Enhancing Taxonomy Completion with Concept Generation via Fusing Relational Representations

Qingkai Zeng, Jinfeng Lin, Wenhao Yu, Jane Cleland-Huang, Meng Jiang
Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA
{qzeng,jlin6,wyu1,jhuang13,mjiang2}@nd.edu,

ABSTRACT

Automatically constructing taxonomy supports many applications in e-commerce, web search, and question answering. Existing taxonomy expansion or completion methods assumed that new concepts were accurately extracted and their embedding vectors had been learned from text corpus. However, one critical and fundamental challenge in fixing the incompleteness of taxonomies is actually the incompleteness of extracted concepts, especially for those whose names have multiple words – they usually have very low frequency in the corpus. To resolve the bottleneck of the extraction-based methods, we propose GenTaxo to enhance taxonomy completion by identifying the positions in existing taxonomies that need new concepts and generating the concept names. Instead of relying on the corpus for concept embeddings, GenTaxo learns the contextual embeddings from both graph-based and language-based relational information at the positions; and it leverages the corpus for pre-training a concept name generator. Experimental results demonstrate that GenTaxo improves the completeness of real-world taxonomies over existing methods.

KEYWORDS

Taxonomy Completion, Concept Generation

ACM Reference Format:
Qingkai Zeng, Jinfeng Lin, Wenhao Yu, Jane Cleland-Huang, Meng Jiang. 2021. Enhancing Taxonomy Completion with Concept Generation via Fusing Relational Representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467308

1 INTRODUCTION

Taxonomies have been widely used to enhance the performance of many applications such as question answering [34, 35], query understanding [10, 15], and personalized recommendation [11]. With the influx of new content in evolving applications, it is necessary to curate these taxonomies to include emergent concepts. However, manual curation is labor-intensive and time-consuming. To this end, many recent studies aim to automatically expand or complete an existing taxonomy. Given a new concept, Shen *et al.* measured the likelihood for each existing concept in the taxonomy to be its hypernym and then added it as a new leaf node [26]. Manzoor *et al.* extended the measurement to be taxonomic relatedness with implicit relational semantics [17]. Zhang *et al.* predicted the position of new concepts considering both hypernyms and hyponyms [41]. In all of these methods, the distance between concepts was measured using their embedding vectors learned from some text corpus, with the underlying assumption that new concepts could be extracted accurately and found frequently in the corpus.

We argue that such an assumption is inappropriate in real-world taxonomies, with examples based on the frequency of the concepts in Medical Subject Headings (MeSH), a widely-used taxonomy of approximately 30,000 terms that is updated annually and manually, in a large-scale public text corpus of 30 million paper abstracts (about 6 billion tokens) from the PubMed database. We observe that many concepts that have multiple words appear fewer than 100 times in the corpus (as depicted by the red outlined nodes in Figure 1) and around half of the terms cannot be found in the corpus (see Table 1 in Section 4). Concept extraction tools [39] often fail to find them at the top of a list of over half a million concept candidates; and there is insufficient data to learn their embedding

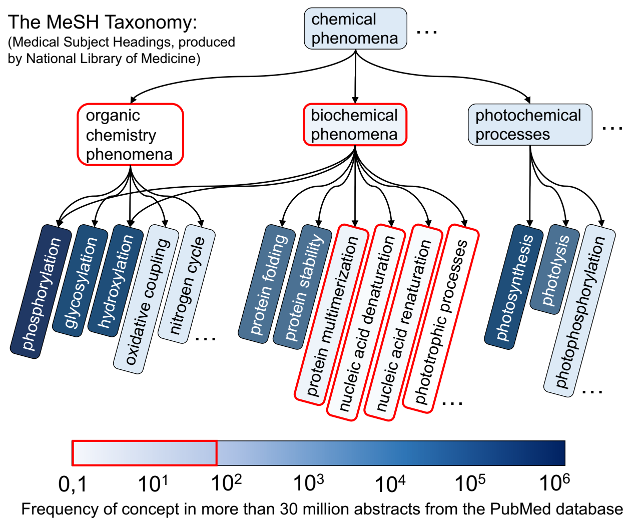


Figure 1: A great number of concepts that are desired to be in the taxonomies (e.g., MeSH) are very rare in even large-scale text corpus (e.g., 30 million PubMed abstracts). So it is hard to extract these concepts or learn their embedding vectors. Concept names need to be generated rather than extracted to fix the incompleteness of taxonomies.

§ Code and data is available at https://github.com/QingkaiZeng/GenTaxo.

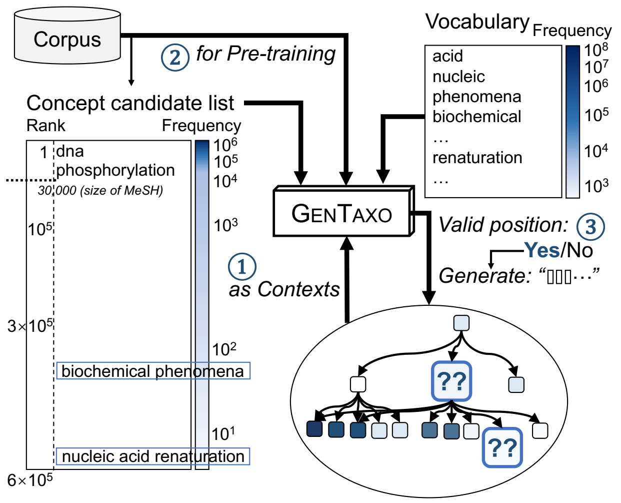


Figure 2: Existing methods extracted a concept candidate list from the corpus and assumed that the top concepts (high frequency or quality) could be added into existing taxonomies. However, frequent concepts may not be qualified and many qualified concepts are very rare. We find that though the concept names are not frequent, their words are. Our idea is to generate concept names at candidate positions with relational contexts to complete taxonomies. Our solution GenTaxo is pre-trained on the corpus and learns to generate the names token by token on the existing taxonomy.

vectors. The incompleteness of concepts is a critical challenge in taxonomy completion, and has not yet been properly studied.

Despite the low frequency of many multi-gram concepts in a text corpus, the frequency of individual words is naturally much higher. Inspired by recent advances in text generation [20, 36], we propose a new task, "taxonomy generation", that identifies whether a new concept fits in a candidate position within an existing taxonomy, and if yes, generates the concept name token by token.

The key challenge lies in the lack of information for accurately generating the names of new concepts when their full names do not (frequently) appear in the text corpus. To address this challenge, our framework for enhancing taxonomy completion, called GenTaxo, has the following novel design features (see Figure 2 and 3):

First, GenTaxo has an encoder-decoder scheme that *learns to generate* any concept in the existing taxonomy using a self-supervised manner. Suppose an existing concept v is masked. Two types of encoders are recruited to fuse both sentence-based and graph-based representations of the masked position learned from the relational contexts. One is a sequence encoder that learns the last hidden states of a group of sentences that describe the relations such as "v_p is a class of" and "v_c is a subclass of", when v_p and v_c are parent and child concepts of the position, respectively. The other is a graph encoder that aggregates information from two-hop neighborhoods in the top-down subgraph (with v at the bottom) and bottom-up subgraph (with v at the top). The fused representations are fed into a GRU-based decoder to generate v's name token by token with a special token [EOS] at the end. With the context fusion, the decoding process can be considered as completing a group of sentences (e.g., "v_p is a class of" and "v_c is a subclass of") with the same ending term while simultaneously creating a hyponym/hypernym node in the top-down/bottom-up subgraphs.

Second, GenTaxo is pre-trained on a large-scale corpus to predict tokens in concept names. The pre-training task is very similar to the popular Mask Token Prediction task [6] except that the masked token must be a token of a concept that appears in the existing taxonomy and is found in a sentence of the corpus.

Third, GenTaxo performs the task of candidate position classification simultaneously with concept name generation. It has a binary classifier that uses the final state of the generated concept name to predict whether the concept is needed at the position. We adopt negative sampling to create "invalid" candidate positions in the existing taxonomy. The classifier is attached after the name generation (not before it) because the quality of the generated name indicates the need for a new concept at the position.

