HiExpan: Task-Guided Taxonomy Construction by Hierarchical Tree Expansion

Jiaming Shen[1], Zeqiu Wu[1*], Dongming Lei[1*], Chao Zhang[1], Xiang Ren[2], Michelle T. Vanni[3], Brian M. Sadler[3], Jiawei Han[1 *]
[1]Department of Computer Science, University of at Illinois Urbana-Champaign, IL, USA
[2]Department of Computer Science, University of Southern California, CA, USA
[3]U.S. Army Research Laboratory, MD, USA
[1]{js2, zeqiuwu1, dlei5, czhang82, hanj}@illinois.edu [2]xiangren@usc.edu
[3]{michelle.t.vanni.civ, brian.m.sadler6.civ}@mail.mil

ABSTRACT

Taxonomies are of great value to many knowledge-rich applications. As the manual taxonomy curation costs enormous human effects, automatic taxonomy construction is in great demand. However, most existing automatic taxonomy construction methods can only build hypernymy taxonomies wherein each edge is limited to expressing the "is-a" relation. Such a restriction limits their applicability to more diverse real-world tasks where the parent-child may carry different relations. In this paper, we aim to construct a *task-guided taxonomy* from a domain-specific corpus, and allow users to input a "seed" taxonomy, serving as the task guidance. We propose an expansion-based taxonomy construction framework, namely HiExpan, which automatically generates key term list from the corpus and iteratively grows the seed taxonomy. Specifically, HiExpan views all children under each taxonomy node forming a coherent set and builds the taxonomy by recursively expanding all these sets. Furthermore, HiExpan incorporates a weakly-supervised relation extraction module to extract the initial children of a newly-expanded node and adjusts the taxonomy tree by optimizing its global structure. Our experiments on three real datasets from different domains demonstrate the effectiveness of HiExpan for building task-guided taxonomies.

KEYWORDS

Taxonomy Construction; Hierarchical Tree Expansion; Set Expansion; Weakly-supervised Relation Extraction

ACM Reference Format:
Jiaming Shen[1], Zeqiu Wu[1][1], Dongming Lei[1][1], Chao Zhang[1], Xiang Ren[2], Michelle T. Vanni[3], Brian M. Sadler[3], Jiawei Han[1] . 2018. HiExpan: Task-Guided Taxonomy Construction by Hierarchical Tree Expansion. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220115

*These two authors have equal contributions.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220115

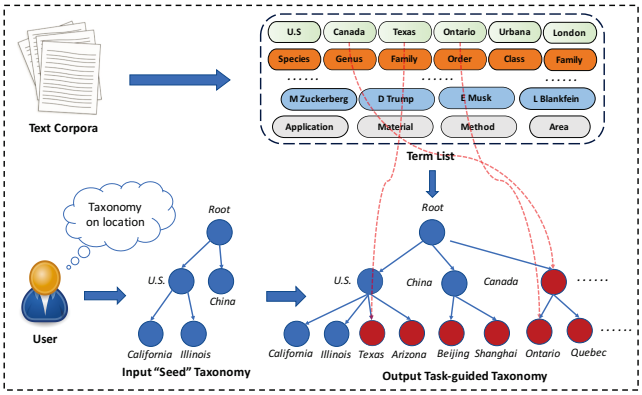


Figure 1: Task-guided taxonomy construction. User provides a "seed" taxonomy tree as task guidance, and we will extract key terms from raw text corpus and generates the desired taxonomy automatically.

1 INTRODUCTION

Taxonomy is the backbone of many knowledge-rich applications such as question answering [49], query understanding [12], and personalized recommendation [52]. Most existing taxonomies are constructed by human experts or in a crowd-sourcing manner. However, such manual constructions are labor-intensive, time-consuming, unadaptable to changes, and rarely complete. As a result, automated taxonomy construction is in great demand.

Existing methods mostly build taxonomies based on "is-A" relations (*e.g.*, a "*panda*" is a "*mammal*" and a "*manmal*" is an "*animal*") [42, 43, 48] by first leveraging pattern-based or distributional methods to extract hypernym-hyponym term pairs and then organizing them into a tree-structured hierarchy. However, such hierarchies cannot satisfy many real-world needs due to its (1) *inflexible semantics*: many applications may need hierarchies carrying more flexible semantics such as "*city-state-country*" in a location taxonomy; and (2) *limited applicability:* the "universal" taxonomy so constructed is unlikely to fit diverse and user-specific application tasks.

This motivates us to work on *task-guided* taxonomy construction, which takes a user-provided "seed" taxonomy tree (as task guidance) along with a domain-specific corpus and generates a desired taxonomy automatically. For example, a user may provide a seed taxonomy containing only two countries and two states along

with a large corpus, and our method will output a taxonomy which covers all the countries and states mentioned in the corpus.

In this study, we propose HiExpan, a framework for task-guided taxonomy construction. Starting with a tiny seed taxonomy tree provided by a user, a weakly supervised approach can be developed by set expansion. A set-expansion algorithm aims to expand a small set of seed entities into a complete set of entities that belong to the same semantic class [33, 35]. Recently we developed an interesting SetExpan algorithm [35], which expands a tiny seed set (*e.g.*, {"*Illinois*", "*California*"}) into a complete set (*e.g.*, U.S. states mentioned in the corpus) by a novel bootstrapping approach. While such an approach is intuitive, there are two major challenges by extending it to generating high-quality taxonomy: (1) modeling global taxonomy information: a term that appears in multiple expanded sets may need conflict resolution and hierarchy adjustment accordingly, and (2) cold-start with empty initial seed set: as an example, initial seed set {"*Ontario*", "*Quebec*"} will need to be found once we add "*Canada*" at the country level as shown in Figure 1.

HiExpan consists of two novel modules for dealing with the above two challenges. First, whenever we observe a conflict (*i.e.*, the same term appearing in multiple positions on taxonomy) during the tree expansion process, we measure a "confidence score" for putting the term in each position and select the most confident position for it. Furthermore, at the end of our hierarchical tree expansion process, we will do a global optimization of the whole tree structure. Second, we incorporate a weakly-supervised relation extraction method to infer parent-child relation information and to find seed children terms under a specific parent. Equipped with these two modules, HiExpan constructs the task-guided taxonomy by iteratively growing the initial seed taxonomy tree. At each iteration, it views all children under a non-leaf taxonomy node as a coherent set and builds the taxonomy by recursively expanding these sets. Whenever a node with no initial children nodes found, it will first conduct seeds hunting. At the end of each iteration, HiExpan detects all the conflicts and resolves them based on their confidence scores.

In summary, this study makes the following contributions:

(1) We introduce a new research problem *task-guided taxonomy construction*, which takes a user-provided seed taxonomy along with a domain-specific corpus as input and aims to output a desired taxonomy that satisfies user-specific application tasks.
(2) We propose HiExpan, a novel expansion-based framework for task-guided taxonomy construction. HiExpan generates the taxonomy by growing the seed taxonomy iteratively. Special mechanisms are also taken by HiExpan to leverage global tree structure information.
(3) We conduct extensive experiments to verify the effectiveness of HiExpan on three real-word datasets from different domains.

The remaining of the paper is organized as follows. Section 2 discusses the related work. Section 3 defines our problem. Then, we present the HiExpan framework in Section 4. In Section 5, we report and analyze the experimental results. Finally, we conclude the paper and discuss some future directions in Section 6.

