which are either too general (e.g., *apple→thing*) or too specific (i.e., edges between synonyms or co-hyponyms, such as *apple→orange*). To realize this intuition, we first sample a random set of terms and collect the edges with highest n_d for these terms (hereafter referred to as *top edges*). We compare the distribution of generality (i.e., g_e) for the top edges vs. the distribution of generality for a set of randomly sampled edges. The assumption is that it is more likely to sample the generality of a correct edge (i.e., edge at right level of generality) from the distribution of top edges as compared to random edges. Hence, given D_t and D_r as the Gaussian distributions estimated from the samples of generality for top edges and random edges respectively, we define the feature as: $g_d(x_i, x_j) = \Pr_{D_t}\big(g_e(x_i, x_j)\big) - \Pr_{D_r}\big(g_e(x_i, x_j)\big)$.

Parameter Tuning. We estimate the weights for features ($\mathbf{w}$ in equation 5), using a support vector machine trained on a manually annotated set of 500 edges. For beam search in the search phase, we use a beam of width 20, and limit the search to subsequences of maximum length 4. We set the rest of the parameters by running grid-search over a manually-defined range of parameters using a small validation set[4]. The final values of parameters are as follows: k=10, b=4, λ_1=λ_2=0.95.

2.2 Aggregation of Subsequences

Up till now, we have described our methodology to generate hypernym subsequences starting from a given term. In this section, we aggregate the hypernym subsequences obtained for a set of seed terms, in order to construct an initial hypernym graph. For that, we undertake the following steps:

Domain Filtering. Given a term t_0, the usual case is that multiple hypernym subsequences corresponding to different senses of the term t_0 are extracted. For example, *apple* can be a *company* or a *fruit*, thus resulting in subsequences *apple→fruit→food* and *apple→software company→company*. However, many of these subsequences will not pertain to the domain of interest (as determined by the seed terms). To eliminate the irrelevant ones, we estimate a smoothed unigram model[5] from all extracted subsequences, and we remove those with generation probabilities below a fixed threshold.

Hypernym Graph Construction. We now aggregate the filtered subsequences into an initial hypernym graph. We construct this graph by grouping the edges with the same start and end nodes together from the filtered subsequences. The weight of each edge is computed as the sum of the scores of subsequences it belongs to (i.e., $\log(\ \Pr(E_k(t)|S) \times \Pr(S)))$. To increase the coverage for compound seed terms that do not yet have a hypernym, we simply add an hypernym edge to their lexical head with weight=∞ (i.e, a very large value) whenever the lexical head is already present in the hypernym graph. Finally, for each cycle in the hypernym graph, we remove the edge with the smallest weight, hence resulting in a DAG. This DAG contains many noisy terms and edges, which are pruned in the next step of our approach.

2.3 Taxonomy Construction

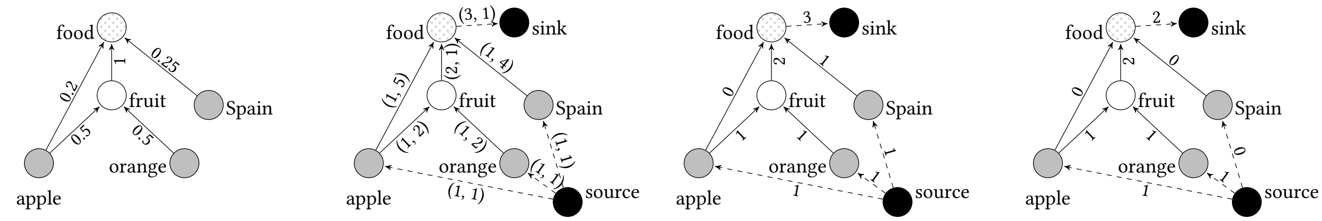
In this step, we aim to induce a tree-like taxonomy from the hypernym DAG obtained in the previous step. We cast this as an instance of the minimum-cost flow problem (MCFP).

MCFP is an optimization problem, which aims to find the cheapest way of sending a certain amount of flow through a flow network.

---

[4] Validation set is excluded from the test set.

[5] We used a weighting function (i.e., step function with cut-off at 50% of the height of the subsequence) to favor terms at lower heights as they are usually more domain-specific.