Furthermore, we develop GenTaxo++ to enhance extraction-based methods, when a set of new concepts, though incomplete, needs to be added to existing taxonomies (as described in the existing studies). In GenTaxo++, we apply GenTaxo to generate concept names in order to expand the set of new concepts. We then use the extraction-based methods with the expanded set to improve the concept/taxonomy completeness.

The main contributions of this work are summarized as follows:

- We propose a new taxonomy completion task that identifies valid positions to add new concepts in existing taxonomies and generates their names token by token.
- We design a novel framework GenTaxo that has three novel designs: (1) an encoder-decoder scheme that fuses sentence-based and graph-based relational representations and learns to generate concept names; (2) a pre-training process for concept token prediction; and (3) a binary classifier to find valid candidate positions. Furthermore, GenTaxo++ is developed to enhance existing extraction-based methods with the generated concepts, when a set of new concepts are available.
- Experiments on *six* real-world taxonomy data sets demonstrate that (1) concept generation can significantly improve the recall and completeness of taxonomies; (2) even some concepts that do not appear in the corpus can be accurately generated token by token at valid positions; and (3) fusing two types of relational representations is effective.

2 PROBLEM DEFINITION

Traditionally, the input of the taxonomy completion task has included two parts [17, 26, 37, 41]: (1) an existing taxonomy $\mathcal{T}_0 = (\mathcal{V}_0, \mathcal{E}_0)$ and (2) a set of new concepts C that have been either manually given or accurately extracted from text corpus $\mathcal{D}$. The overall goal is completing the existing taxonomy $\mathcal{T}_0$ into a larger one $\mathcal{T} = (\mathcal{V}_0 \cup C, \mathcal{E}')$.

We argue that existing technologies, which defined the task as above, could fail when the concepts cannot be extracted (due to their low frequency) or cannot be found in the corpus. We aim to mitigate this problem by *generating* the absent concepts to achieve better taxonomy completion performance. Suppose $v = (t_1, t_2, \ldots, t_{T_v})$, where $t_i \in \mathcal{W}$ is the i-th token in the name of concept v, T_v is the number of v's tokens (length of v), $\mathcal{W}$ is the token vocabulary which includes words and punctuation marks (e.g., comma).

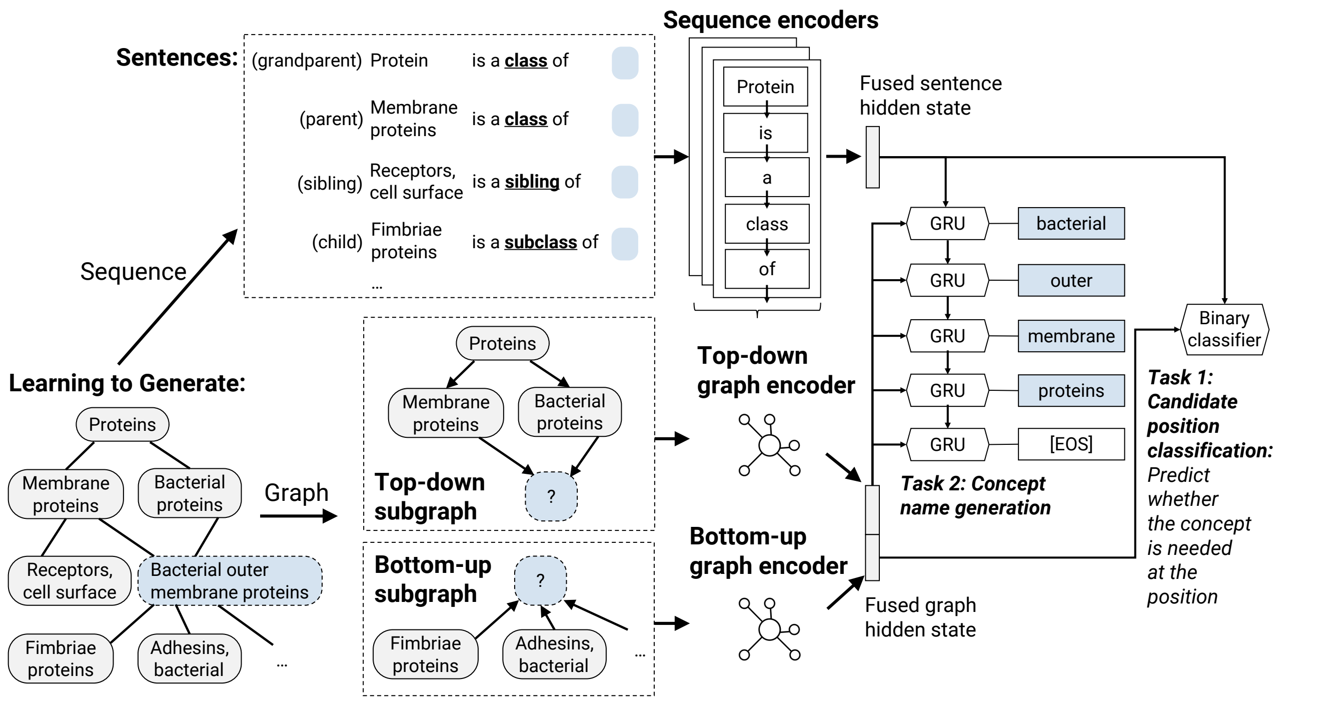

Sentences:
(grandparent) Protein is a class of
(parent) Membrane proteins is a class of
(sibling) Receptors, cell surface is a sibling of
(child) Fimbriae proteins is a subclass of
Sequence
Learning to Generate:
Proteins
Membrane proteins
Bacterial proteins
Receptors, cell surface
Bacterial outer membrane proteins
Fimbriae proteins
Adhesins, bacterial
Graph
Top-down subgraph
Bottom-up subgraph
Sequence encoders
Protein
is
a
class
of
Fused sentence hidden state
Top-down graph encoder
Bottom-up graph encoder
Fused graph hidden state
GRU
bacterial
outer
membrane
proteins
[EOS]
Task 2: Concept name generation
Binary classifier
Task 1: Candidate position classification: Predict whether the concept is needed at the position

Figure 3: Architecture of GenTaxo. Suppose the concept "bacterial outer membrane proteins" is masked (Left bottom). Given the masked position in the taxonomy, the model learns to generate the name of the concept. It has two types of encoders to represent the relational contexts of the position. Sequence encoders learn a group of sentences that describe the relations with the masked concept (Middle top). Graph encoders learn the masked concept's embedding vector by aggregating information from its top-down and bottom-up subgraphs (Middle bottom). The fused sentence-based and graph-based hidden states are used to predict whether the masked position is valid, i.e., needs a concept (Rightest: Task 1). They are also fed into GRU-based decoders to generate the concept's name (Right: Task 2).

Definition 2.1 (Taxonomy). We follow the definition of taxonomy in [41]. A taxonomy $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is a directed acyclic graph where each node $v \in \mathcal{V}$ represents a concept (i.e., a word or a phrase) and each directed edge $\langle u,\ v\rangle \in \mathcal{E}$ implies a relation between a parent-child pair such as "is a type of" or "is a class of". We expect the taxonomy to follow a hierarchical structure where concept u is the most specific concept that is more general than concept v. Note that a taxonomy node may have multiple parents.

In most of the taxonomies, the parent-child relation can be specified as a hypernymy relation between concept u and v, where u is the hypernym (parent) and v is the hyponym (child). We use the terms "hypernym" and "parent", "hyponym" and "child" interchangeably throughout the paper.

Definition 2.2 (Candidate Position). Given two sets of concepts $\mathcal{V}_p, \mathcal{V}_c \subset \mathcal{V}_0$, if v_c is one of the descendants of v_p in the existing taxonomy for any pair of concepts $v_p \in \mathcal{V}_p$ and $v_c \in \mathcal{V}_c$, then a candidate position acts as a placeholder for a new concept v. It becomes a valid position when v satisfies (1) v_p is a parent of v and (2) v_c is a child of v. When it is a valid position, we add edges $\langle v_p,\ v\rangle$ and $\langle v,\ v_c\rangle$ and delete redundant edges to update $\mathcal{E}'$.