2 RELATED WORK

In this section, we review related work in following three categories.

2.1 Taxonomy Construction

Most existing approaches to taxonomy construction focus on building hypernym-hyponym taxonomies wherein each parent-child pair expresses the "is-a" relation. Typically, they consist of two key steps: (1) hypernymy relation acquisition (*i.e.*, obtaining hypernym-hyponym pairs), and (2) structured taxonomy induction (*i.e.*, organizing all hypernymy relations into a tree structure).

Methods for hypernymy relation acquisition fall into two classes: pattern-based and distributional. One pioneering pattern-based method is Hearst patterns [11] in which lexical syntactic patterns (*e.g.*, "NP_x such as NP_y") are leveraged to match hypernymy relations. Later studies extend this method by incorporating more linguistic rules [18, 31, 38] or designing generalized patterns such as "*star-pattern*" [24], "*SOL pattern*" [23], and "*meta-pattern*" [13]. These methods could achieve high precision in the result pairs but often suffer low recalls (*i.e.*, many hypernym-hyponym pairs do not match the pre-defined patterns). Along another line, distributional methods predict whether a pair of terms $\langle x, y \rangle$ holds a hypernymy relation based on their distributional representations. Early studies first extract statistical features (*e.g.*, the context words of a term), calculate pairwise term similarity using symmetric metrics (*e.g.*, cosine, Jaccard) [15] or asymmetric metrics (*e.g.*, WeedsPrec [47], SLQS [32]), and predict if $\langle x, y \rangle$ holds a hypernymy relation. More recently, a collections of *supervised* methods [2, 4, 8, 19, 46, 50] are proposed to leverage pre-trained word embeddings and curated training data to directly learn a relation classification/prediction model. However, neither pattern-based nor distributional techniques can be applied to our problem because they are designed exclusively for acquiring hypernym-hyponym pairs, whereas we aim to construct a *task-guided* taxonomy where the parent-child relations are task-specific and subject to user guidance.

For the structured taxonomy induction step, most methods first build a graph where edges represent noisy hypernymy relations, extracted in the former step, and then derive a tree-like taxonomy from this graph. Kozareva and Hovy [14] iteratively retain the longest paths between root and leaf terms and remove other conflicting edges. Navigli *et al.* [25] and Velardi *et al.* [42] use the same longest-path idea to weigh edges and then find the largest-weight taxonomy as a Maximum Spanning Tree. Bansal *et al.* [3] build a factor graph to model hypernymy relations and regard taxonomy induction as a structured learning problem, which can be inferred with loop belief propagation. Recently, Gupta *et al.* [9] propose to build the initial graph using hypernym subsequence (instead of single hypernym pair) and model taxonomy induction as a minimum-cost flow problem [26]. Comparing with these methods, our approach leverages the weak supervision in "seed" taxonomy and builds a task-specific taxonomy in which two terms can hold a non-hypernymy relation. Further, our taxonomy construction framework jointly acquires task-specific relations and induces taxonomy structure, instead of performing the two tasks separately.

2.2 Set Expansion

Our work is also closely related to set expansion — the task of expanding a small set of seed entities into a complete set of entities that belong to the same semantic class [44]. One line of works, including *Google Set* [41], *SEAL* [45] and Lyretail [7], solves this

task by submitting a query of seed entities to an online search engine and mining top-ranked webpages. Other works aim to tackle the task in a *corpus-based* setting where the set is expanded by offline processing a given corpus. They either perform a one-time ranking of all candidate entities [10, 27, 37] or do iterative pattern-based bootstrapping [33, 35, 36]. In this work, in addition to just adding new entities into the set, we go beyond one step and aim to organize those expanded entities in a tree-structured hierarchy (*i.e.*, a taxonomy).

2.3 Weakly-supervised Relation Extraction

There have been studies on weakly supervised relation extraction, which aims at extracting a set of relation instances containing certain semantic relationships. Our method is related to corpus-level relation extraction that identifies relation instances from the entire text corpora [22, 29, 30, 51]. In the weakly supervised setting, there are generally two approaches for corpus-level relation extraction. The first is pattern-based [1, 13, 23], which usually uses bootstrapping to iteratively extract textual patterns and new relation instances. The second approach [21, 28, 40] tries to learn low-dimensional representations of entities such that entities with similar semantic meanings have similar representations. Unfortunately, all these existing methods require a considerable amount of relation instances to train an effective relation classifier, which is infeasible in our setting as we only have a limited number seeds specified by users. Furthermore, these studies do not consider organizing the relation pairs into a taxonomy structure.

3 PROBLEM FORMULATION

The input for our taxonomy construction framework includes two parts: (1) a corpus $\mathcal{D}$ of documents; and (2) a "seed" taxonomy $\mathcal{T}^0$. The "seed" taxonomy $\mathcal{T}^0$, given by a user, is a tree-structured hierarchy and serves as the task guidance. Given the corpus $\mathcal{D}$, we aim to expand this seed taxonomy $\mathcal{T}^0$ into a more complete taxonomy $\mathcal{T}$ for the task. Each node $e \in \mathcal{T}$ represents a term[1] extracted from corpus $\mathcal{D}$ and each edge $\langle e_1, e_2 \rangle$ denotes a pair of terms that satisfies the task-specific relation. We use $\mathcal{E}$ and $\mathcal{R}$ to denote all the nodes and edges in $\mathcal{T}$ and thus $\mathcal{T} \stackrel{\text{def}}{=} (\mathcal{E}, \mathcal{R})$.

Example 3.1. Figure 1 shows an example of our problem. Given a collection of Wikipedia articles (*i.e.*, $\mathcal{D}$) and a "seed" taxonomy containing two countries and two states in the "*U.S.*" (*i.e.*, $\mathcal{T}^0 = (\mathcal{E}^0, \mathcal{R}^0)$), we aim to output a taxonomy $\mathcal{T}$ which covers all countries and states mentioned in corpus $\mathcal{D}$ and connects them based on the task-specific relation "*located in*", indicated by $\mathcal{R}^0$.

4 THE HIEXPAN FRAMEWORK

In this section, we first give an overview of our proposed HiExpan framework in Section 4.1. Then, we discuss our key term extraction module and hierarchical tree expansion algorithm in Section 4.2 and Section 4.3, respectively. Finally, we present our taxonomy global optimization algorithm in Section 4.4.

[1]In this work, we use the word "*term*" and "*entity*" interchangeably.

4.1 Framework Overview

In short, HiExpan views all children under each taxonomy node forming a coherent *set*, and builds the taxonomy by recursively expanding all these sets. As shown in Figure 1, two first-level nodes (*i.e.*, "*U.S.*" and "*China*") form a set representing the semantic class "*Country*" and by expanding it, we can obtain all the other countries. Similarly, we can expand the set {"*California*", "*Illinois*"} to find all the other states in the U.S.

