Taxonomy Completion via Triplet Matching Network

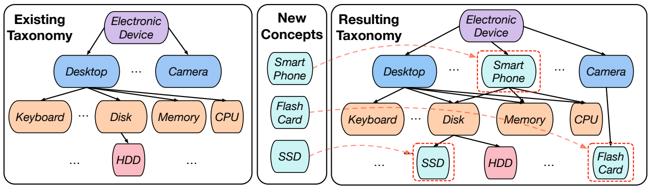
Jieyu Zhang[⋆,1] **Xiangchen Song**,[2] **Ying Zeng**,[3] **Jiaze Chen**,[3] **Jiaming Shen**,[2] **Yuning Mao**,[2] **Lei Li** [3]
[1] Paul G. Allen School of Computer Science & Engineering, University of Washington, WA, USA
[2] Department of Computer Science, University of Illinois Urbana-Champaign, IL, USA
[3] ByteDance AI Lab
jieyuz2@cs.washington.edu, {xs22, js2, yuningm2}@illinois.edu, {zengying.ss, chenjiaze, lileilab}@bytedance.com

Abstract

Automatically constructing taxonomy finds many applications in e-commerce and web search. One critical challenge is as data and business scope grow in real applications, new concepts are emerging and needed to be added to the existing taxonomy. Previous approaches focus on the taxonomy expansion, i.e. finding an appropriate hypernym concept from the taxonomy for a new query concept. In this paper, we formulate a new task, "taxonomy completion", by discovering both the hypernym and hyponym concepts for a query. We propose **T**riplet **M**atching **N**etwork (TMN), to find the appropriate ⟨hypernym, hyponym⟩ pairs for a given query concept. TMN consists of one primal scorer and multiple auxiliary scorers. These auxiliary scorers capture various fine-grained signals (e.g., query to hypernym or query to hyponym semantics), and the primal scorer makes a holistic prediction on ⟨query, hypernym, hyponym⟩ triplet based on the internal feature representations of all auxiliary scorers. Also, an innovative channel-wise gating mechanism that retains task-specific information in concept representations is introduced to further boost model performance. Experiments on four real-world large-scale datasets show that TMN achieves the best performance on both taxonomy completion task and the previous taxonomy expansion task, outperforming existing methods.

Introduction

Taxonomies, formulated as directed acyclic graphs or trees, have been widely used to organize knowledge in various domains, such as news domain (Vrandecic 2012; Mao et al. 2019), scientific domain (Lipscomb 2000; Sinha et al. 2015; Shen et al. 2018c) and online commerce (Karamanolakis, Ma, and Dong 2020; Mao et al. 2020). Equipped with these curated taxonomies, researchers are able to boost the performance of numerous downstream applications such as query understanding (Hua et al. 2017; Yang, Zhang, and Han 2020), content browsing (Yang 2012), personalized recommendation (Zhang et al. 2014; Huang et al. 2019), and web search (Wu et al. 2012; Liu et al. 2019).



Query	Candidate Position	Label
Smart Phone	*< Electronic Device, CPU >*	Positive
Smart Phone	*< Electronic Device, HDD >*	Partially-correct Negative
Smart Phone	*< Desktop, CPU>*	Partially-correct Negative
Smart Phone	*< Desktop, Keyboard>*	Negative

Figure 1: An example of completing one "*Electronic Device*" taxonomy. The table illustrates different types of candidate positions for a given query "*Smart Phone*".

As human knowledge is constantly growing and new concepts emerge everyday, it is needed to dynamically complete an existing taxonomy. Figure 1 shows an illustrative example where a taxonomy of "*Electronic Device*" is completed to include new devices (*e.g.*, "*Smart Phone*") and hardware (*e.g.*, "*SSD*"). Most existing taxonomies are curated by domain experts. However, such manual curations are labor-intensive, time-consuming and rarely-complete, and therefore infeasible to handle the influx of new contents in online streaming setting. To this end, many recent studies (Shen et al. 2020; Manzoor et al. 2020; Yu et al. 2020) investigate the problem of *taxonomy expansion* which aims to automatically expand an existing taxonomy. Specifically, given a query concept, these methods first rank each concept in the existing taxonomy based on how likely it is the hypernym of the query concept measured by an *one-to-one* matching score between the two concepts. Then, the query concept is added into the existing taxonomy as the hyponym of the top-ranked concepts. Notice that such a formulation is built upon one strong assumption: all new concepts can only be added into existing taxonomy as hyponyms (i.e., leaf nodes[1]). However, we argue that such a "hyponym-only" assumption is inappropriate in real applications. For example, in Fig 1, the term "*Smart Phone*" is invented much later than term "*CPU*", which means that when "*Smart Phone*" emerges, "*CPU*" already exists in taxonomy. In this case, it is inappropriate to add "*Smart Phone*" into taxonomy as leaf node because "*CPU*" is a hyponym of "*Smart Phone*".