We reduce the task of generating concept names for taxonomy completion as the problem below: Given text corpus $\mathcal{D}$ and a candidate position on an existing taxonomy $\mathcal{T}_0$, predict whether the position is valid, and if yes, generate the name of the concept v from the token vocabulary $\mathcal{W}$ (extracted from $\mathcal{D}$) to fill in the position.

3 GENTAXO: GENERATE CONCEPT NAMES

Overall architecture. Figure 3 presents the architecture of GenTaxo. The goal is to learn from an existing taxonomy to identify valid candidate positions and generate concept names at the positions. Given a taxonomy, it makes *valid* candidate positions by masking an existing concept in the taxonomy; it also makes *invalid* candidate positions using negative sampling strategies which will be discussed in Section 3.2.

Here we focus on the valid positions and masked concepts. As shown in the left bottom side of Figure 3, suppose the term "bacterial outer membrane proteins" is masked. GenTaxo adopts an encoder-decoder scheme in which the encoders represent the taxonomic relations in forms of sentences and subgraphs (see the middle part of the figure) and the decoders perform two tasks to achieve the goal (see the right-hand side of the figure).

3.1 Encoders: Representing Taxonomic Relations

Here we introduce how to represent the taxonomic relations into sentences and subgraphs, and use sequence and graph encoders to generate and fuse the relational representations.

3.1.1 Representing Taxonomic Relations as Sentences. Given a taxonomy $\mathcal{T}_0 = (\mathcal{V}_0, \mathcal{E}_0)$ and a candidate position for masked concept

v, we focus on three types of taxonomic relations between the masked concept v and some concepts $u \in \mathcal{V}_0$:

(1) *Parent or ancestor.* Suppose there is a sequence of nodes $(v_1, v_2, \ldots, v_l)$ where $v_1 = u$, $v_l = v$, and $\langle v_i,\ v_{i+1}\rangle \in \mathcal{E}_0$ for any $i = 1, \ldots, l-1$. In other words, there is a path from u to v in $\mathcal{T}_0$. When $l = 1$, u is a parent of v; when $l \geq 2$, u is an ancestor. We denote $u = (t_1, t_2, \ldots, t_{T_u})$, where T_u is the length of concept u and $\{t_i\}_{i=1}^{T_u}$ are its tokens. Then we create a sentence with the template below:
"$t_1\ t_2\ \ldots\ t_{T_u}$ is a class of [MASK]"
And we denote the set of parent or ancestor nodes of v (i.e., all possible u as described above) by $\mathcal{V}_p(v)$.

(2) *Child or descendant.* Similar as above expect $v_1 = v$ and $v_l = u$, we have a path from v to u. When $l = 1$, u is a child of v; when $l \geq 2$, u is a descendant. Then we create a sentence:
"$t_1\ t_2\ \ldots\ t_{T_u}$ is a subclass of [MASK]"
We denote the set of child or descendant nodes of v by $\mathcal{V}_c(v)$.

(3) *Sibling.* If there is a concept $p \in \mathcal{V}_0$ and two edges $\langle p,\ v\rangle$ and $\langle p,\ u\rangle$, then u is a sibling node of v. We create a sentence:
"$t_1\ t_2\ \ldots\ t_{T_u}$ is a sibling of [MASK]"

Given the candidate position for concept v, we can collect a set of sentences from its relational contexts, denoted by $\mathcal{S}(v)$. Examples can be found in the left top dashed box of Figure 3.

Sentence encoders: We apply neural network models that capture sequential patterns in the sentences and create the hidden states at the masked position. The hidden states are vector representations that encode the related concept node u and the relational phrase to indicate the masked concept v. The hidden states will later support the tasks of concept name generation and candidate position classification. We provide *three* options of the sentence encoders: BiGRU, Transformer, and SciBERT [4]. Their descriptions can be found in Appendix.

3.1.2 *Representing Taxonomic Relations as Subgraphs.*

The relations between the masked concept v and its surrounding nodes can be represented as two types of subgraphs, because of the hierarchical structure of $\mathcal{T}_0$ (see the middle bottom part of Figure 3):

(1) Top-down subgraph: It consists of all parent and ancestor nodes of v, denoted by $G_{\text{down}}(v) = \{\mathcal{V}_{\text{down}}(v), \mathcal{E}_{\text{down}}(v)\}$, where $\mathcal{V}_{\text{down}}(v) = \mathcal{V}_p(v)$ and $\mathcal{E}_{\text{down}}(v) = \{\langle v_i,\ v_j\rangle \in \mathcal{E} \mid v_i, v_j \in \mathcal{V}_{\text{down}}(v)\}$. The role of v is the very specific concept of any other concepts in this subgraph. So, the vector representations of this masked position should be aggregated in a top-down direction. The aggregation indicates the relationship of being from more general to more specific.

(2) Bottom-up subgraph: Similarly, it consists of all child and descendant nodes of v, denoted by $G_{\text{up}}(v) = \{\mathcal{V}_{\text{up}}(v), \mathcal{E}_{\text{up}}(v)\}$, where $\mathcal{V}_{\text{up}}(v) = \mathcal{V}_c(v)$ and $\mathcal{E}_{\text{up}}(v) = \{\langle v_i,\ v_j\rangle \in \mathcal{E} \mid v_i, v_j \in \mathcal{V}_{\text{up}}(v)\}$. The representations of this masked position should be aggregated in a bottom-up direction. The aggregation indicates the relationship of being from specific to general.

Graph encoders: We adopt two independent graph neural networks (GNNs) to encode the relational contexts in $G_{\text{down}}(v)$ and $G_{\text{up}}(v)$ separately. Given a subgraph G, GNN learns the graph-based hidden state of *every node* on the final layer through the graph structure, while we will use that of the node v for next steps.

v was specifically denoted for the masked concept node. In this paragraph, we denote any node on G by v for convenience. We initialize the vector representations of v randomly, denoted by $\mathbf{v}^{(0)}$. Then, the K-layered GNN runs the following functions to generate the representations of v on the k-th layer ($k = 1, \ldots, K$):

$$\begin{aligned}\mathbf{a}^{(k-1)} &= \textsc{Aggregate}_k\left(\left\{\left(\mathbf{u}^{(k-1)}\right) : \forall u \in \mathcal{N}(v) \cup G\right\}\right),\\ \mathbf{v}^{(k)} &= \textsc{Combine}_k\left(\mathbf{v}^{(k-1)}, \mathbf{a}^{(k-1)}\right),\end{aligned}$$

where $\mathcal{N}(v)$ is the set of neighbors of v in G.

There are a variety of choices for $\textsc{Aggregate}_k(\cdot)$ and $\textsc{Combine}_k(\cdot)$. For example, $\textsc{Aggregate}_k(\cdot)$ can be mean pooling, max pooling, graph attention, and concatenation [9, 13]. One popular choice for $\textsc{Combine}_k(\cdot)$ is graph convolution: $\mathbf{v}^{(k)} = \sigma\left(\mathbf{W}^{(k)}\left(\mathbf{v}^{(k-1)} \oplus \mathbf{a}^{(k-1)}\right)\right)$, where $\mathbf{W}^{(k)}$ is the weight matrix for linear transformation on the k-th layer and $\sigma(\cdot)$ is a nonlinear function.

For the masked concept v, we finally return the graph-based hidden state $\mathbf{v}_G$ at K-th layer, i.e., $\mathbf{v}_G = \mathbf{v}^{(K)}$.

3.1.3 *Representations Fusion.*

As aforementioned, we have a set of sentence-based hidden states $\{\mathbf{h}(s)\}_{s \in \mathcal{S}(v)}$ from the sentence encoder; also, we have two graph-based hidden states $\mathbf{v}_{G_{\text{down}}}$ and $\mathbf{v}_{G_{\text{up}}}$ from the graph encoder. In this section, we present how to fuse these relational representations for decoding concept names.