(a): Noisy hypernym graph (H).

(b): Flow network F with (capacity, cost) values for each edge.

(c): Flow values (f) for each edge found using demand $d = 3$.

(d): Flow values (f) for each edge found using demand $d = 2$.

Figure 4: Execution of the minimum cost flow algorithm starting from our hypernym graph.

It has been used to find the optimal solution in applications like the *transportation problem* [17], where the goal is to find the cheapest paths to send commodities from a group of facilities to the customers via a transportation network. Analogously, we cast the problem of taxonomy induction as finding the cheapest way of sending the seed terms to the root terms through a carefully designed flow network F. We use the *network simplex algorithm* [27] to compute the optimal flow for F, and we select all edges with positive flow as part of our final taxonomy. We now describe our method for constructing the flow network F. In what follows, we refer to Figure 4 at the different steps.

Flow Network Construction. Let V be the vocabulary of input seed terms (e.g., *apple*, *orange*, and *Spain* in Figure 4); H is the noisy hypernym graph constructed in Section 2.2 (cf. Figure 4(a)); $w(x, y)$ is the weight of the edge $x \rightarrow y$ in H; D_x is the set of descendants of term x in H (e.g., *apple* is a descendant of *food*); R is the set of given roots[6] (e.g., *food* in Figure 4). The construction of the flow network F proceeds as follows (cf. Figure 4(b)):

i) For an edge $x \rightarrow y$ in H, add the edge $x \rightarrow y$ in F. Set the capacity (c) of the added edge as $c(x, y) = |D_x \cap V|$. Set the cost (a) of that edge as $a(x, y) = 1/w(x, y)$.

ii) Add a sentinel *source* node s. $\forall v \in V$, add an edge $s \rightarrow v$ with $c(s, v) = a(s, v) = 1$.

iii) Add a sentinel *sink* node t. $\forall r \in R$, add edge $r \rightarrow t$ with $c(r, t) = |D_r \cap V|$ and $a(r, t) = 1$.

Minimum-cost Flow. Given a demand d of the total flow to be sent from s to t, the goal of MCFP is to find flow values (f) for each edge in F that minimize the total cost of flow over all edges: $\sum_{(u,v) \in F} a(u, v) \cdot f(u, v)$. In our construct, demand d represents the maximum number of seed terms that can be included in the final taxonomy. Figures 4(c) and 4(d) show the minimum-cost flow for demand d=3 and d=2 respectively. In both cases, the edge *apple→food* receives f=0 due to the presence of edges *apple→fruit* and *fruit→food* with lower costs. For d=2, the edge *source→Spain* has f=0, implying that the noisy term *Spain* would be removed from the final taxonomy. Intuitively, demand d serves as a parameter for discarding potentially noisy terms in the input vocabulary. More formally, d can be defined as $\alpha|V|$, where α, a user-defined parameter, indicates the desired *coverage* over seed terms. If the vocabulary contains only accurate terms, α is set to 1. For a given α, we run the network simplex algorithm with $d = \alpha|V|$ to compute the minimum-cost flow for F. The final taxonomy consists of all edges with flow > 0.

3 EVALUATION

The aim of the empirical evaluation is to address the following questions:

- How does our approach compare to the state-of-the-art approaches under the assumption of a clean input vocabulary?
- How does our approach perform on a noisy input vocabulary?
- What are the benefits of extracting longer hypernym subsequences compared to single hypernym edges?

To this end, we perform two experiments. In Section 3.1, we compare our taxonomy induction approach against the state of the art, under the simplifying assumption of a clean input vocabulary. Evaluations are performed automatically by computing standard precision, recall and F1 measures against a gold standard.

We then drop the simplifying assumption in Section 3.2, where we show that our taxonomy induction performs well even under the presence of significant noise in the input vocabulary. Evaluation is performed both manually as well as automatically against WordNet as the gold standard. We also demonstrate that the subsequences-based approach significantly outperforms an edges-based variant, thus demonstrating the utility of hypernym subsequences.

In the remainder of this section, we use *SubSeq* to refer to our approach towards taxonomy induction (cf. Section 2).