In this paper, instead, we define and investigate a new *taxonomy completion* task *without* the strong "hyponym-only"

[*] This work is done while interning at ByteDance AI Lab.

[1] Nodes with zero out-degree in a directed acyclic graph.

assumption. Formally, given an existing taxonomy and a set of new concepts, we aim to automatically complete the taxonomy to incorporate these new concepts by discovering the most likely $\langle$hypernym, hyponym$\rangle$ pairs of each new concept. For instance, in Fig 1, one of the most likely candidate pairs for "*Smart Phone*" is $\langle$"*Electronic Device*", "*CPU*"$\rangle$. This formulation leads to a novel *one-to-pair* matching problem different from the previous *one-to-one* setting in taxonomy expansion task that only seeks for a new concept's most likely hypernyms while ignores its possible hyponyms. Note that the hypernym/hyponym concept within the candidate $\langle$hypernym, hyponym$\rangle$ pair could be a "pseudo concept" in case there is no appropriate one for a given query concept. We can easily see that the taxonomy expansion task is a special case of taxonomy completion when the hyponym concepts are always "pseudo concept".

Tackling the new taxonomy completion task is challenging because the induced one-to-pair matching problem results in the existence of a special type of negative candidate $\langle$hypernym, hyponym$\rangle$ pairs we called *partially-correct negative candidates*. Before introducing partially-correct negative candidates, we first clarify that for a given query concept n_q, a candidate pair of existing concepts $\langle n_p, n_c\rangle$ is *positive* if $n_p(n_c)$ is the true hypernym (hyponym) of n_q and *negtiave* otherwise. Then, a candidate pair $\langle n_p, n_c\rangle$ is partially-correct negative if either n_p is true hypernym but n_c is not true hyponym or vice versa. We illustrate the different types of candidate pairs in the table of Fig 1. Due to the high correlation of positive and partially-correct negative candidates, the model might struggle to distinguish one from another.

To solve the aforementioned challenge, we propose a novel **T**riplet **M**atching **N**etwork (TMN), which learns a scoring function to output the matching score of a $\langle$query, hypernym, hyponym$\rangle$ triplet and leverages *auxiliary signals* to help distinguish positive pairs from partially-correct negative ones. Specifically, auxiliary signals are binary signals indicating whether one component within the pair is positive or not, in contrast to binary *primal signals* that reveal holistically whether a candidate position is positive or not. To make best use of the auxiliary signals to handle the existence of partially-correct negative, TMN consists of multiple *auxiliary scorers* that learn different auxiliary signals via corresponding auxiliary loss and one *primal scorer* that aggregates internal feature representations of auxiliary scorers to output the final matching score. The auxiliary and primal scorers are jointly trained in an auxiliary learning framework. In this way, we encourage the model to learn meaningful internal feature representations for the primal scorer to discriminate between positive, negative, and partially-correct negative candidates. In addition, we propose an innovative technique called *channel-wise gating mechanism* to regulate the representations of concepts. It produces a channel-wise gating vector based on the $\langle$query, hypernym, hyponym$\rangle$ triplet, and then modifies the embeddings using this channel-wise gating vector to reduce the effect of irrelevant information stored in embeddings while retain the most task-specific information when calculating matching scores.

In the experiments, we benchmark the taxonomy completion task on four real-world taxonomies from different domains using modified version of multiple one-to-one matching models and state-of-the-art taxonomy expansion methods. Our experimental results show that TMN outperforms the baselines by a large margin on both taxonomy completion task and taxonomy expansion task. Finally, ablation study demonstrates the effectiveness of each component of TMN, and efficiency analysis shows the efficiency of TMN at inference stage.