Fusing sentence-based hidden states: We use the self-attention mechanism to fuse the hidden states with a weight matrix $\mathbf{W}_{\text{seq}}$:

$$\mathbf{h}_{\text{seq}} = \sum_{s \in \mathcal{S}(v)} w(s) \cdot \mathbf{h}(s), \text{ where } w(s) = \frac{\exp(\sigma(\mathbf{W}_{\text{seq}}\mathbf{h}(s)))}{\sum\limits_{s' \in \mathcal{S}(v)} \exp(\sigma(\mathbf{W}_{\text{seq}}\mathbf{h}(s')))}.$$

Fusing graph-based hidden states: We adopt a learnable weighted sum to fuse $\mathbf{v}_{G_{\text{down}}}$ and $\mathbf{v}_{G_{\text{up}}}$ with weight matrices $\mathbf{W}_{\text{down}}$ and $\mathbf{W}_{\text{up}}$:

$$\mathbf{v}_{\text{graph}} = \beta \cdot \mathbf{v}_{G_{\text{down}}} + (1 - \beta) \cdot \mathbf{v}_{G_{\text{up}}},$$

where $\beta = \frac{\exp(\sigma(\mathbf{W}_{\text{down}} \cdot \mathbf{v}_{G_{\text{down}}}))}{\exp(\sigma(\mathbf{W}_{\text{down}} \cdot \mathbf{v}_{G_{\text{down}}})) + \exp(\sigma(\mathbf{W}_{\text{up}} \cdot \mathbf{v}_{G_{\text{up}}}))}$.

Fusing the fused sentence- and graph-based hidden states: Given a masked concept v, there are a variety of strategies to fuse $\mathbf{h}_{\text{seq}}$ and $\mathbf{v}_{\text{graph}}$: $\mathbf{v}_{\text{fuse}} = \textsc{Fuse}(\mathbf{h}_{\text{seq}}, \mathbf{v}_{\text{graph}})$, such as mean pooling, max pooling, attention, and concatenation. Take concatenation as an example: $\textsc{Fuse}(\mathbf{a}, \mathbf{b}) = \mathbf{a} \oplus \mathbf{b}$. We will investigate these fusing strategies in the ablation study at the experiments section.

3.2 Decoders: Identifying Valid Positions and Generating Concept Names

Given a masked position v, we now have the fused representations of its relational contexts $\mathbf{v}_{\text{fuse}}$ from the above encoders. We perform two tasks jointly to complete the taxonomy: one is to identify whether the candidate position is valid or not; the other is to generate the name of concept for the masked position.

3.2.1 *Task 1: Candidate Position Classification.*

Given a candidate position, this task predicts whether it is valid, i.e., a concept is needed. If the position has a masked concept in the existing taxonomy, it is a valid position; otherwise, this invalid position is produced by negative sampling. We use a three-layer feed forward neural network (FFNN) to estimate the probability of being a valid position with the fused representations: $p_{\text{valid}}(v) = \text{FFNN}(\mathbf{v}_{\text{fuse}})$. The loss term on the particular position v is based on cross entropy:

$$L_1(\Theta; v) = -(y_v \cdot \log(p_{\text{valid}}(v)) + (1 - y_v) \cdot \log(1 - p_{\text{valid}}(v))),$$

where $y_v = 1$ when v is valid as observed; otherwise, $y_v = 0$.

Negative sampling: Suppose a valid position is sampled by masking an existing concept $v \in \mathcal{V}_0$, whose set of parent/ancestor nodes is denoted by $\mathcal{V}_p(v)$ and set of child/descendant nodes is denoted by $\mathcal{V}_c(v)$. We create r_{neg} negative samples (i.e., invalid positions) by replacing one concept in either $\mathcal{V}_p(v)$ or $\mathcal{V}_c(v)$ by a random concept in $\mathcal{V}_{\text{neg}}(v) = \mathcal{V}_0 \setminus (\mathcal{V}_p(v) \cup \mathcal{V}_c(v) \cup \{v\})$. We will investigate the effect of negative sampling ratio r_{neg} in experiments.

3.2.2 Task 2: Concept Name Generation. We use a Gated Recurrent Unit (GRU)-based decoder to generate the name of concept token by token which is a variable-length sequence $v = (v_1, v_2, \ldots, v_{T_v})$. As shown at the right of Figure 3, we add a special token [EOS] to indicate the concept generation is finished.

We initialize the hidden state of the decoder as $\mathbf{v}_0 = \mathbf{h}_{\text{seq}}$. Then the conditional language model works as below:

$$\mathbf{v}_t = \text{GRU}(v_{t-1}, \mathbf{v}_{t-1} \oplus \mathbf{v}_{\text{graph}}),$$
$$p(v_t | v_{t-1}, \ldots, v_1, \mathbf{v}_{\text{fuse}}) = \textsc{Readout}(v_{t-1}, \mathbf{v}_t, \mathbf{v}_{\text{fuse}}),$$

where $\textsc{Readout}(\cdot)$ is a nonlinear multi-layer function that predicts the probability of token v_t. Then this task can be regarded as a sequential multi-class classification problem. The loss term is:

$$L_2(\Theta; v) = -\log(p(v|\mathbf{v}_{\text{fuse}})) = -\sum_{t=1}^{T_v} \log(p(v_t | v_{<t}, \mathbf{v}_{\text{fuse}})),$$

where v is the target sequence (i.e., the concept name) and $\mathbf{v}_{\text{fuse}}$ is the fused relational representations of the masked position.

Pre-training: To perform well in Task 2, a model needs the ability of predicting tokens in a concept's name. So, we pre-train the model with the task of predicting masked concept's tokens (MCT) in sentences of text corpus $\mathcal{D}$. We find all the sentences in $\mathcal{D}$ that contain at least one concept in the existing taxonomy $\mathcal{T}_0$. Given a sentence $X = (x_1, x_2, \ldots, x_n)$ where $(x_s, \ldots, x_e) = v = (v_1, \ldots, v_{T_v})$ is an existing concept. Here a token x_m is masked ($s \le m \le e$) and predicted using all past and future tokens. The loss function is:

$$L_{\text{MCT}}(\Theta; x_m) = -\log(p(x_m | x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_n)).$$

3.2.3 Joint Training. The joint loss function is a weighted sum of the loss terms of the above two tasks:

$$L = \sum_{v \in \mathcal{V}_0} (L_1(\Theta; v) + \lambda L_2(\Theta; v)) + \sum_{i=1}^{r_{\text{neg}}} \mathbb{E}_{v_{\text{neg}} \sim \mathcal{V}_{\text{neg}}(v)} L_1(\Theta; v_{\text{neg}}),$$

where λ is introduced as a hyper-parameter to control the importance of Task 2 concept name generation.

3.3 GenTaxo++: Enhancing Extraction-based Methods with GenTaxo

While GenTaxo is designed to replace the process of extracting new concepts by concept generation, GenTaxo++ is an alternative solution when the set of new concepts is given and of high quality. GenTaxo++ can use any extraction-based method [17, 24–26, 32, 37, 40, 41] as the main framework and iteratively expand the set of new concepts using concept generation (i.e., GenTaxo) to continuously improve the taxonomy completeness. We choose TaxoExpan as the extraction-based method GenTaxo++ [26].

	#Concepts	#Tokens	#Edges	Depth
Computer Science domains (Corpus: DBLP)				
MAG-CS	29,484	16,398	46,144	6
(found in corpus)	18,338 (62.2%)	13,914 (84.9%)		
OSConcepts	984	967	1,041	4
DroneTaxo	263	247	528	4
Biology/Biomedicine domains: (Corpus: PubMed)				
MeSH	29,758	22,367	40,186	15
(found in corpus)	14,164 (47.6%)	22,193 (99.2%)		
SemEval-Sci	429	573	452	8
SemEval-Env	261	317	261	6

Table 1: Statistics of six taxonomy data sets.