3.1 Evaluation against the State of the Art

Setup. We use the setting of the SemEval 2016 task for taxonomy extraction [5]. The task provides 6 sets of input terminologies, related to three domains (food, environment and science), for four different languages (English, Dutch, French and Italian). The task requires participants to generate taxonomies for each (terminology, language) pair, which are further evaluated using a variety of techniques, including comparison against a gold standard. Except for a few restricted resources used to construct gold standard, the participants are allowed to use external corpora for hypernymy extraction and taxonomy induction. Participants are compared against each other and against a high-precision string inclusion baseline.

We compare SubSeq with TAXI, the system that reached the first place in all subtasks of the SemEval task [28]. TAXI harvests candidate hypernyms using substring inclusion and lexico-syntactic patterns from text corpora. It further utilizes an SVM trained with individual hypernymy edge features, such as frequency counts and

---

[6] If roots are not provided, a small set of upper terms can be used as roots [38].

	TAXI			SubSeq		
	P	R	F1	P	R	F1
EN	33.2	31.7	32.2	**44.9**	**31.9**	**37.2**
NL	**48.0**	19.7	27.6	42.3	**20.7**	**27.9**
FR	33.4	24.1	27.7	**41.0**	**24.4**	**30.5**
IT	**53.7**	20.7	29.1	49.0	**21.8**	**29.9**

Table 3: Precision (P), Recall (R) and F1 Metrics for TAXI vs. SubSeq across different languages. Results are aggregated over all domains per language.

substring inclusion to classify edges as positive and negative. The positive edges are added to the taxonomy. Panchenko et al. [28] also report that alternate configurations of TAXI with different term-level and edge-level features as well as different classifiers such as Logistic Regression, Gradient Boosted Trees, and Random Forest fail to provide improvements over their approach.

In contrast to SubSeq, which discovers new hypernyms for the seed terms, SemEval task provides the additional assumption that all the terms in the gold standard taxonomies (i.e., including leaf terms and non-leaf terms) are present in the input vocabulary. This would unfairly lower the performance of SubSeq, as SubSeq would find hypernyms, which are possibly correct but not present in the gold standard. Hence, to ensure a fair comparison, we restrict the subsequence extraction and hypernym graph construction step of SubSeq (cf. Section 2) to candidate hypernyms present in the input vocabulary. Furthermore, since candidate hypernymy extraction is orthogonal to our work, we reuse the candidate hypernymy relations made available by TAXI. As a consequence, TAXI and SubSeq are identical in input data conditions as well as evaluation metrics, and only differ in the core taxonomy induction approach.

Evaluation Results. Table 3 shows the language-wise precision, recall and F1 values computed against the gold standard for SubSeq and TAXI. Aggregated over all domains, SubSeq outperforms TAXI for all four languages. It achieves >15% relative improvement in F1 for English and 7% improvement overall. Both methods perform significantly better for English, which can be attributed to the higher accuracy of candidate hypernymy relations for English. Figure 5 shows the performance of SubSeq compared to TAXI and the SemEval baseline across different domains and languages. SubSeq performs best for food domain, where it outperforms TAXI across all the languages. SubSeq performs best for English, where it outperforms TAXI across 3/4 domains.

In our experiments, we noticed that SubSeq achieves the largest improvements when a greater number of hypernym subsequences are found during the subsequence extraction step. For example, SubSeq achieves an average 32.23% relative improvement in F1 over TAXI for the food domain, where on an average 0.67 subsequences are found per term, compared to only 0.44 for the other domains. Similarly, SubSeq performs best for English datasets, where, on an average, 1.09 subsequences are found per term, compared to only 0.32 for other languages. The variation in the number of extracted subsequences per term can be attributed to two factors: (i) number of terms in the input vocabulary, and (ii) number of candidate hypernymy relations available. Due to the assumption that all candidate