Contributions. To summarize, our major contributions include: (1) a more realistic task called taxonomy completion which simultaneously finds hypernym and hyponym concepts of new concepts; (2) a novel and effective Triple Matching Network (TMN) to solve the one-to-pair matching problem induced from the taxonomy completion task by leveraging auxiliary signals and an innovative channel-wise gating mechanism; and (3) extensive experiments that verify both the effectiveness and efficiency of TMN framework on four real-world large-scale taxonomies from different domains.

Problem Formulation

The input of the taxonomy completion task includes two parts: (1) an existing taxonomy $\mathcal{T}^0 = (\mathcal{N}^0, \mathcal{E}^0)$ and (2) a set of new concepts $\mathcal{C}$. We assume each concept $n_i \in \mathcal{N}^0 \cup \mathcal{C}$ has an initial embedding vector $\mathbf{x}_i \in \mathbb{R}^d$ as in previous studies (Jurgens and Pilehvar 2016; Vedula et al. 2018; Aly et al. 2019). The overall goal is to complete the existing taxonomy $\mathcal{T}^0$ into a larger one $\mathcal{T} = (\mathcal{N}^0 \cup \mathcal{C}, \mathcal{E}')$ by adding the new concept $n_q \in \mathcal{C}$ between any pair of existing concepts $\langle n_p, n_c\rangle$ to form two new taxonomic relations $\langle n_p, n_q\rangle$ and $\langle n_q, n_c\rangle$. Following previous studies (Shen et al. 2020; Manzoor et al. 2020), we assume the new concepts in $\mathcal{C}$ are independent to each other and thus reduce the more generic taxonomy completion task into $|C|$ independent simplified problems. Note that we use the term "hypernym" and "parent", "hyponym" and "child" interchangeably throughout the paper.

Taxonomy. Follow (Shen et al. 2020), we define a taxonomy $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ as a directed acyclic graph where each node $n \in \mathcal{N}$ represents a concept (*i.e.*, a word or a phrase) and each directed edge $\langle n_p, n_c\rangle \in \mathcal{E}$ indicates a relation expressing that concept n_p is the most specific concept that is more general than concept n_c. Here, the relation types of edges are implicitly defined by existing taxonomy.

Candidate Position. A valid candidate position is a pair of concepts $\langle n_p, n_c\rangle$ where n_c is one of the descendants of n_p in the existing taxonomy. This definition reduces the search space of candidate positions. Note that n_p or n_c could be a "pseudo concept" acting as a placeholder.

Positive Position. For a query concept n_q, positive position $\langle n_p, n_c\rangle$ is a candidate position wherein n_p and n_c is the true parent and child of n_q, respectively.

Negative Position. For a query concept n_q, negative position $\langle n_p, n_c\rangle$ is a candidate position wherein n_p or n_c is not the true parent or child of n_q, respectively.

Partially-correct Negative Position. For a query concept n_q, partially-correct negative position $\langle n_p, n_c\rangle$ is a negative position but n_p or n_c is the true parent or child of n_q.

The TMN Framework

In this section, we first introduce our one-to-pair matching model which leverages auxiliary signals to augment primal matching task. Then, we present a novel channel-wise gating mechanism designed for regulating concept embedding to boost the model performance. Finally, we discuss how to generate self-supervision data from the existing taxonomy and use them to train the TMN model. The overall model architecture is presented in Fig 2.