The details of GenTaxo++ are as follows. We start with an existing taxonomy $T_0 = (\mathcal{V}_0, \mathcal{E}_0)$ and a *given* set of new concepts C_0. During the i-th iteration, we first use GenTaxo to *generate* a set of new concepts, and then expand the set of new concepts and use the extraction-based method to update the taxonomy ($i \ge 1$):

$$\begin{aligned} T'_{i-1} = (\mathcal{V}'_{i-1}, \mathcal{E}'_{i-1}) &\leftarrow \textsc{GenTaxo}(T_{i-1}), \\ C_i &= (C_{i-1} \cup \mathcal{V}'_{i-1}) \setminus \mathcal{V}_{i-1}, \\ T_i &\leftarrow \textsc{ExtractionMethod}(T_{i-1}, C_i). \end{aligned}$$

The iterative procedure terminates when $C_i == \emptyset$. In this procedure, we have two hyperparameters:

(1) Concept quality threshold τ: GenTaxo predicts the probability of being a valid position $p_{\text{valid}}(v)$ which can be considered as the quality of the generated concept v. We have a constraint on adding generated concepts to the set: $p_{\text{valid}}(v) \ge \tau$, for any $v \in C_i$. When τ is bigger, the process is more cautious: fewer concepts are added each iteration.
(2) Maximum number of iterations max_{iter}: An earlier stop is more cautious but may cause the issue of low recall.

We investigate the effect of these hyperparameters in experiments.

4 EXPERIMENTAL SETTINGS

In this work, we propose GenTaxo and GenTaxo++ to complete taxonomies by concept generation. We conduct experiments to answer the following research questions (RQs):

- **RQ1:** Do the proposed approaches perform better than existing methods on *taxonomy completion*?
- **RQ2:** Given *valid* positions in an existing taxonomy and a corresponding large text corpus, which method *produce more accurate concept names*, the proposed concept generation or existing extraction-and-filling methods?
- **RQ3:** How do the components (e.g., sequence encoders, graph encoders, representation fusion) and hyperparameters (e.g., r_{neg}, τ, $\max_{\text{iter}}$) impact the performance of GenTaxo?

4.1 Datasets

Table 1 presents the statistics of six taxonomies from two different domains we use to evaluate the taxonomy completion methods:

- **MAG-CS [28]:** The Microsoft Academic Graph (MAG) taxonomy has over 660 thousand scientific concepts and more than 700 thousand taxonomic relations. We follow the processing in TaxoExpan [26] to select a partial taxonomy under the "computer science" (CS) node.
- **OSConcepts [22]:** It is a taxonomy manually crated in a popular textbook "Operating System Concepts" for OS courses.

MeSH:	**Acc**	Acc-Uni	Acc-Multi
HiExpan [27]	9.89	18.28	8.49
TaxoExpan [26]	16.32	28.35	14.31
Graph2Taxo [24]	10.35	19.02	8.90
STEAM [37]	17.04	27.61	15.26
ARBORIST [17]	16.01	27.91	14.19
TMN [41]	16.53	27.52	14.85
GenTaxo ($r_{neg}=0$)	26.72	28.13	**26.29**
GenTaxo ($r_{neg}=0.15$)	**26.93**	**31.34**	26.07

Table 2: Evaluating the quality of generated concepts (RQ2).

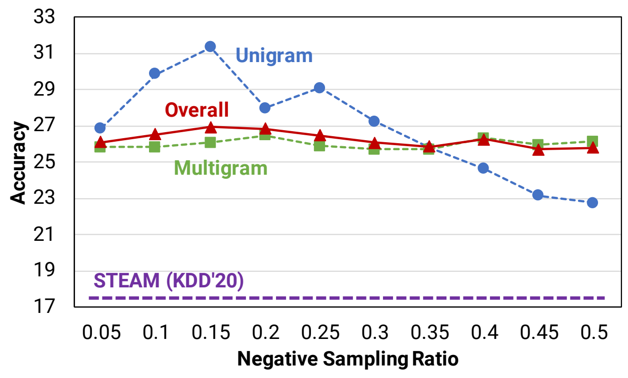


Figure 4: Effect of negative sampling ratio r_{neg} on the quality of concept name generation.

- **DroneTaxo:**[1] is a human-curated hierarchical ontology on unmanned aerial vehicle (UAV).
- **MeSH [14]:** It is a taxonomy of medical and biological terms suggested by the National Library of Medicine (NLM).
- **SemEval-Sci** and **SemEval-Env [5]:** They are from the shared task of taxonomy construction in SemEval2016. We select two scientific-domain taxonomies ("science" and "environment"). Both datasets have been used in [32, 37].

We use two different corpora for the two different domains of data: (1) **DBLP corpus** has about 156K paper abstracts from the computer science bibliography website; (2) **PubMed corpus** has around 30M abstracts on MEDLINE. We observe that on the two largest taxonomies, around a half of concept names and a much higher percentage of unique tokens can found *at least once* in the corpus, which indicates a chance of generating rare concept names token by token. The smaller taxonomies show similar patterns.

4.2 Evaluation Methods

We randomly divide the set of concepts of each taxonomy into training, validation, and test sets at a ratio of 3:1:1. We build "existing" taxonomies with the training sets following the same method in [41]. To answer RQ1, we use Precision, Recall, and F1 score to evaluate the completeness of taxonomy. The Precision is calculated by dividing the number of correctly inserted concepts by the number of total inserted concepts, and Recall is calculated by dividing the the number of correct inserted concepts by the number of total concepts. For RQ2, we use accuracy (Acc) to evaluate the quality of generated concepts. For IE models, we evaluate what percent of concepts in taxonomy can be correctly extracted/generated when a

[1]http://www.dronetology.net/

Largest two:	MAG-CS P	MAG-CS R	MAG-CS **F1**	MeSH P	MeSH R	MeSH **F1**
HiExpan	19.61	8.23	11.59	17.77	5.66	8.59
TaxoExpan	36.19	20.20	25.92	**26.87**	11.79	16.39
Graph2Taxo	23.43	12.97	16.70	26.13	10.35	14.83
STEAM	**36.73**	23.42	28.60	26.05	11.23	15.69
ARBORIST	29.72	15.90	20.72	26.19	10.76	15.25
TMN	28.82	23.09	25.64	23.73	9.84	13.91
GenTaxo	36.15	28.19	31.67	21.47	17.10	19.03
GenTaxo++	36.24	**28.68**	**32.01**	22.61	**17.66**	**19.83**

Computer Science:	OSConcepts P	OSConcepts R	OSConcepts **F1**	DroneTaxo P	DroneTaxo R	DroneTaxo **F1**
HiExpan	21.77	13.71	16.82	43.24	30.77	35.95
TaxoExpan	30.43	21.32	25.07	60.98	48.08	53.77
Graph2Taxo	22.88	13.71	17.15	44.90	23.31	30.69
STEAM	30.71	19.79	24.07	58.33	53.85	56.00
ARBORIST	**31.09**	18.78	23.42	52.38	42.31	46.81
TMN	30.65	19.29	23.68	47.72	40.38	43.74
GenTaxo	18.32	12.18	14.63	11.63	9.62	10.53
GenTaxo++	30.18	**25.89**	**27.87**	**65.96**	**59.62**	**62.63**

Biology/ Biomedicine:	SemEval-Sci P	SemEval-Sci R	SemEval-Sci **F1**	SemEval-Env P	SemEval-Env R	SemEval-Env **F1**
HiExpan	14.63	10.34	12.12	15.79	8.11	10.72
TaxoExpan	24.14	29.17	26.42	23.07	16.22	19.05
Graph2Taxo	26.19	18.96	21.99	21.05	10.81	14.28
STEAM	35.56	27.58	31.07	46.43	35.13	39.99
ARBORIST	**41.93**	22.41	29.21	46.15	32.43	38.09
TMN	34.15	24.14	28.29	37.93	29.73	33.33
GenTaxo	11.43	6.90	8.61	16.13	13.51	14.70
GenTaxo++	38.78	**32.76**	**35.52**	**48.28**	**37.84**	**42.42**

Table 3: Performance on taxonomy completion: Bold for the highest among all. Underlined for the best baseline. (RQ1)

position is given. We use Uni-grams (Acc-Uni), and Accuracy on Multi-grams (Acc-Multi) for scenarios where dataset contains only Uni-grams and multi-gram concepts.