Given a corpus $\mathcal{D}$, we first extract all key terms using a phrase mining tool followed by part-of-speech filter. Since the generated term list contains many task-irrelevant terms (*e.g.*, people's names are totally irrelevant to a location taxonomy), we use a set expansion technique to carefully select best terms, instead of exhaustively testing all possible terms in the list. We refer this process as *width expansion* as it increases the *width* of taxonomy tree. Furthermore, to address the challenge that some nodes do not have an initial child (*e.g.*, the node "*Mexico*" in Figure 2), we find the "seed" children by applying a weakly-supervised relation extraction method, which we refer as *depth expansion*. By iteratively applying these two expansion modules, our hierarchical tree expansion algorithm will first grow the taxonomy to its full size. Finally, we adjust the taxonomy tree by optimizing its global structure. In the following, we describe each module of HiExpan in details.

4.2 Key Term Extraction

We use AutoPhrase, a state-of-the-art phrase mining algorithm [17, 34], to extract all key terms in the given corpus $\mathcal{D}$. AutoPhrase outputs a key term list and identifies the in-corpus occurrences of each key term. After that, we apply a Part-of-Speech (POS) tagger to the corpus and obtain the POS tag sequence of each key term occurrence. Then, we retain the key term occurrence whose corresponding POS tag sequence contains a noun POS tag (*e.g.*, "*NN*", "*NNS*", "*NNP*"). Finally, we aggregate the key terms that have at least one remaining occurrence in the corpus into the key term list. Although the key term list so generated is noisy and may contain some task-irrelevant terms, recall is more critical for this step because we can recognize and simply ignore the false positives at the later stages of HiExpan, but have no chance to remedy the mistakenly excluded task-relevant terms.

4.3 Hierarchical Tree Expansion

The hierarchical tree expansion algorithm in HiExpan is designed to first grow the taxonomy tree. It is based on (1) algorithm SetExpan [35] which expands a small set of seed entities into a complete set of entities that belong to the same semantic class, and (2) REPEL [29] which utilizes a few relation instances (*i.e.*, a pair of entities satisfying a target relation) as seeds to extract more instances of the same relation. Our choice of these two algorithms is motivated by their effectiveness to leverage the weak supervision in the tiny "seed" taxonomy $\mathcal{T}^0$ specified by a user.

4.3.1 Width Expansion. Width expansion aims to find the sibling nodes of a given set of children nodes which share the same parent, as demonstrated in the following example.

Example 4.1 (Width Expansion). Figure 2 shows two expected width expansion results. When given the set {"*U.S.*", "*China*"}, we

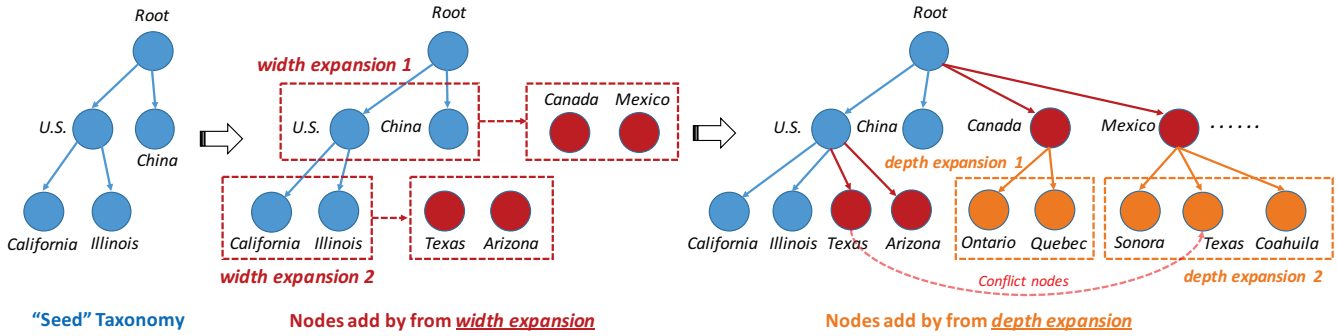


Figure 2: An overview of our hierarchical tree expansion algorithm.

want to find their sibling nodes, "*Canada*", "*Mexico*", and put them under parent node "*Root*". Similarly, we aim to find all siblings of {"*California*", "*Illinois*"} and attach them under parent node "*U.S.*".

This naturally forms a set expansion problem and thus we adapt the SetExpan algorithm in [35] for addressing it. Compared with original SetExpan algorithm, the width expansion algorithm in this paper incorporates the term embedding feature and better leverages the entity type feature. In the following, we first discuss different types of features and similarity measures used, and then describe the width expansion algorithm in details.

Features. We use the following three types of features:

- *skip-pattern*[2]: Given a target term e_i in a sentence, one of its skip-pattern features is "w_{-1} _ w_1" where w_{-1} and w_1 are two context words and e_i is replaced with a placeholder. One advantage of skip-pattern feature is that it imposes strong positional constraints. For example, one skip-pattern of term "*California*" in sentence "*We need to pay California tax.*" is "*pay* _ *tax*". Following [33, 35], we extract up to six skip-patterns of different lengths for one target term e_i in each sentence.
- *term embedding*: We use either the SkipGram model in word2vec [21] or REPEL [29] (described in Section 4.3.2) to learn the term embeddings. We will first use "_" to concatenate tokens in a multi-gram term (*e.g.*, "*Baja California*") and then learn the embedding of this term. The advantage of term embedding feature is that it captures the semantics of each term.
- *entity type*: We obtain each entity's type information by linking it to Probase [48]. The return types serve as the features of that entity. For entities that are not linkable, they simply do not have this entity type feature.

Similarity Measures. A key component in width expansion algorithm is to compute the sibling similarity of two entities e_1 and e_2, denoted as $sim_{sib}(e_1, e_2)$. We first assign the weight between each pair of entity and skip-pattern as follows:

$$f_{e,sk} = \log(1 + X_{e,sk}) \left[\log |V| - \log(\sum_{e'} X_{e',sk}) \right], \tag{1}$$

where $X_{e,sk}$ is the raw co-occurrence count between entity e and skip-pattern sk, and $|V|$ is the total number of candidate entities.

[2]This feature was originally referred as "skip-gram" feature in [35]. Here we change the terminology to avoid the confusion with the SkipGram model used in word2vec [21] for training word embeddings.

Similarly, we can define the association weight between an entity and a type as follows:

$$f_{e,ty} = \log(1 + C_{e,ty}) \left[\log |V| - \log(\sum_{e'} C_{e',ty}) \right], \tag{2}$$

where $C_{e,ty}$ is the confidence score returned by Probase and indicates how confident it believes that entity e has a type ty.

After that, we calculate the similarity of two sibling entities using skip-pattern features as follows:

$$sim^{sk}_{sib}(e_1, e_2|SK) = \frac{\sum_{sk \in SK} \min(f_{e_1,sk}, f_{e_2,sk})}{\sum_{sk \in SK} \max(f_{e_1,sk}, f_{e_2,sk})}, \tag{3}$$

where SK denotes a selected set of "discriminative" skip-pattern features (see below for details). Similarly, we can calculate $sim^{tp}_{sib}(e_1, e_2)$ using all the type features. Finally, we use the cosine similarity to compute the similarity between two entities based on their embedding features $sim^{emb}_{sib}(e_1, e_2)$.