Modeling One-To-Pair Matching

In this work, we seek to learn a model $s : \mathcal{N} \times (\mathcal{N} \times \mathcal{N}) \rightarrow \mathbb{R}$ with parameter Θ that can measure the relatedness of a query concept n_q and a candidate position, *i.e.*, a pair of concepts $t = \langle n_p, n_c \rangle$, in existing taxonomy $\mathcal{T}^0$. A straightforward instantiation of s is as follows:

$$s(n_q, n_p, n_c) = f(\mathbf{x}_q, [\mathbf{x}_p, \mathbf{x}_c]) = f(\mathbf{x}_q, \mathbf{x}_t) \tag{1}$$

Where f is a parameterized scoring function of choice that outputs the relatedness score of n_q and $\langle n_p, n_c \rangle$, and $[\cdot]$ represents the concatenation operation. This formulation simply degenerates one-to-pair matching into one-to-one matching by using concatenation of $\mathbf{x}_p$ and $\mathbf{x}_c$ as representation of candidate position. Here, we choose the neural tensor network (Socher et al. 2013) as our base model:

$$s(n_q, n_p, n_c) = \mathbf{u}^T \sigma(h(\mathbf{x}_q, \mathbf{x}_t)) \tag{2}$$

$$h(\mathbf{x}_q, \mathbf{x}_t) = \mathbf{x}_q \mathbf{W}^{[1:k]} \mathbf{x}_t + \mathbf{V} \begin{bmatrix} \mathbf{x}_q \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b} \tag{3}$$

Where $\sigma = \tanh(\cdot)$ is a standard nonlinearity applied element-wise, $\mathbf{W}^{[1:k]} \in \mathbb{R}^{d \times 2d \times k}$ is a tensor and the bilinear tensor product $\mathbf{x}_q \mathbf{W}^{[1:k]} \mathbf{x}_t$ results in a vector $\mathbf{r} \in \mathbb{R}^k$, where each entry is computed by one slice $i = 1, \ldots, k$ of the tensor: $\mathbf{h}_i = \mathbf{x}_q \mathbf{W}^i \mathbf{x}_t$. The other parameters are the standard form of a neural network: $\mathbf{V} \in \mathbb{R}^{k \times 2d}$ and $\mathbf{u} \in \mathbb{R}^k$, $\mathbf{b} \in \mathbb{R}^k$. We call the vector $\mathbf{h}$ output by $h(\cdot, \cdot)$ the internal feature representation of neural tensor network.

However, such a naïve instantiation only measures the coarse-grained relatedness of query n_q and the whole candidate pair $\langle n_p, n_c \rangle$ but fails to capture the fine-grained relatedness of $\langle n_q, n_p \rangle$ and $\langle n_q, n_c \rangle$, preventing the model from learning to clearly distinguish positive candidates from partially-correct negatives candidates.

To address the limitation of the naive approach, we propose a novel expressive **Triplet Matching Network** (TMN). Specifically, we develop multiple auxiliary scorers to capture both coarse- and fine-grained relatedness in one-to-pair matching, and one primal scorer that inputs the internal feature representations of all auxiliary scorers and outputs final matching scores. For each auxiliary scorer, we adopt neural tensor network as in Eq. 2 as instantiation due to its expressiveness, and the corresponding k-dimension internal feature representation $\mathbf{h}$ is as in Eq. 3. Assume we have l auxiliary scorers, each with k-dimension internal feature representation $\mathbf{h}_j$ and $j = 1, \ldots, l$. Then, the primal scorer is a single-layer projection with non-linear activation function:

$$s_{primal}(n_q, n_p, n_c) = \mathbf{u}_p^T \sigma([\mathbf{h}_1, \ldots, \mathbf{h}_l]) \tag{4}$$

Where, again, $\sigma = \tanh(\cdot)$ and $\mathbf{u}_p \in \mathbb{R}^{kl}$. Now we elaborate the three auxiliary scorers that capture both coarse- and fine-grained relatedness:

$$s_1(n_q, n_p) = \mathbf{u}_1^T \sigma(\mathbf{h}_1(\mathbf{x}_q, \mathbf{x}_p)) \tag{5}$$

$$s_2(n_q, n_c) = \mathbf{u}_2^T \sigma(\mathbf{h}_2(\mathbf{x}_q, \mathbf{x}_c)) \tag{6}$$

$$s_3(n_q, n_p, n_c) = \mathbf{u}_3^T \sigma(\mathbf{h}_3(\mathbf{x}_q, [\mathbf{x}_p, \mathbf{x}_c])) \tag{7}$$

Where the auxiliary scorer s_1 and s_2 capture the fine-grained relatedness of $\langle n_q, n_p \rangle$ and $\langle n_q, n_c \rangle$ respectively, while s_3 is for coarse-grained relatedness between n_q and $\langle n_p, n_c \rangle$.