4.3 Baselines

This work proposes the first method that generates concepts for taxonomy completion. Therefore, We compare GenTaxo and GenTaxo++ with state-of-the-art extraction-based methods below:

- **HiExpan [27]** uses textual patterns and distributional similarities to capture the "isA" relations and then organize the extracted concept pairs into a DAG as the output taxonomy.
- **TaxoExpan [26]** adopts GNNs to encode the positional information and uses a linear layer to identify whether the candidate concept is the parent of the query concept.
- **Graph2Taxo [24]** uses cross-domain graph structure and constraint-based DAG learning for taxonomy construction.
- **STEAM [37]** models the mini-path information to capture global structure information to expand the taxonomy.
- **ARBORIST [17]** is the state-of-the-art method for taxonomy expansion. It aims for taxonomies with heterogeneous edge semantics and adopts a large-margin ranking loss to guaranteed an upper-bound on the shortest-path distance between the predicted parents and actual parents.

- **TMN [41]** is the state-of-the-art method for taxonomy completion. It uses a triplet matching network to match a query concept with (hypernym, hyponym)-pairs.

For RQ1, the extraction-based baselines as well as GenTaxo++ are offered the concepts from the test set if they can be extracted from the text corpus but *NOT* to the pure generation-based GenTaxo. For RQ2, given a *valid* position in an existing taxonomy, it is considered as accurate if a baseline can extract the desired concept from text corpus and assign it to that position or if GenTaxo can generate the concept's name correctly. Hyperparameter settings are in Appendix.

5 EXPERIMENTAL RESULTS

5.1 RQ1: Taxonomy Completion

Table 3 presents the results on taxonomy completion. We have three main observations. First, TaxoExpan and STEAM are either the best or the second best among all the baselines on the six datasets. When TaxoExpan is better, the gap between the two methods in terms of F1 score is no bigger than 0.7% (15.69 vs 16.39 on MeSH); when STEAM is better, its F1 score is at least 2.2% higher than TaxoExpan (e.g., 28.60 vs 25.92 on MAG-CS). So, STEAM is generally a stronger baseline. This is because the sequential model that encodes the mini-paths from the root node to the leaf node learns useful information. The sequence encoders in our GenTaxo learn such information from the template-based sentences. STEAM loses to TaxoExpan on MeSH and OSConcepts due to the over-smoothing issue of too long mini-paths. We also find that ARBORIST and TMN do not perform better than STEAM. This indicates that GNN-based encoder in STEAM captures more structural information (e.g., sibling relations) than ARBORIST's shortest-path distance and TMN's scoring function based on hypernym-hyponym pairs.

Second, on the largest two taxonomies MAG-CS and MeSH, GenTaxo outperforms the best extraction-based methods STEAM and TaxoExpan in terms of recall and F1 score. This is because with the module of concept generation, GenTaxo can produce more desired concepts beyond those that are extracted from the corpus. Moreover, GenTaxo uses the fused representations of relational contexts. Compared with STEAM, GenTaxo encodes both top-down and bottom-up subgraphs. When TaxoExpan considers one-hop egonets of a query concept, GenTaxo aggregates multi-hop information into the fused hidden states.

Third, on the smaller taxonomies GenTaxo performs extremely bad due to the insufficient amount of training data (i.e., fewer than 600 concepts). Note that we assumed the set of new concepts was given for all the extraction-based methods, while GenTaxo was not given it and had to rely on name generation to "create" the set. In a fair setting – when we allow GenTaxo to use the set of new concepts, then we have – GenTaxo++ performs consistently better than all the baselines. This is because it takes advantages of both the concept generation and extraction-and-fill methodologies.

5.2 RQ2: Concept Name Generation

Table 2 presents the results on concept name generation/extraction: Given a *valid* position on an existing taxonomy, evaluate the accuracy of the concept names that are (1) extracted and filled by baselines or (2) generated by GenTaxo. Our observations are:

MeSH: (F1 score)	PT	1 hop	+ 2 hops		+ 3 hops
			Sibling	Grand-p/c	
GRU	×	18.19	18.29	18.92	17.41
	✓	18.35	**19.03**	18.40	17.49
Transformer [?]	×	17.89	18.13	17.97	17.04
	✓	18.02	18.53	18.19	17.07
SciBERT [4]	×	18.05	18.16	18.12	17.29
	✓	18.11	18.87	18.23	17.41

Table 4: Ablation study on sequence encoders for sentence-based relational contexts. (PT is for Pre-training. Grand-p/c is for grandparent/grandchild.)

MeSH: (F1 score)	U/D	1 hop	+ 2 hops	+ 3 hops
GAT [31]	U	18.17	18.43	17.11
	D	18.29	18.94	17.39
GCN [12]	U	18.10	18.27	17.09
	D	18.21	18.35	17.15
GraphSAGE [9]	U	18.12	18.36	17.05
	D	18.19	18.66	17.20
GraphTransformer [13]	U	18.22	18.79	17.13
	D	18.35	**19.03**	17.49

Table 5: Ablation study on graph encoders for subgraph contexts on the taxonomy. (U/D is for undirected/directed subgraph settings in GNN-based aggregation.)

First, among all baselines, STEAM achieves the highest on multi-gram concepts, and TaxoExpan achieves the highest accuracy on uni-gram concepts. This is because the mini-path from root to leaf may encode the naming system for a multi-gram leaf node.

Second, the accuracy scores of TaxoExpan, STEAM, ARBORIST, and TMN are not significantly different from each other (within 1.03%). This indicates that these extraction-based methods are unfortunately limited by the presence of concepts in corpus.

Third, compared GenTaxo (the last two lines in the table) with STEAM, we find GenTaxo achieves 9.7% higher accuracy. This is because GenTaxo can assemble frequent tokens into infrequent or even unseen concepts.

Then, *do negative samples help learn to generate concept names?* In Figure 4, we show the accuracy of GenTaxo given different values of negative sampling ratio r_{neg}.

First, We observe that GenTaxo performs consistently better than the strongest baseline STEAM in this case. And the overall accuracy achieves the highest when $r_{\text{neg}} = 0.15$. From our point of view, the negative samples accelerate the convergence at early stage by providing a better gradient descending direction for loss function. However, too many negative samples would weaken the signal from positive examples, making it difficult for the model to learn knowledge from them.

Second, we find that the uni-gram and multi-gram concepts have different kinds of sensitivity to the ratio but comply to the same trend. Generally uni-grams have higher accuracy because generating fewer tokens is naturally an easier task; however, they take a smaller percentage of the data. So the overall performance is closer to that on multi-grams. And our GenTaxo focuses on generating new multi-gram concepts.

Sentence-based	Graph-based	Fusion method Fuse(·)	P	R	F1
✓	×	-	23.62	14.83	18.22
×	✓	-	7.14	9.06	7.98
✓	✓	Mean Pooling	20.29	17.10	18.56
		Max Pooling	18.36	17.25	17.79
		Attention	20.75	17.20	18.81
		Concatenation	21.47	17.10	**19.03**

Table 6: Ablation study on the methods of fusing sentence-based and graph-based relational representations.

5.3 RQ3: Ablation Studies

In this section, we perform ablation studies to investigate three important designs of GenTaxo: (3.1) Sequence encoders for sentence-based relational contexts; (3.2) Graph encoders for subgraph-based contexts; (3.3) Representation fusion methods. We use MeSH in these studies, so based on Table 3, the F1 score of GenTaxo is *19.03*.

5.3.1 Sequence encoders for sentence-based relational contexts. We consider three types of encoders: GRU, Transformer, and SciBERT that was pre-trained by a general masked language model (MLM) task on massive scientific corpora. We use our proposed task (Masked Concept's Tokens) to pre-train GRU/Transformer and fine-tune SciBERT. We add sentences that describe 1-hop relations (i.e., parent or child), sibling relations, 2-hops relations, and 3-hops relations step by step. Table 4 presents the results of all the combinations. Our observations are as follows.