To combine the above three similarities, we notice that a good pair of sibling entities should appear in similar contexts, share similar embeddings, and have similar types. Therefore, we use a multiplicative measure to calculate the sibling similarity as follows:

$$sim_{sib}(e_1, e_2|SK) = \sqrt{(1 + sim^{sk}_{sib}(e_1, e_2|SK)) \cdot sim^{emb}_{sib}(e_1, e_2)} \cdot \sqrt{1 + sim^{tp}_{sib}(e_1, e_2)}. \tag{4}$$

The Width Expansion Process. Given a seed entity set S and a candidate entity list V, a straightforward idea to compute each candidate entity's average similarity with all entities in the seed set S using all the features. However, this approach can be problematic because (1) the feature space is huge (*i.e.*, there are millions of possible skip-pattern features) and noisy, and (2) the candidate entity list V is noisy in the sense that many entities in V are completely irrelevant to S. Therefore, we take a more conservative approach by first selecting a set of quality skip-pattern features and then scoring an entity only if it is associated with at least one quality skip-pattern feature.

Starting with the seed set S, we first score each skip-pattern feature based on its accumulated strength with entities in S (*i.e.*, $score(sk) = \sum_{e \in S} f_{e,sk}$), and then select top 200 skip-pattern features with maximum scores. After that, we use sampling without replacement method to generate 10 subsets of skip-pattern features $SK_t, t = 1, 2, \ldots, 10$. Each subset SK_t has 120 skip-pattern features.

Given an SK_t, we will consider a candidate entity in V only if it has association will at least one skip-pattern feature in SK_t. The score of a considered entity is calculated as follows:

$$score(e|S, SK_t) = \frac{1}{|S|} \sum_{e' \in S} sim_{sib}(e, e'|SK_t). \quad (5)$$

For each SK_t, we can obtain a rank list of candidate entities L_t based on their scores. We use r_t^i to denote the rank of entity e_i in L_t and if e_i does not appear in L_t, we set $r_t^i = \infty$. Finally, we calculate the mean reciprocal rank (mrr) of each entity e_i and add those entities with average rank above r into the set S as follows:

$$mrr(e_i) = \frac{1}{10} \sum_{t=1}^{10} \frac{1}{r_t^i}, \qquad S = S \cup \{e_i | mrr(e_i) > \frac{1}{r}\}. \quad (6)$$

The key insight of above aggregation mechanism is that an irrelevant entity will not appear frequently in multiple L_t at top positions and thus likely has a low mrr score. The same idea in proved effective in [35]. In this paper, we set $r = 5$.

4.3.2 Depth Expansion. The width expansion algorithm requires an initial seed entity set to start with. This requirement is satisfied for nodes in the initial seed taxonomy $\mathcal{T}^0$ as their children nodes can naturally form such a set. However, for those newly-added nodes in taxonomy tree (*e.g.*, the node "*Canada*" in Figure 2), they do not have any child node and thus we cannot directly apply the width expansion algorithm. To address this problem, we use *depth expansion* algorithm to acquire a target node's initial children by considering the relations between its sibling nodes and its niece/nephew nodes. A concrete example is shown below.

Example 4.2 (Depth Expansion). Consider the node "*Canada*" in Figure 2 as an example. This node is generated by the previous width expansion algorithm and thus does not have any child node. We aim to find its initial children (*i.e.*, "*Ontario*" and "*Quebec*") by modeling the relation between the siblings of node "*Canada*" (*e.g.*, "*U.S.*") and its niece/nephew node (*e.g.*, "*California*", "*Illinois*"). Similarly, given the target node "*Mexico*", we want to find its initial children such as node "*Sonora*".

Our depth expansion algorithm relies on term embeddings, which encode the term semantics in a fix-length dense vector. We use $\mathbf{v}(t)$ to denote the embedding vector of term t. As shown in [8, 19, 21], the offset of two terms' embeddings can represent the relationship between them, which leads to the following observation that $\mathbf{v}(\text{"}U.S\text{"}) - \mathbf{v}(\text{"}California\text{"}) \approx \mathbf{v}(\text{"}Canada\text{"}) - \mathbf{v}(\text{"}Ontario\text{"})$. Therefore, given a target parent node e_t, a set of reference edges $E = \{\langle e_p, e_c \rangle\}$ where e_p is the parent node of e_c, we calculate the "goodness" of putting node e_x under parent node e_t as follows:

$$sim_{par}(\langle e_t, e_x \rangle) = cos\left(\mathbf{v}(e_t) - \mathbf{v}(e_x), \frac{1}{|E|} \sum_{\langle e_p, e_c \rangle} \mathbf{v}(e_p) - \mathbf{v}(e_c)\right), \quad (7)$$

where $cos(\mathbf{v}(x), \mathbf{v}(y))$ denotes the cosine similarity between vector $\mathbf{v}(x)$ and $\mathbf{v}(y)$. Finally, we score each candidate entity e_i based on $sim_{par}(\langle e_t, e_i \rangle)$ and select top-3 entities with maximum score as the initial children nodes under node e_t.

The term embedding is learned from REPEL [29], a model for weakly-supervised Relation Extraction using Pattern-enhanced Embedding Learning. It takes a few seed relation mentions (e.g. "US-Illinois" and "US-California") and outputs term embeddings as well as reliable relational phrases for target relation type(s). REPEL consists of a pattern module which learns a set of reliable textual patterns, and a distributional module, which learns a relation classifier on term representations for prediction. As both modules provide extra supervision for each other, the distributional module learns term embeddings supervised by more reliable patterns from the pattern module. By doing so, the learned term embeddings carry more useful information than those obtained from other embedding models like word2vec [21] and PTE [39], specifically for finding relation tuples of the target relation type(s).

4.3.3 Conflict Resolution. Our hierarchical tree expansion algorithm *iteratively* applies width expansion and depth expansion to grow the taxonomy tree to its full size. As the supervision signal from the user-specified seed taxonomy $\mathcal{T}^0$ is very weak (*i.e.*, only few nodes and edges are given), we need to make sure those nodes introduced in the first several iterations are of high quality and will not mislead the expansion process in later iterations to a wrong direction. In this work, for each task-related term, we aim to find its single best position on our output task-guided taxonomy $\mathcal{T}$. Therefore, when finding a term appears in multiple positions during our tree expansion process, we say a "conflict" happens and aim to resolve such conflict by finding the best position that term should reside in.

Given a set of conflicting nodes C which corresponds to different positions of a same entity, we apply the following three rules to select the best node out of this set. First, if any node is in the seed taxonomy $\mathcal{T}^0$, we directly select this node and skip the following two steps. Otherwise, for each pair of nodes in C, we check whether one of them is the ancestor of the other and retain only the ancestor node. After that, we calculate the "confidence score" of each remaining node $e \in C$ as follows:

$$conf(e) = \frac{1}{|sib(e)|} \sum_{e' \in sib(e)} sim_{sib}(e, e'|SK) \cdot sim_{par}(\langle par(e), e \rangle), \quad (8)$$

where $sib(e)$ denotes the set of all sibling nodes of e and $par(e)$ represents its parent node. The skip-pattern feature in SK is selected based on its accumulated strength with entities in $sib(e)$. This equation essentially captures a node's joint similarity with its siblings and its parent. The node with highest confidence score will be selected. Finally, for each node in C that is not selected, we will delete the whole subtree rooted by it, cut all the sibling nodes added after it, and put it in its parent node's "children backlist". A concrete example is shown below.