Given above formulations, primal scorer s_{primal} can be trained using primal signals indicating whether $\langle n_p, n_c \rangle$ is positive candidate of n_q or not, and auxiliary scorers can be trained via corresponding auxiliary signals. Particularly, s_1 will be trained to learn whether n_p is positive parent of n_q, s_2 is to learn whether n_c is positive child of n_q, and s_3 captures coarse-grained relatedness between n_q and $\langle n_p, n_c \rangle$ so its auxiliary signal is exactly the same as primal signal. Although s_3 and s_{primal} share the same supervision signals and both aim to capture relatedness between n_q and $\langle n_p, n_c \rangle$, s_{primal} outputs matching score based on the internal feature representations of all auxiliary scorers including s_3, which enables s_{primal} to rely on s_1 or s_2 when s_3 struggle to differentiate positive candidates from partially-correct candidates negatives.

Channel-wise Gating Mechanism

As the nature of taxonomy, concepts under the same ancestor are semantically related to each other, which makes it challenging for model to learn the true taxonomic relations based on concept embeddings, especially in bottom-level of a taxonomy. For instance, in Fig 1, the model needs to learn that "*Disk*" is the true parent of "*SSD*" but "*Memory*" is not. However, "*Disk*" and "*Memory*" are siblings in taxonomy, which makes them highly-related, and therefore hard to distinguish based on their embeddings learned from a more general corpus.

To mitigate this problem, instead of directly using initial embedding vectors of concepts, we propose a novel channel-wise gating mechanism to regulate the information stored in initial embedding vectors, reducing the negative effects of irrelevant or spurious information on learning taxonomic relations. Specially, to distinguish "*Disk*" and "*Memory*" that both belong to "*Desktop*", we would like to filter out the shared information stored in their embeddings related to "*Desktop*" in order to push the model to focus on the remaining more specific information. Formally, we give the formulation of channel-wise gating mechanism as follows:

$$\mathbf{g}_p = \theta(\mathbf{W}_1[\mathbf{x}_q, \mathbf{x}_p, \mathbf{x}_c]) \tag{8}$$

$$\hat{\mathbf{x}}_p = \mathbf{g}_p \odot \mathbf{x}_p \tag{9}$$

$$\mathbf{g}_c = \theta(\mathbf{W}_2[\mathbf{x}_q, \mathbf{x}_p, \mathbf{x}_c]) \tag{10}$$

$$\hat{\mathbf{x}}_c = \mathbf{g}_c \odot \mathbf{x}_c \tag{11}$$

Where $\theta(\cdot)$ is a sigmoid function and $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times 3d}$. $\odot$ is element-wise multiplication. $\mathbf{g}$ is the channel-wise gating vector dependent on embeddings of both query and candidate positions. We treat each dimension of concept embedding