First, the pre-training process on related corpus is useful for generating a concept's name at a candidate position in an existing taxonomy. The pre-training task is predicting a token in an existing concept, strongly relevant with the target task. We find that pre-training improves the performance of the three sequence encoders.

Second, surprisingly we find that GRU performs *slightly better* than Transformer and SciBERT (19.03 vs 18.53 and 18.87). The reason may be that the sentence templates that represent relational contexts of a masked position always place two concepts in a relation at the beginning or end of the sentence. Because GRU encodes the sentences from both left-to-right and right-to-left directions, it probably represents the contexts better than the attention mechanisms in Transformer and SciBERT. SciBERT is pre-trained on MLM and performs better than Transformer.

Third, it is not having all relations in 3-hops neighborhood of the masked concept that generates the highest F1 score. On all the three types of sequence encoders, we find that the best collection of constructed sentences are those that describe 1-hop relations (i.e., parent and child) and sibling relations which is a typical relationship of two hops. Because 1-hop ancestor (parent), 2-hop ancestor (grandparent), and 3-hop ancestor are all the "class" of the masked concept, if sentences were created for all these relations, sequence encoders could not distinguish the levels of specificity of the concepts. Similarly, it is good to represent only the closest type of descendant relationships (i.e., child) as "subclass" of the masked concept. And sibling relations are very useful for generating concept names. For example, in Figure 2, "nucleic acid denaturation" and "nucleic acid renaturation" have similar naming patterns when they are sibling concepts.

5.3.2 Graph encoders for subgraph-based relational contexts. We consider four types of GNN-based encoders: GAT [31], GCN [12], GraphSAGE [9], and GraphTransformer [13]. We add 1-hop, 2-hop, and 3-hop relations step by step in constructing the top-down and bottom-up subgraphs. The construction forms either undirected or directed subgraphs in the information aggregation GNN algorithms. Table 5 presents the results. Our observations are as follows.

First, we find that encoding directed subgraphs can achieved a better performance than encoding undirected subgraphs for all the four types of graph encoders. This is because the directed subgraph can represent asymmetric relations. For example, it can distinguish parent-child and child-parent relations. In directed subgraphs, the edges always point from parent to child while such information is missing in undirected graphs.

Second, the best graph encoder is GraphTransformer and the second best is Graph Attention Network (GAT). They both have the attention mechanism which plays a significant role in aggregating information from top-down and bottom-up subgraphs for generating the name of concept. GraphTransformer adopts the Transformer architecture (of *all* attention mechanism) that can capture the contextual information better and show stronger ability of generalization than GAT.

Third, we find that all the types of graph encoders perform the best with 2-hops subgraphs. The reason may be that the GNN-based architectures cannot effectively aggregate multi-hop information. In other words, they suffer from the issue of over smoothing when they use to encode information from 3-hops neighbors.

5.3.3 Fusion methods on sentence- and graph-based representations. We investigate whether the fusion of these two types of relational representations can improve the performance. We also look at the effect of different fusion methods in terms of the choice of Fuse(·). Results are presented in Table 6. We have three observations.

First, sentence-based relational representations perform much more effectively than those based on subgraphs because the sequence encoders are naturally good at generating concept names as a task of completing sentences.

Second, the representation fusion significantly improves recall (from 14.83% to 17.25%) showing that the two types of encoders are complementary in generating missing concepts.

Third, concatenation is the best Fuse(·) because it maintains the information from each encoder at the most so that the precision does not drop much when recall improves.

6 RELATED WORK

Our work is closely related to the following three research topics.
Taxonomy Construction. Many methods used a two-step scheme: (1) extracted hypernym-hyponym pairs from corpus, then (2) organized the extracted relations into hierarchical structures. Pattern-based [21, 33, 38] and embedding-based methods [16, 23] were widely used in the first step. The second step was often considered as graph optimization and solved by maximum spanning tree [3], optimal branching [30], and minimum-cost flow [8]. Mao *et al.* used reinforcement learning to organize the hypernym-hyponym pairs by optimizing a holistic tree metric as a reward function over the training taxonomies [18]. Shang *et al.* transferred knowledge from an existing taxonomy to generate one for a new domain [24].

Taxonomy Expansion. These methods aimed at collecting emergent terms and placing them at appropriate positions in an existing taxonomy. Aly *et al.* adopted hyperbolic embedding to expand and refine an existing taxonomy [2]. Shen *et al.* [27] and Vedula *et al.* [29] applied semantic patterns to determine the position of the new terms. Fauceglia *et al.* used a hybrid method to combine lexico-syntactic patterns, semantic web, and neural networks [7]. Manzoor *et al.* proposed a joint-learning framework to simultaneously learn latent representations for concepts and semantic relations [17]. Shen *et al.* proposed a position-enhanced graph neural network to encode the relative position of each term [26]. Yu *et al.* applied a mini-path-based classifier instead of hypernym attachment [37].
Keypharse Generation. This is the most relevant task with the proposed concept name generation in taxonomies. Meng *et al.* [19, 20] applied Seq2Seq to generate keyphrases from scientific articles. Ahmad *et al.* proposed a joint learning method to select, extract, and generate keyphrases [1]. Our approaches combine textual and taxonomic information to generate the concept names accurately.

7 CONCLUSIONS

In this work, we proposed GenTaxo to enhance taxonomy completion by identifying the positions in existing taxonomies that need new concepts and generating the concept names. It learned position embeddings from both graph-based and language-based relational contexts. Experimental results demonstrated that GenTaxo improves the completeness of real-world taxonomies than extraction-based methods.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This Research was supported by NSF CCF-1901059.

REFERENCES

[1] Wasi Uddin Ahmad, Xiao Bai, Soomin Lee, and Kai-Wei Chang. 2020. Select, Extract and Generate: Neural Keyphrase Generation with Syntactic Guidance. *arXiv preprint arXiv:2008.01739.*

[2] Rami Aly, Shantanu Acharya, Alexander Ossa, Arne Köhn, Chris Biemann, and Alexander Panchenko. 2019. Every Child Should Have Parents: A Taxonomy Refinement Algorithm on Hyperbolic Term Embeddings. In *ACL.*

[3] Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *ACL.*

[4] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676.*

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

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL.*

[7] Nicolas Rodolfo Fauceglia, Alfio Gliozzo, Sarthak Dash, Md Faisal Mahbub Chowdhury, and Nandana Mihindukulasooriya. 2019. Automatic Taxonomy Induction and Expansion. In *EMNLP.*

[8] Amit Gupta, Rémi Lebret, Hamza Harkous, and Karl Aberer. 2017. Taxonomy induction using hypernym subsequences. In *CIKM.*

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS.*

[10] Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. 2016. Understand short texts by harvesting and analyzing semantic knowledge. *TKDE.*

[11] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *WSDM.*

[12] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR.*

[13] Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *NAACL.*

[14] Carolyn E Lipscomb. 2000. Medical subject headings (MeSH). *Bulletin of the Medical Library Association.*

[15] Bang Liu, Weidong Guo, Di Niu, Chaoyue Wang, Shunnan Xu, Jinghong Lin, Kunfeng Lai, and Yu Xu. 2019. A user-centered concept mining system for query and document understanding at tencent. In *KDD.*

[16] 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.*

[17] Emaad Manzoor, Rui Li, Dhananjay Shrouty, and Jure Leskovec. 2020. Expanding Taxonomies with Implicit Edge Semantics. In *WWW.*

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

[19] Rui Meng, Xingdi Yuan, Tong Wang, Sanqiang Zhao, Adam Trischler, and Daqing He. 2020. An Empirical Study on Neural Keyphrase Generation. *arXiv preprint arXiv:2009.10229.*

[20] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep Keyphrase Generation. In *ACL.*

[21] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: a taxonomy of relational patterns with semantic types. In *EMNLP.*