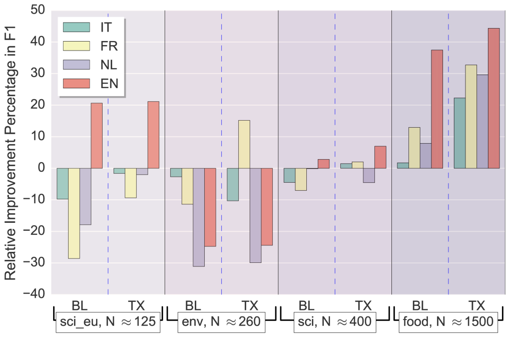


Figure 5: Relative improvement % in F1 for SubSeq, compared to TAXI (TX) and the SemEval Baseline (BL), for different domains and languages. *N* is the average number of terms in the input vocabulary for that domain. *Science eurovoc* datasets are shown separately, as they have significantly fewer input terms than other science datasets.

hypernyms belong to the input vocabulary, larger vocabularies of food domain make it more likely for a candidate hypernym to be found, and hence for a subsequence to be extracted. In a similar fashion, the larger set of available candidate hypernyms for English (~65 million vs. < 2.2 million for other languages) makes it more likely for a subsequence to be extracted for English datasets.

Overall this experiment shows that under the assumption of a clean input vocabulary, SubSeq is more effective that TAXI for most domains in English, and domains with large vocabularies such as food in other languages.

3.2 Evaluation with Noisy Vocabulary

In the previous experiment, we performed taxonomy induction under the simplifying assumption that a clean input vocabulary of relevant domain terms is available. However, as explained in Section 1, in practice, this assumption is rarely satisfied for most domains. Hence, in this experiment, we evaluate the performance of SubSeq in the presence of significant noise in the input vocabulary. TAXI is inapplicable in this setting, as it assumes a clean input vocabulary consisting of both leaf and non-leaf terms. Instead, we compare SubSeq against a baseline, which is an edges-based variant of SubSeq.

Setup. We first build a corpus of relevant documents for the food domain by collecting all English Wikipedia articles with titles matching at least one seed term (post lemmatization) in the SemEval food vocabulary. In total, 1,344 matching Wikipedia articles are found from the initial set of 1,555 seed terms. We run *TermSuite* [6], a state-of-the-art term extraction approach to extract an initial terminology of 12,645 terms. All terms with occurrence counts < 5 in the corpus are removed, thus resulting in a final terminology of 3,977 terms. The final terminology contains numerous noisy terms that are not food items, such as *South Asia* and *triangular*.

We now describe the edge-based baseline, hereafter referred to as *TopEdge*, which extracts individual hypernym edges for terms in the vocabulary. TopEdge is identical to SubSeq, except that rather than extracting hypernym subsequences, it extracts direct hypernyms

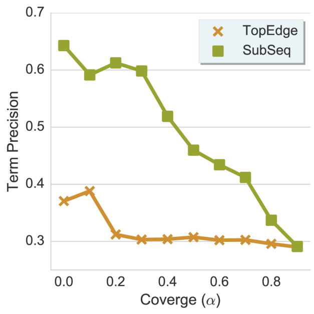


Figure 6: Term precision for SubSeq vs. TopEdge.

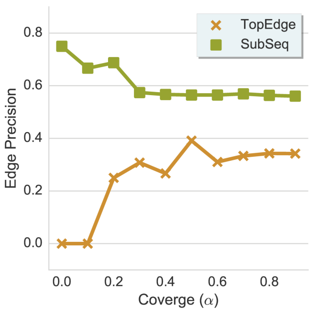


Figure 7: Edge precision for SubSeq vs. TopEdge.

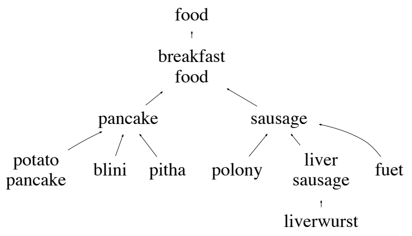


Figure 8: A section of SubSeq taxonomy (α=0.9).

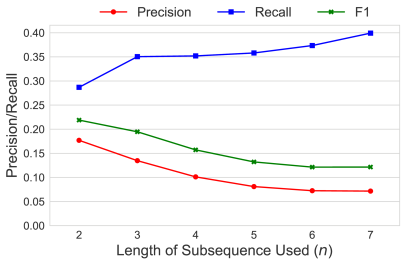


Figure 9: Precision/Recall vs. subsequence length (n).