Example 4.3 (Conflict Resolution). In Figure 2, we can see there are two "*Texas*" nodes, one under "*U.S.*" and the other under "*Mexico*". As none of them is from initial "seed" taxonomy and they do not hold an ancestor-descendant relationship, we need to calculate each node's confidence score based on Eq. (8). Since "*Texas*" has a stronger relation with other states in U.S., comparing with those in Mexico, we will select the "*Texas*" node under "*U.S.*". Then, for the other node under "*Mexico*", we will delete it and cut "*Coahuila*", a sibling node added after "*Texas*". Finally, we let the node "*Mexico*" to remember that "*Texas*" is not one of its children, which prevents the "*Texas*" node being added back later. Notice that although the

Algorithm 1: Hierarchical Tree Expansion.

Input: A seed taxonomy $\mathcal{T}^0$; a candidate term list V; maximum expansion iteration *max_iter*.
Output: A task-guided taxonomy $\mathcal{T}$.

$\mathcal{T} \leftarrow \mathcal{T}^0$;
for *iter from 1 to max_iter* **do**
  $q \leftarrow queue([\mathcal{T}.rootNode])$;
  while *q is not empty* **do**
    $e_t \leftarrow q.pop()$;
    ⊡ ***Depth Expansion***;
    if $e_t.children$ *is empty* **then**
      $S \leftarrow$ DEPTH-EXPANSION(e_t);
      $e_t.children \leftarrow S$;
      $q.push(S)$;
    ⊡ ***Width Expansion***;
    $C_{new} \leftarrow$ WIDTH-EXPANSION$(e_t.children)$;
    $e_t.children = e_t.children \oplus C_{new}$;
    $q.push(C_{new})$;
  ⊡ ***Conflict Resolution***;
  Identify conflicting nodes in $\mathcal{T}$ and resolve the conflicts;
Return $\mathcal{T}$;

"*Coahuila*" node is cut here, it may be added back in a later iteration by our tree expansion algorithm.

Summary. Algorithm 1 shows the whole process of hierarchical tree expansion. It iteratively expands the children of every node on a currently expanded taxonomy tree, starting from the root of this tree. Whenever a target node e_t with no children is found, it first applies depth expansion to obtain the initial children nodes S and then uses width expansion to acquire more children nodes C_{new}. At the end of each iteration, it resolves all the conflicting nodes. The iterative process terminates after expanding the tree *max_iter* times and the final expanded taxonomy tree $\mathcal{T}$ will be returned.

4.4 Taxonomy Global Optimization

In Algorithm 1, a node will be selected and attached onto the taxonomy based on its "local" similarities with other sibling nodes and its parent node. While modeling only the "local" similarity can simplify the tree expansion process, we find the resulting taxonomy may not be the best from a "global" point of view. For example, when expanding the France regions, we find that the entity "Molise", an Italy region, will be mistakenly added under the "France" node, likely because it shares many similar contexts with some other regions of France. However, when we take a global view of the taxonomy and ask the following question—*which country is Molise located in?*, we can easily put "Molise" under "Italy" as it shares more similarities with those in Italy than in France.

Motivated by the above example, we propose a *taxonomy global optimization module* in HiExpan. The key idea is to adjust each two contiguous levels of the taxonomy tree and to find the best "parent" node at the upper level for each "child" node at the lower level. In Figure 2, for example, the upper level consists of all the countries while the lower level contains each country' first-level administrative divisions. Intuitively, our taxonomy global optimization makes the following two hypotheses: (1) entities that have the same parent are similar to each other and form a coherent set, and (2) each entity is more similar to its correct parent compared with other siblings of its correct parent.

Formally, suppose there are m "parent" nodes at the upper level and n "child" nodes at the lower level, we use $\mathbf{W} \in \mathbb{R}^{n\times n}$ to model the entity-entity sibling similarity and use $\mathbf{Y^c} \in \mathbb{R}^{n\times p}$ to capture the two entities's parenthood similarity. We let $\mathbf{W_{ij}} = sim_{sib}(e_i, e_j)$ if $i \neq j$, otherwise we set $\mathbf{W_{ii}} = 0$. We set $\mathbf{Y^c_{ij}} = sim_{par}(\langle e_j, e_i\rangle)$. Furthermore, we define another $n \times p$ matrix $\mathbf{Y}^s$ with $\mathbf{Y^s_{ij}} = 1$ if a child node e_i is under parent node e_j and $\mathbf{Y^s_{ij}} = 0$ otherwise. This matrix captures the current parent assignment of each child node. We use $\mathbf{F} \in \mathbb{R}^{n\times p}$ to represent the child nodes' parent assignment we intend to learn. Given a $\mathbf{F}$, we can assign each "child" node e_i to a "parent" node $e_j = \arg\max_j \mathbf{F_{ij}}$. Finally, we propose the following optimization problem to reflect the previous two hypotheses:

$$\min_{\mathbf{F}} \sum_{i,j}^{n} \mathbf{W_{ij}} \left\| \frac{\mathbf{F_i}}{\sqrt{\mathbf{D_{ii}}}} - \frac{\mathbf{F_j}}{\sqrt{\mathbf{D_{jj}}}} \right\|_2^2 + \mu_1 \sum_{i=1}^{n} \left\| \mathbf{F_i} - \frac{\mathbf{Y^c_i}}{\|\mathbf{Y^c_i}\|_1} \right\|_2^2 + \mu_2 \sum_{i=1}^{n} \left\| \mathbf{F_i} - \mathbf{Y^s_i} \right\|_2^2, \tag{9}$$

where $\mathbf{D_{ii}}$ is the sum of i-th row of $\mathbf{W}$, and μ_1, μ_2 are two nonnegative model hyper-parameters. The first term in Eq. (9) corresponds to our first hypothesis and models two entities' sibling similarity. Namely, if two entities are similar to each other (*i.e.*, large $\mathbf{W_{ij}}$), they should have similar parent node assignments. The second term in Eq. (9) follows our second hypothesis to model the parenthood similarity. Finally, the last term in Eq. (9) serves as the smoothness constraints and captures the taxonomy structure information before the global adjustment.

To solve the above optimization problem, we take the derivative of its objective function with respect to $\mathbf{F}$ and can obtain the following closed form solution:

$$\begin{aligned} \mathbf{F}^* &= (\mathbf{I} - \alpha S)^{-1} \cdot (\beta_1 \mathbf{Y^c} + \beta_2 \mathbf{Y^s}), \\ \mathbf{S} &= \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}, \end{aligned} \tag{10}$$

where $\alpha_1 = \frac{1}{1+\mu_1+\mu_2}$, $\beta_1 = \frac{\mu_1}{1+\mu_1+\mu_2}$ and $\beta_2 = \frac{\mu_2}{1+\mu_1+\mu_2}$. The calculation procedure is similar to the one in [53].

5 EXPERIMENTS

5.1 Experimental Setup

5.1.1 Datasets. We use three corpora from different domains to evaluate the performance of HiExpan: (1) **DBLP** contains about 156 thousand paper abstracts in computer science field; (2) **Wiki** is a subset of English Wikipedia pages used in [16, 35]; (3) **PubMed-CVD** contains a collection of 463 thousand research paper abstracts regarding cardiovascular diseases retrieved from the PubMed[3]. Table 1 lists the details of these datasets used in our experiment. All datasets are available for download at: http://bit.ly/2Jbilte.