[22] James L Peterson and Abraham Silberschatz. 1985. *Operating system concepts.*

[23] Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING.*

[24] Chao Shang, Sarthak Dash, Md Faisal Mahbub Chowdhury, Nandana Mihindukulasooriya, and Alfio Gliozzo. 2020. Taxonomy Construction of Unseen Domains via Graph-based Cross-Domain Knowledge Transfer. In *ACL.*

[25] Jingbo Shang, Xinyang Zhang, Liyuan Liu, Sha Li, and Jiawei Han. 2020. Nettaxo: Automated taxonomy construction from text network. In *WWW.*

[26] Jiaming Shen, Zhihong Shen, Chenyan Xiong, Chi Wang, Kuansan Wang, and Jiawei Han. 2020. TaxoExpan: Self-supervised Taxonomy Expansion with Position-Enhanced Graph Neural Network. In *WWW.*

[27] Jiaming Shen, Zeqiu Wu, Dongming Lei, Chao Zhang, Xiang Ren, Michelle T Vanni, Brian M Sadler, and Jiawei Han. 2018. Hiexpan: Task-guided taxonomy construction by hierarchical tree expansion. In *KDD.*

[28] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. 2015. An overview of mas and applications. In *WWW.*

[29] Nikhita Vedula, Patrick K Nicholson, Deepak Ajwani, Sourav Dutta, Alessandra Sala, and Srinivasan Parthasarathy. 2018. Enriching taxonomies with functional domain knowledge. In *SIGIR.*

[30] Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. Ontolearn reloaded: A graph algorithm for taxonomy induction. In *Computational Linguistics.*

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR.*

[32] Suyuchen Wang, Ruihui Zhao, Xi Chen, Yefeng Zheng, and Bang Liu. 2021. Enquire One's Parent and Child Before Decision: Fully Exploit Hierarchical Structure for Self-Supervised Taxonomy Expansion. In *WWW.*

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

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

[35] Wenhao Yu, Lingfei Wu, Yu Deng, Qingkai Zeng, Ruchi Mahindru, Sinem Guven, and Meng Jiang. 2021. Technical Question Answering across Tasks and Domains. In *NAACL.*

[36] Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2020. A Survey of Knowledge-Enhanced Text Generation. *arXiv preprint arXiv:2010.04389.*

[37] Yue Yu, Yinghao Li, Jiaming Shen, Hao Feng, Jimeng Sun, and Chao Zhang. 2020. STEAM: Self-Supervised Taxonomy Expansion with Mini-Paths. In *KDD.*

[38] Qingkai Zeng, Mengxia Yu, Wenhao Yu, Jinjun Xiong, Yiyu Shi, and Meng Jiang. 2019. Faceted hierarchy: A new graph type to organize scientific concepts and a construction method. In *TextGraphs workshop.*

[39] Qingkai Zeng, Wenhao Yu, Mengxia Yu, Tianwen Jiang, Tim Weninger, and Meng Jiang. 2020. Tri-Train: Automatic Pre-Fine Tuning between Pre-Training and Fine-Tuning for SciNER. In *EMNLP.*

[40] Chao Zhang, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian Sadler, Michelle Vanni, and Jiawei Han. 2018. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. In *KDD.*

[41] Jieyu Zhang, Xiangchen Song, Ying Zeng, Jiaming Shen, Yuning Mao, Lei Li, et al. 2021. Taxonomy Completion via Triplet Matching Network. In *AAAI.*

A APPENDIX

A.1 Sentence encoders

The descriptions of three sentence encoders we use are as below.

(1) BiGRU: Given a sentence $s \in \mathcal{S}(v)$, we denote n as the length of s which is also the position index of [MASK]. A three-layered Bidirectional Gated Recurrent Units (GRU) network creates the hidden state of the [MASK] token as $\mathbf{h}(s) = \overrightarrow{\mathbf{h}_n(s)} \oplus \overleftarrow{\mathbf{h}_n(s)}$, where $\oplus$ denotes vector concatenation.

(2) Transformer: It is an encoder-decoder architecture with only attention mechanisms without any Recurrent Neural Networks (e.g., GRU). The attention mechanism looks at an input sequence and decides at each step which other parts of the sequence are important. The hidden state is that of the [MASK] token: $\mathbf{h}(s) = \mathbf{h}_n(s)$.

(3) SciBERT: It is a bidirectional Transformer-based encoder [4] that leverages unsupervised pretraining on a large corpus of scientific publications (from semanticscholar.org) to improve performance on scientific NLP tasks such as sequence tagging, sentence classification, and dependency parsing.

A.2 Baselines Implementation

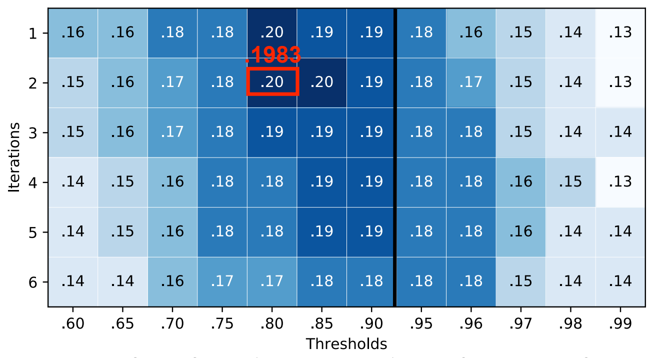
HiExpan, TaxoExpan, STEAM, and ARBORIST are used for taxonomy expansion. Given a subgraph which includes the ancestor, parent, and sibling nodes as the seed taxonomy, the final output is a taxonomy expanded by the target concepts. For taxonomy completion methods, we set the threshold for deciding whether to add the target node to the specified position of taxonomy as 0.5. For Graph2Taxo, we input the graph-based context within two hops from the target concept in order to construct the taxonomy structure. To evaluate Graph2Taxo, we check the parent and child nodes of the target concept. When both sets matched the ground truth, the prediction result is marked as correct.

A.3 Hyperparameter Settings

We use stochastic gradient descent (SGD) with momentum as our optimizer. We applied a scheduler with "warm restarts" that reduces the learning rate from 0.25 to 0.05 over the course of 5 epochs as a cyclical regiment. Models are trained for 50 epochs based on the validation loss. All the dropout layers used in our model had a default ratio of 0.3. The number of dimensions of hidden states is 200 for sequence encoders and 100 for graph encoders, respectively. We search for the best loss weight λ in {0.1, 0.2, 0.5, 1, 2, 5} and set it as 2. We set $r_{\text{neg}} = 0.15$, $\tau = 0.8$, and $\max_{\text{iter}} = 2$ as default. Their effects will be investigated in the experiments.

A.4 Hyperparameter Analysis in GenTaxo++

Figure 5 presents the F1 score of GenTaxo++ under different values of quality threshold τ and maximum number of iterations max_{iter}. First, quality control in the iterative process is important. While the performance is not good when τ is too small or too big, choosing a proper value is not very sensitive. The range between 0.8 and 0.9 is good, and we choose the best τ at 0.8. Second, the best max_{iter} should not be a big number. The main reason is error propagation – as the number of iteration increases, the false positives from previous iterations will affect the next-step model training.



Iterations	.60	.65	.70	.75	.80	.85	.90	.95	.96	.97	.98	.99
1	.16	.16	.18	.18	.20	.19	.19	.18	.16	.15	.14	.13
2	.15	.16	.17	.18	.20	.20	.19	.18	.17	.15	.14	.13
3	.15	.16	.17	.18	.19	.19	.19	.18	.18	.15	.14	.14
4	.14	.15	.16	.18	.18	.19	.19	.18	.18	.16	.15	.13
5	.14	.15	.16	.18	.18	.19	.19	.18	.18	.16	.14	.14
6	.14	.14	.16	.17	.17	.18	.18	.18	.18	.15	.14	.14

Figure 5: Effect of τ and max_{iter} on the performance of GenTaxo++ in terms of F1 score on MeSH. The GenTaxo++ result reported in Table 3 is highlighted as red.