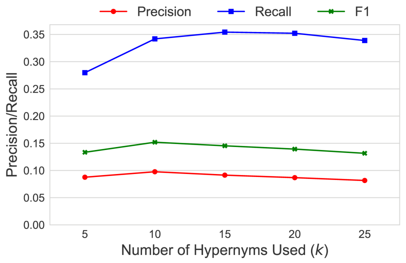


Figure 10: Precision/Recall vs. number of hypernyms used (k).

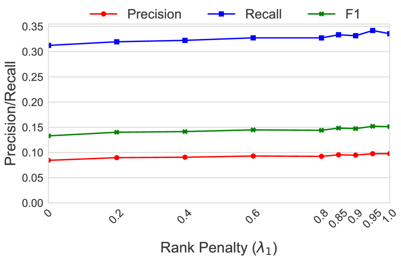


Figure 11: Precision/Recall vs. rank penalty (λ_1).

for terms with the highest hypernym probability $\Pr_e(x_1, x_2)$ (cf. Equation 5). It starts with the seed terms, and recursively extracts hypernyms for terms that do not yet have a hypernym until a fixed number of iterations. The aggregation and taxonomy construction steps are identical to SubSeq (cf. Sections 2.2 and 2.3). Since the only difference between SubSeq and TopEdge is the extraction of hypernym subsequences compared to individual hypernym edges, this experiment also serves to evaluate the utility of extracting hypernym subsequences.

Evaluation Results. We compare the quality of the taxonomies induced by TopEdge and SubSeq against the sub-hierarchy of WordNet rooted at *food* as the gold standard. More specifically, we compute two metrics, i.e., *term precision* and *edge precision*. Term precision of a taxonomy is computed for the set of the input vocabulary terms retained by the taxonomy as: the ratio of the number of terms in the food sub-hierarchy of WordNet to the total number of terms present in WordNet. Edge precision is computed as the ancestor precision: all nodes from the taxonomy that are not present in the WordNet are removed, and precision is computed on the hypernymy relations from the initial vocabulary to the root[7].

Figures 6 and 7 show the term precision and edge precision for TopEdge and SubSeq taxonomy induction methods for varying values of required coverage, i.e., α (cf. Section 2.3). Both Term and edge precision scores for SubSeq are significantly higher than TopEdge across all values of α, hence demonstrating the utility of hypernym subsequences. For both methods, precision scores decrease with increase in α. This behavior is expected, because as α increases additional potentially-noisy seed terms are included in the output taxonomies. Figure 8 shows a section of the SubSeq taxonomy for α=0.9.

We also performed a manual evaluation to judge the quality of the taxonomic edges that are *not* present in the WordNet. Two authors independently annotated 100 such edges each of TopEdge and SubSeq taxonomies for α=0.5. The precision for SubSeq was found to be 86% compared to 52% for TopEdge, with a high inter-annotator agreement (0.68). Both evaluations show that the precision of SubSeq taxonomies is quite high, thus demonstrating the efficacy of SubSeq in inducing taxonomies from noisy terminologies.

When α=1, i.e., all input terms are included in the final taxonomy, term precision is 30%, indicating that only 30% of the terms extracted by the terminology extraction algorithm belong to the WordNet food sub-hierarchy. In contrast, the term precision for the original seed terms provided by SemEval is 75.8%, hence confirming the presence of significant noise in the output of the terminology extraction approach.

Overall, this experiment demonstrates that SubSeq is an effective approach towards taxonomy induction under the presence of significant noise in input terminologies. It also shows that extraction of hypernym subsequences is beneficial and results in significantly more accurate taxonomies.

Parameter Sensitivity. We now discuss the effect of parameters on the efficacy of subsequence extraction. To this end, we first construct a gold standard by sampling a set of 100 terms from the food domain randomly and extracting their generalization paths

---

[7] Trivial edges $t \rightarrow$ *food* are ignored for all terms t.

from WordNet. For a set of parameters, we run subsequence extraction and compute the precision and recall averaged over the top-5 paths per term. The parameters we focus on are the: subsequence length (n), number of hypernyms used (k), and rank-penalty (λ_1) (cf. Equations 3 and 4).