5.1.2 Compared Methods. To the best of our knowledge, we are the first to study the problem of task-guided taxonomy construction with user guidance, and thus there is no suitable baseline to compare with directly. Therefore, here we evaluate the effectiveness

[3]https://www.ncbi.nlm.nih.gov/pubmed.

Table 1: Datasets statistics.

Dataset	File Size	# of Sentences	# of Entities
Wiki	1.02GB	1.50M	41.2K
DBLP	520MB	1.10M	17.1K
PubMed-CVD	1.60GB	4.48M	36.1K

of HiExpan by comparing it with a heuristic set-expansion based method and its own variations as follows:

- HSetExpan is a baseline method which iteratively applies SetExpan algorithm [35] at each level of taxonomy. For each lower level node, this method finds its best parent node to attach according to the children-parent similarity measure defined in Eq. (7).
- NoREPEL is a variation of HiExpan without the REPEL [29] module which jointly leverages pattern-based and distributional methods for embedding learning. Instead, we use the SkipGram model [21] for learning term embeddings.
- NoGTO is a variation of HiExpan without the taxonomy global optimization module. It directly outputs the taxonomy generated by hierarchical tree expansion algorithm.
- HiExpan is the full version of our proposed framework, with both REPEL embedding learning module and taxonomy global optimization module enabled.

5.1.3 Parameter Setting. We use the above methods to generate three taxonomies, one for each corpus. When extracting the key term list using AutoPhrase [34], we treat phrases that occur over 15 times in the corpus to be frequent. The embedding dimension is set to 100 in both REPEL [29] and SkipGram model [21]. The maximum expansion iteration number *max_iter* is set to 5 for all above methods. Finally, we set the two hyper-parameters used in taxonomy global optimization module as $\mu_1 = 0.1$ and $\mu_2 = 0.01$.

5.2 Qualitative Results

In this subsection, we show the taxonomy trees generated by HiExpan across three text corpora with different user-guidances. Those seed taxonomies are shown in the left part of Figure 3.

- As shown in Figure 3(a), the "seed" taxonomy containing three countries and six states/provinces. At the first level, we have "United States", "China" as well as "Canada". Under the node "United States", we are given "California", "Illinois", as well as "Florida" as initial seeds. We do the same for "Shandong", "Zhejiang" and "Sichuan" under node "China". Our goal is to output a taxonomy which covers all countries and state/provinces mentioned in the corpus and connects them based the "country-state/province" relation. On the right part of Figure 3(a), we show a fragment of the taxonomy generated by HiExpan which contains the expanded countries and Canadian provinces. HiExpan first uses the depth expansion algorithm to find initial children under "Canada" (*i.e.*, "Alberta" and "Manitoba") and then, starting from the set {"Alberta", "Manitoba"}, it applies the width expansion algorithm to obtain more Canadian provinces. These steps are repeated and finally HiExpan is able to find countries like "England", "Australia", "Germany" in the first-level of taxonomy and to discover states/provinces of each country.
- Figure 3(b) shows parts of the taxonomy generated by HiExpan on the DBLP dataset. Given the initial seed taxonomy (the left part of Figure 3(b)), HiExpan automatically discovers many computer science subareas such as "information retrieval", "wireless networks" and "image processing". We can also zoom in to look at the taxonomy at a more granular level. Taking the node "natural language processing" as an example, HiExpan successfully finds major subtopics in natural language processing such as "question answering", "text summarization", and "word sense disambiguation". HiExpan can also find subtopics under image processing even without any initial seeds entities. As shown on the right part of Figure 3(b), we have obtained high-quality subtopics of "image processing" such as "image enhancement", "image compression", "skin detection", and etc.
- In Figure 3(c), we let HiExpan to run on the PubMed-CVD data and show parts of the resulting taxonomy. We feed the model with 3 seeds at the top level, namely "cardiovascular abnormalities", "vascular diseases" and "heart disease" along with 3 seeds under each top-level node. At the top level, HiExpan generates labels such as "coronary artery diseases", "heart failures", "heart diseases", and "cardiac diseases". Here, we notice that many labels, *e.g.*, "heart disease" and "cardiac disease" are actually synonyms. These synonyms are put at the same level in the taxonomy generated by HiExpan since they share same semantics and appear in similar contexts. We leave synonyms discovery and resolution as an important future work.

Table 2 shows the effect of taxonomy global optimization module in HiExpan. From the experiment on the Wiki dataset, we observe that 'the node 'London" was originally attached to "Australia", but after applying the taxonomy global optimization module, this node is correctly moved under "England". Similarly, in the DBLP dataset, the term "unsupervised learning" was initially located under "data mining" but later being moved under the parent node "machine learning". This demonstrates the effectiveness of our taxonomy global optimization module.

5.3 Quantitative Results

In this subsection, we quantitatively evaluate the quality of the taxonomies constructed by different methods.

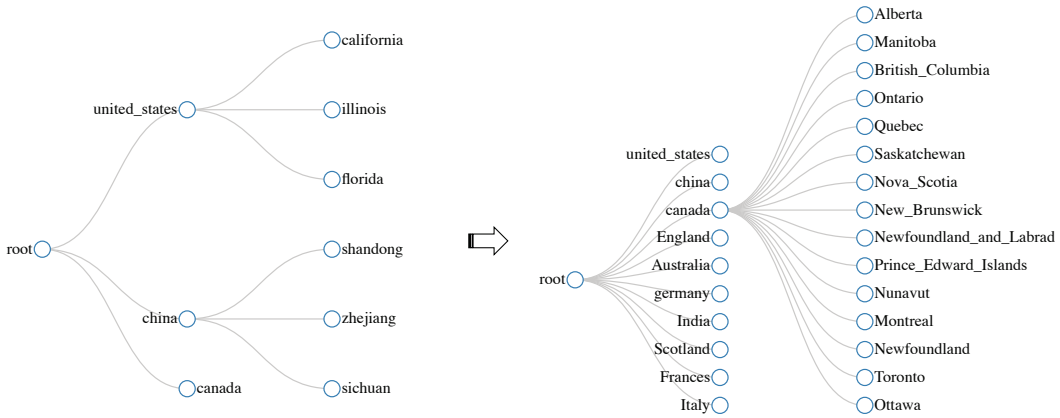
5.3.1 Evaluation Metrics. Evaluating the quality of an entire taxonomy is challenging due to the existence of multiple aspects that should be considered and the difficulty of obtaining gold standard [43]. Following [5, 6, 20], we use *Ancestor-F*1 and *Edge-F*1 for taxonomy evaluation in this study.

Ancestor-F1 measures correctly predicted ancestral relations. It enumerates all the pairs on the predicted taxonomy and compares these pairs with those in the gold standard taxonomy.