Figure 9 shows the precision/recall values for varying values of subsequence lengths (before the expansion phase). Precision decreases and recall increases as the subsequence length increases. This can be intuitively explained by the observation that candidate hypernyms (cf. Table 1) usually only contain hypernyms up to 3/4 levels. Hence, longer subsequences would typically drift from the original term, thus causing loss of precision. Figure 10 shows the effect of the number of candidate hypernyms used (k) for subsequence extraction. As k increases, both precision and recall increase initially, but drop afterwards. This shows the benefit of utilizing lower-ranked hypernyms for subsequence extraction. However, it also illustrates the significant noise present in candidate hypernyms beyond a certain k. Figure 11 shows the effect of rank-penalty (λ_1), the parameter used to penalize candidate hypernyms with lower frequency counts. Both precision and recall are low for lower values of λ_1 and peak at λ_1=0.95.

We also evaluated the sensitivity to other parameters. We found out that subsequence extraction is fairly stable across different values of beam width and length penalty (λ_2). Moreover, we observed that the number of subsequences per term (b in Equation 3) is also inconsequential beyond a value of 4 as irrelevant subsequences are filtered out by domain filtering (cf. Section 2).

4 RELATED WORK

Taxonomy induction is a well-studied task, and multiple different lines of work have been proposed in the prior literature. Early work on taxonomy induction aims to extend the existing partial taxonomies (e.g., WordNet) by inserting missing terms at appropriate positions. Widdows [39] places the missing terms in regions with most semantically-similar neighbors. Snow et al. [34] use a probabilistic model to attach novel terms in an incremental greedy fashion, such that the conditional probability of a set of relational evidence given a taxonomy is maximized. Yang and Callan [40] cluster terms incrementally using an ontology metric learnt from a set of heterogeneous features such as co-occurrence, context, and lexico-syntactic patterns.

A different line of work aims to exploit collaboratively-built semi-structured content such as Wikipedia for inducing large-scale taxonomies. Wikipedia links millions of entities (e.g., *Johnny Depp*) to a network of inter-connected categories of different granularity (e.g. *Hollywood Actors*, *Celebrities*). WikiTaxonomy [29, 30] labels these links as hypernymy or non-hypernymy, using a cascade of heuristics based on the syntactic structure of Wikipedia category labels, the topology of the network and lexico-syntactic patterns for detecting subsumption and meronymy, similar to Hearst patterns [13]. WikiNet [24] extends WikiTaxonomy by expanding non-hypernymy relations into fine-grained relations such as *part-of, located-in, etc.* YAGO induces a taxonomy by employing heuristics linking Wikipedia categories to corresponding synsets in WordNet [14]. More recently, Flati et al. [7] and Gupta et al. [9] propose approaches towards multilingual taxonomy induction from Wikipedia, resulting in taxonomies for over 270 languages. However, as pointed out by Hovy et al. [16], these taxonomy induction approaches are non-transferable, i.e., they only work for Wikipedia, because they employ lightweight heuristics that exploit the semi-structured nature of Wikipedia content.

Although taxonomy induction approaches based on external lexical resources achieve high precision, they usually suffer from incomplete coverage over specific domains. To address this issue, another line of work focuses on building lexical taxonomies automatically from a domain-specific corpus or Web. Kozareva and Hovy [19] start from an initial set of root terms and basic level terms and use hearst-like lexico-syntactic patterns recursively to harvest new terms from the Web. Hypernymy relations between terms are induced by searching the Web again with surface patterns. The graph of extracted hypernyms is subsequently pruned using heuristics based on the out-degree of nodes and the path lengths between terms. Velardi et al. [38] extract hypernymy relations from textual definitions discovered on the Web, and further employ an optimal branching algorithm to induce a taxonomy. More recently, Bordea et al. [4, 5] introduced the first shared tasks on open-domain Taxonomy Extraction, thus providing a common ground for evaluation. INRIASAC, the top system in 2015 task, uses features based on substrings and co-occurrence statistics [8] whereas TAXI, the top system in 2016 task, uses lexico-syntactic patterns, substrings and focused crawling [28].

In contrast to taxonomy induction approaches which use external resources, taxonomy induction approaches from a domain corpus or Web typically face two main obstacles. First, they assume the availability of a clean input vocabulary of seed terms. This requirement is not satisfied for most domains, thus requiring a time-consuming manual cleaning of noisy input vocabularies. Second, they ignore the relationship between terms and senses. For example, taxonomies induced from WordNet or Wikipedia produce different hypernyms for each sense of the term *apple* (e.g., *apple* is a *fruit* or a *company*). To tackle the second obstacle, taxonomy induction approaches from a domain corpus employ domain filtering to perform implicit sense disambiguation. This is done by removing hypernyms corresponding to domain-irrelevant senses of the terms [38]. Although taxonomies should ideally contain senses rather than terms, term taxonomies have shown significant efficacy in a variety of NLP tasks [2, 3, 38].

To put it in context, our approach is similar to the previous attempts at inducing taxonomies without using external resources such as WordNet or Wikipedia. One key differentiator, however, is that it is robust to the presence of significant noise in the input vocabulary, thus dealing with the first obstacle above. To deal with the second obstacle, our approach performs implicit sense disambiguation via domain filtering at two different steps: (i) domain filtering of subsequences (cf. Section 2.2); (ii) assigning lower cost for likely in-domain edges when applying the minimum-cost flow optimization (cf. Section 2.2 & 2.3).

5 CONCLUSIONS

In this paper, we proposed a novel probabilistic framework for extracting hypernym subsequences from individual hypernymy relations. We also presented a minimum cost-flow optimization approach to taxonomy induction from a noisy hypernym graph. We demonstrated that our subsequence-based approach outperforms state-of-the-art taxonomy induction approaches that utilize individual hypernymy edge features. Unlike previous approaches, our taxonomy induction approach is robust to the significant presence of noise in the input terminology. It also provides a user-defined parameter for controlling the accuracy and coverage of terms and edges in output taxonomies. As a consequence, our approach is applicable to arbitrary domains without any manual intervention, thus truly automating the process of taxonomy induction.

Acknowledgements. This work is supported by a Sinergia Grant by the Swiss National Science Foundation (SNF 147609). The authors would like to thank Marius Paşca for helpful discussions.

REFERENCES

[1] Daniele Alfarone and Jesse Davis. 2015. Unsupervised learning of an is-a taxonomy from a limited domain-specific corpus. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence.* AAAI Press, 1434–1441.

[2] Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured Learning for Taxonomy Induction with Belief Propagation.. In *ACL (1).* 1041–1051.

[3] Chris Biemann. 2005. Ontology Learning from Text: A Survey of Methods. *LDV Forum* 20, 2 (2005), 75–93. http://www.jlcl.org/2005_Heft2/Chris_Biemann.pdf

[4] 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.* Association for Computational Linguistics.

[5] 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.* Association for Computational Linguistics.

[6] Damien Cram and Béatrice Daille. 2016. Termsuite: Terminology extraction with term variant detection. *ACL 2016* (2016), 13.

[7] Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2016. MultiWiBi: The multilingual Wikipedia bitaxonomy project. *Artif. Intell.* 241 (2016), 66–102. https://doi.org/10.1016/j.artint.2016.08.004

[8] Gregory Grefenstette. 2015. INRIASAC: Simple hypernym extraction methods. *arXiv preprint arXiv:1502.01271* (2015).

[9] Amit Gupta, Rémi Lebret, Hamza Harkous, and Karl Aberer. 2017. 280 Birds with One Stone: Inducing Multilingual Taxonomies from Wikipedia using Character-level Classification. *arXiv preprint arXiv:1704.07624* (2017).

[10] Amit Gupta, Francesco Piccinno, Mikhail Kozhevnikov, Marius Pasca, and Daniele Pighin. 2016. Revisiting Taxonomy Induction over Wikipedia. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan.* 2300–2309.

[11] Iryna Gurevych and Elisabeth Wolf. 2010. Expert-Built and Collaboratively Constructed Lexical Semantic Resources. *Language and Linguistics Compass* 4, 11 (2010), 1074–1090.

[12] Zellig S Harris. 1954. Distributional structure. *Word* 10, 2-3 (1954), 146–162.