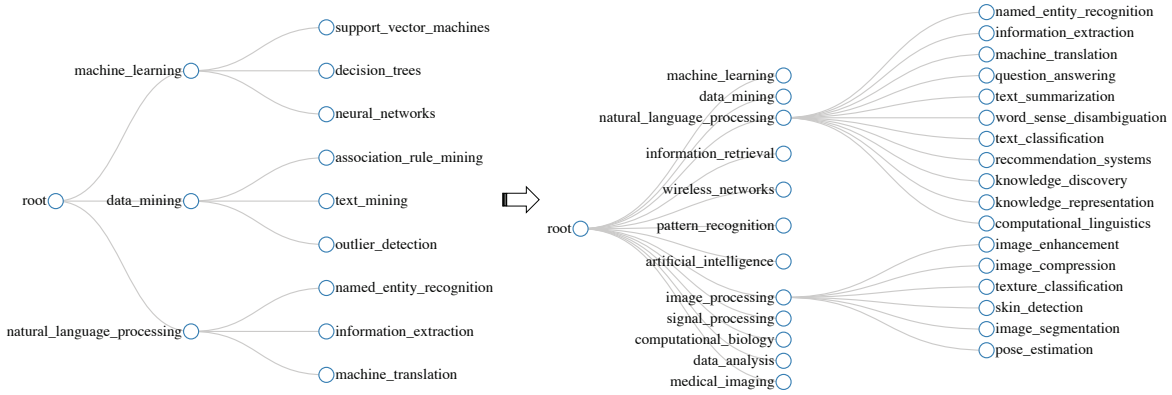
$$P_a = \frac{|\text{is-ancestor}_{\text{pred}} \cap \text{is-ancestor}_{\text{gold}}|}{|\text{is-ancestor}_{\text{pred}}|},$$

$$R_a = \frac{|\text{is-ancestor}_{\text{pred}} \cap \text{is-ancestor}_{\text{gold}}|}{|\text{is-ancestor}_{\text{gold}}|},$$

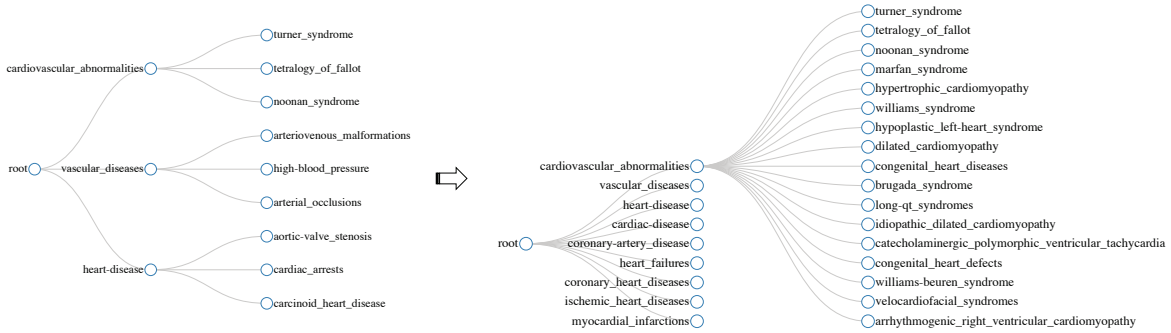
$$F1_a = \frac{2P_a * R_a}{P_a + R_a},$$



(a) Parts of the taxonomy generated by HiExpan on the Wiki dataset.



(b) Parts of the taxonomy generated by HiExpan on the DBLP dataset.



(c) Parts of the taxonomy generated by HiExpan on the PubMed-CVD dataset.

Figure 3: Qualitative results: we show the taxonomy trees generated by HiExpan across three different corpora.

where P_a, R_a, $F1_a$ denote the ancestor precision, ancestor recall, and ancestor F1-score, respectively.

Edge-F1 compares edges predicted by different taxonomy construction methods with edges in the gold standard taxonomy. Similarly, we denote edge-based metrics as P_e, R_e, and $F1_e$, respectively.

To construct the gold standard, we extract all the parent-child edges in taxonomies generated by different methods in table 3. Then we pool all the edges together and ask five people, including the second and third author of this paper as well as three volunteers, to judge these pairs independently. We show them seed parent-child

Table 2: NoGTO shows the parent of an entity before applying taxonomy structure optimization. HiExpan shows the parent node of this entity after optimizing the taxonomy structure.

Dataset	Entity	NoGTO	HiExpan
Wiki	London	Australia	England
	Chiba	China	Japan
	Molise	Frances	Italy
	New_South_Wales	England	Australia
	Shropshire	Scotland	England
DBLP	unsupervised_learning	data_mining	machine_learning
	social_network_analysis	natural_language_processing	data_mining
	multi-label_classification	information_retrieval	machine_learning
	pseudo-relevance_feedback	computational_biology	information_retrieval
	function_approximate	data_analysis	machine_learning

Table 3: Quantitative results: we show the quantitative results of the taxonomies constructed by HSetExpan, NoREPEL, NoGTO, and HiExpan. P_a, R_a, $F1_a$ denote the ancestor-Precision, ancestor-Recall, and ancestor-F1-score, respectively. Similarly, we denote edge-based metrics as P_e, R_e, and $F1_e$, respectively.

Method	Wiki						DBLP						PubMed-CVD					
	P_a	R_a	$F1_a$	P_e	R_e	$F1_e$	P_a	R_a	$F1_a$	P_e	R_e	$F1_e$	P_a	R_a	$F1_a$	P_e	R_e	$F1_e$
HSetExpan	0.740	0.444	0.555	0.759	0.471	0.581	0.743	**0.448**	**0.559**	0.739	0.448	0.558	0.524	0.438	0.477	0.513	0.459	0.484
NoREPEL	0.696	0.596	0.642	0.697	0.576	0.631	0.722	0.384	0.502	0.705	**0.464**	0.560	0.583	**0.473**	0.522	0.593	**0.541**	0.566
NoGTO	0.827	0.708	0.763	0.810	0.671	0.734	0.821	0.366	0.506	0.779	0.433	0.556	0.729	0.443	0.551	0.735	0.506	0.599
HiExpan	**0.847**	**0.725**	**0.781**	**0.848**	**0.702**	**0.768**	**0.843**	0.376	0.520	**0.829**	0.460	**0.592**	**0.733**	0.446	**0.555**	**0.744**	0.512	**0.606**

pairs as well as the generated parent-child pairs, and ask them to evaluate whether the generated parent-child pairs have the same relation as the given seed parent-child pairs. After collecting these answers from the annotators, we simply use majority voting to label the pairs. We then use these annotated data as the gold standard. The labeled dataset is available at: http://bit.ly/2Jbilte.

5.3.2 Evaluation Results. Table 3 shows both the ancestor-based and edge-based precision/recalls as well as F1-scores of different methods. We can see that HiExpan achieves the best overall performance, and outperforms other methods, especially in terms of the precision. Comparing the performance of HiExpan, NoREPEL, and NoGTO, we see that both the REPEL and the taxonomy global optimization modules play important roles in improving the quality of the generated taxonomy. Specifically, REPEL learns more discriminative representations by iteratively letting the distributional module and pattern module mutually enhance each other, and the taxonomy global optimization module leverages the global information from the entire taxonomy tree structure. In addition, HiExpan resolves the "conflicts" at the end of each tree expansion iteration by cutting many nodes on a currently expanded taxonomy. This leads HiExpan to generate a smaller tree comparing with the one generated by HSetExpan, given that both methods running the same number of iterations. However, we can see that HiExpan still beats HSetExpan on Wiki dataset and PubMed-CVD dataset, in terms of the recall. This further demonstrates the effectiveness of our HiExpan framework.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we introduce a new research problem *task-guided taxonomy construction* and propose a novel expansion-based framework HiExpan for solving it. HiExpan views all children under a taxonomy node as a coherent set and builds the taxonomy by recursively expanding these sets. Furthermore, HiExpan incorporates a weakly-supervised relation extraction module to infer parent-child relation and adjusts the taxonomy tree by optimizing its global structure. Experimental results on three public datasets corroborate the effectiveness of HiExpan.