[13] Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2.* Association for Computational Linguistics, 539–545.

[14] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artif. Intell.* 194 (2013), 28–61.

[15] Eduard Hovy, Zornitsa Kozareva, and Ellen Riloff. 2009. Toward completeness in concept extraction and classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2.* Association for Computational Linguistics, 948–957.

[16] Eduard H. Hovy, Roberto Navigli, and Simone Paolo Ponzetto. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artif. Intell.* 194 (2013), 2–27. https://doi.org/10.1016/j.artint.2012.10.002

[17] Morton Klein. 1967. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science* 14, 3 (1967), 205–220.

[18] Tomáš Kliegr, Václav Zeman, and Milan Dojchinovski. 2014. Linked hypernyms dataset-generation framework and use cases. In *3rd Workshop on Linked Data in Linguistics: Multilingual Knowledge Resources and Natural Language Processing.* Citeseer, 82.

[19] 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.* Association for Computational Linguistics, 1110–1118.

[20] Zornitsa Kozareva, Ellen Riloff, and Eduard H Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs.. In *ACL*, Vol. 8. 1048–1056.

[21] Douglas B Lenat. 1995. CYC: A large-scale investment in knowledge infrastructure. *Commun. ACM* 38, 11 (1995), 33–38.

[22] George A. Miller. 1994. WORDNET: A Lexical Database for English. In *Human Language Technology, Proceedings of a Workshop held at Plainsboro, New Jerey, USA, March 8-11, 1994.* http://aclweb.org/anthology/H/H94/H94-1111.pdf

[23] 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.* Association for Computational Linguistics, 1135–1145.

[24] Vivi Nastase, Michael Strube, Benjamin Boerschinger, Cäcilia Zirn, and Anas Elghafari. 2010. WikiNet: A Very Large Scale Multi-Lingual Concept Network. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta.*

[25] Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *IJCAI*, Vol. 2. 2.

[26] Michael P Oakes. 2005. Using Hearst's Rules for the Automatic Acquisition of Hyponyms for Mining a Pharmaceutical Corpus.. In *RANLP Text Mining Workshop*, Vol. 5. 63–67.

[27] James B Orlin. 1997. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming* 78, 2 (1997), 109–129.

[28] 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. *Proceedings of SemEval* (2016), 1320–1327.

[29] S. Ponzetto and M. Strube. 2007. Deriving a Large Scale Taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence.* Vancouver, British Columbia, 1440–1445.

[30] Simone Paolo Ponzetto and Michael Strube. 2008. WikiTaxonomy: A Large Scale Knowledge Resource. In *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings.* 751–752.

[31] Simone Paolo Ponzetto and Michael Strube. 2011. Taxonomy induction based on a collaboratively built knowledge repository. *Artificial Intelligence* 175, 9-10 (2011), 1737–1756.

[32] Francesco Sclano and Paola Velardi. 2007. Termextractor: a web application to learn the shared terminology of emergent web communities. In *Enterprise Interoperability II.* Springer, 287–290.

[33] Julian Seitner, Christian Bizer, Kai Eckert, Stefano Faralli, Robert Meusel, Heiko Paulheim, and Simone Paolo Ponzetto. 2016. A Large DataBase of Hypernymy Relations Extracted from the Web. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

[34] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, 801–808.

[35] Rion Snow, Daniel Jurafsky, Andrew Y Ng, and others. 2004. Learning syntactic patterns for automatic hypernym discovery.. In *NIPS*, Vol. 17. 1297–1304.

[36] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007.* 697–706.

[37] Bongwon Suh, Gregorio Convertino, Ed H Chi, and Peter Pirolli. 2009. The singularity is not near: slowing growth of Wikipedia. In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration.* ACM, 8.

[38] Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. OntoLearn Reloaded: A Graph-Based Algorithm for Taxonomy Induction. *Computational Linguistics* 39, 3 (2013), 665–707. https://doi.org/10.1162/COLI_a_00146

[39] Dominic Widdows. 2003. Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1.* Association for Computational Linguistics, 197–204.

[40] Hui Yang and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1.* Association for Computational Linguistics, 271–279.