As a first-punch solution for constructing a task-guided taxonomy, HiExpan can be improved in many ways. First, we find in the experiments that HiExpan places synonyms at the same level of taxonomy since they share same semantic meanings and appear in similar contexts. These synonyms will make generated taxonomy less informative, with reduced overall quality. It is an interesting direction to extend HiExpan to automatically discover and resolve those synonyms. Further, as an expansion-based framework, HiExpan may facilitate interactive user guidance in taxonomy construction, which is another interesting task in the future.

ACKNOWLEDGEMENTS

This research is sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). We thank Xinwei He, Yunyi

Zhang, and Luyu Gao for helping label the datasets and providing valuable comments and discussions. Also, we would like to thank anonymous reviewers for valuable feedback.

REFERENCES

[1] Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *ACM DL*.

[2] Luis Espinosa Anke, José Camacho-Collados, Claudio Delli Bovi, and Horacio Saggion. 2016. Supervised Distributional Hypernym Discovery via Domain Adaptation. In *EMNLP*.

[3] Mohit Bansal, David Burkett, Gerard de Melo, and Dan Klein. 2014. Structured Learning for Taxonomy Induction with Belief Propagation. In *ACL*.

[4] Marco G Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung chieh Shan. 2012. Entailment above the word level in distributional semantics. In *EACL*.

[5] 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*.

[6] Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2). In *SemEval-2016*.

[7] Zhe Chen, Michael Cafarella, and HV Jagadish. 2016. Long-tail vocabulary dictionary extraction from the web. In *WSDM*.

[8] Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning Semantic Hierarchies via Word Embeddings. In *ACL*.

[9] Amit Gupta, Rémi Lebret, Hamza Harkous, and Karl Aberer. 2017. Taxonomy Induction Using Hypernym Subsequences. In *CIKM*.

[10] Yeye He and Dong Xin. 2011. SEISA: set expansion by iterative similarity aggregation. In *WWW*.

[11] Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *COLING*.

[12] Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. 2017. Understand Short Texts by Harvesting and Analyzing Semantic Knowledge. *TKDE* (2017).

[13] Meng Jiang, Jingbo Shang, Taylor Cassidy, Xiang Ren, Lance M. Kaplan, Timothy P. Hanratty, and Jiawei Han. 2017. MetaPAD: Meta Pattern Discovery from Massive Text Corpora. In *KDD*.

[14] Zornitsa Kozareva and Eduard H. Hovy. 2010. A Semi-Supervised Method to Learn and Construct Taxonomies Using the Web. In *EMNLP*.

[15] Dekang Lin. 1998. An Information-Theoretic Definition of Similarity. In *ICML*.

[16] Xiao Ling and Daniel S. Weld. 2012. Fine-Grained Entity Recognition. In *AAAI*.

[17] Jialu Liu, Jingbo Shang, Chi Chiu Wang, Xiang Ren, and Jiawei Han. 2015. Mining Quality Phrases from Massive Text Corpora. *SIGMOD* (2015).

[18] Anh Tuan Luu, Jung jae Kim, and See-Kiong Ng. 2014. Taxonomy Construction Using Syntactic Contextual Evidence. In *EMNLP*.

[19] Anh Tuan Luu, Yi Tay, Siu Cheung Hui, and See-Kiong Ng. 2016. Learning Term Embeddings for Taxonomic Relation Identification Using Dynamic Weighting Neural Network. In *EMNLP*.

[20] Yuning Mao, Xiang Ren, Jiaming Shen, Xiaotao Gu, and Jiawei Han. 2018. End-to-End Reinforcement Learning for Automatic Taxonomy Induction. In *ACL*.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.

[22] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/IJCNLP*.

[23] Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *EMNLP-CoNLL*.

[24] Roberto Navigli and Paola Velardi. 2010. Learning Word-Class Lattices for Definition and Hypernym Extraction. In *ACL*.

[25] Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A Graph-Based Algorithm for Inducing Lexical Taxonomies from Scratch. In *IJCAI*.

[26] James B. Orlin. 1996. A polynomial time primal network simplex algorithm for minimum cost flows. In *SODA*.

[27] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-Scale Distributional Similarity and Entity Set Expansion. In *EMNLP*.

[28] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*.

[29] Meng Qu, Xiang Ren, Yu Zhang, and Jiawei Han. 2018. Weakly-supervised Relation Extraction by Pattern-enhanced Embedding Learning. In *WWW*.

[30] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. In *HLT-NAACL*.

[31] Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What Is This, Anyway: Automatic Hypernym Discovery. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*.

[32] Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet Selective: Supervised Distributional Hypernymy Detection. In *COLING*.

[33] Xin Rong, Zhe Chen, Qiaozhu Mei, and Eytan Adar. 2016. Egoset: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion. In *WSDM*.

[34] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei Han. 2018. Automated Phrase Mining from Massive Text Corpora. *TKDE* (2018).

[35] Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. SetExpan: Corpus-Based Set Expansion via Context Feature Selection and Rank Ensemble. In *ECML/PKDD*.

[36] Bei Shi, Zhenzhong Zhang, Le Sun, and Xianpei Han. 2014. A Probabilistic Co-Bootstrapping Method for Entity Set Expansion. In *COLING*.

[37] Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen. 2010. Corpus-based Semantic Class Mining: Distributional vs. Pattern-Based Approaches. In *COLING*.

[38] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning Syntactic Patterns for Automatic Hypernym Discovery. In *NIPS*.

[39] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks. In *KDD*.

[40] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*.

[41] Simon Tong and Jeff Dean. 2008. System and methods for automatically creating lists. (2008). US Patent 7,350,187.

[42] Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. OntoLearn Reloaded: A Graph-Based Algorithm for Taxonomy Induction. *Computational Linguistics* (2013).

[43] Chengyu Wang, Xiaofeng He, and Aoying Zhou. 2017. A Short Survey on Taxonomy Learning from Text Corpora: Issues, Resources and Recent Advances. In *EMNLP*.

[44] Richard C. Wang and William W. Cohen. 2007. Language-Independent Set Expansion of Named Entities Using the Web. In *ICDM*.

[45] Richard C. Wang and William W. Cohen. 2008. Iterative Set Expansion of Named Entities Using the Web. In *ICDM*.

[46] Julie Weeds, Daoud Clarke, Jeremy Reffin, David J. Weir, and Bill Keller. 2014. Learning to Distinguish Hypernyms and Co-Hyponyms. In *COLING*.

[47] Julie Weeds, David J. Weir, and Diana McCarthy. 2004. Characterising Measures of Lexical Distributional Similarity. In *COLING*.

[48] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD Conference*.

[49] Shuo Yang, Lei Zou, Zhongyuan Wang, Jun Yan, and Ji-Rong Wen. 2017. Efficiently Answering Technical Questions - A Knowledge Graph Approach. In *AAAI*.

[50] Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning Term Embeddings for Hypernymy Identification. In *IJCAI*.

[51] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks. In *EMNLP*.

[52] Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander J. Smola. 2014. Taxonomy discovery for personalized recommendation. In *WSDM*.

[53] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with Local and Global Consistency. In *NIPS*.