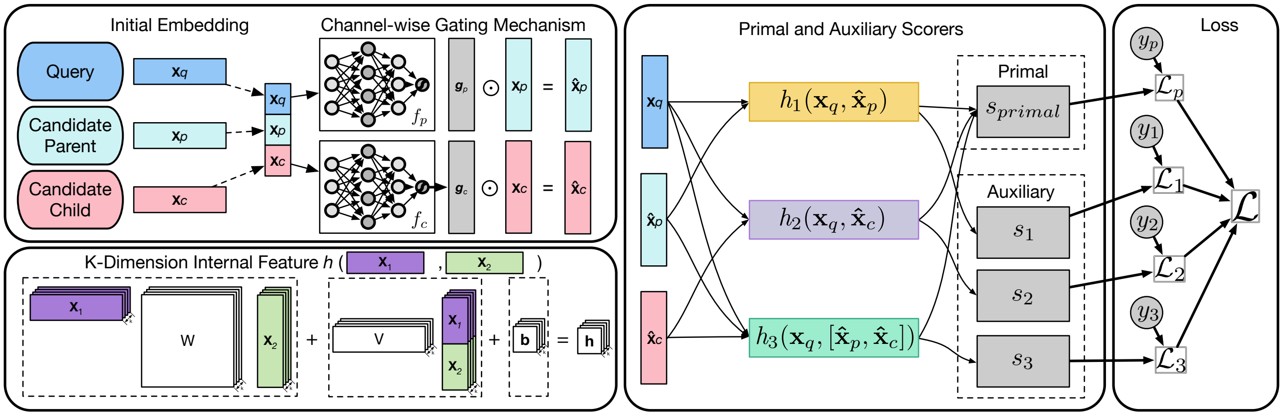

Initial Embedding
Channel-wise Gating Mechanism
Query
Candidate Parent
Candidate Child
$\mathbf{x}_q$
$\mathbf{x}_p$
$\mathbf{x}_c$
f_p
f_c
$\mathbf{g}_p$ ⊙ $\mathbf{x}_p$ = $\hat{\mathbf{x}}_p$
$\mathbf{g}_c$ ⊙ $\mathbf{x}_c$ = $\hat{\mathbf{x}}_c$
K-Dimension Internal Feature h ($\mathbf{x}_1$, $\mathbf{x}_2$)
W
V
b
h
Primal and Auxiliary Scorers
$h_1(\mathbf{x}_q, \hat{\mathbf{x}}_p)$
$h_2(\mathbf{x}_q, \hat{\mathbf{x}}_c)$
$h_3(\mathbf{x}_q, [\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_c])$
Primal
s_{primal}
Auxiliary
s_1
s_2
s_3
Loss
y_p
$\mathcal{L}_p$
y_1
$\mathcal{L}_1$
y_2
$\mathcal{L}_2$
y_3
$\mathcal{L}_3$
$\mathcal{L}$

Figure 2: Overview of **TMN** framework.

as a channel to store information, and the output value of sigmoid lies in the interval $[0, 1]$, which enables the channel-wise gating vector $\mathbf{g}$ to downscale irrelevant information in certain channels while retain task-specific ones.

With the gated embedding $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_c$ in hand, we now replace the initial embedding vectors in Eq. 3 with it to facilitate TMN. Notably, this simple channel-wise gating mechanism is ready to be plugged in any matching models.

Jointly Learning Primal and Auxiliary Scorers

In this section, we first introduce how to generate self-supervision data as well as primal and auxiliary signals from the existing taxonomy, and then propose to jointly learn the primal and auxiliary scorers.

Self-supervision Generation. Given one node n_q in the existing taxonomy $\mathcal{T}^0 = (\mathcal{N}^0, \mathcal{E}^0)$ as "*query*", we first construct a positive $\langle n_p, n_c \rangle$ pair by using one of its parents n_p and one of its children n_c. Then, we sample N negative pairs from valid candidates positions that are not positive positions. Notably, it is allowed that one end of the negative pair is true parent/child of n_q. Given a candidate pair $\langle n_p, n_c \rangle$, the primal signal, *i.e.* y, indicates whether the whole pair is positive or not, while the auxiliary signals, *i.e.* y_p and y_c, represent whether n_p (n_c) is the true parent (child) of query n_q respectively regardless of the primal signal. The generated positive and negative pairs as well as corresponding primal and auxiliary signals collectively consist of one training instance $(\mathbf{X}, \mathbf{y})$. By repeating the above process for each node in $\mathcal{T}^0$, we obtain the full self-supervision dataset $\mathbb{D} = \{(\mathbf{X_1}, \mathbf{y_1}), \ldots, (\mathbf{X}_{|\mathcal{N}^0|}, \mathbf{y}_{|\mathcal{N}^0|})\}$.

Learning Objective. We learn our model on $\mathbb{D}$ using the following objective:

$$\mathcal{L}(\Theta) = \mathcal{L}_p + \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_2 + \lambda_3 \mathcal{L}_3 \tag{12}$$

where $\mathcal{L}_p$ represents the loss for primal scorer and $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$ are auxiliary losses for auxiliary scorers s_1, s_2 and s_3 respectively. The hyperparameters λ_1, λ_2 and λ_3 are weights to adjust relative effects of each auxiliary loss. The above objective function is similar to multi-task learning at the first glance, but it is an auxiliary learning strategy that only cares the performance of primal task, *i.e.* primal scorer in our case, and the auxiliary loss are meant to augment the learning of primal task.

Table 1: Dataset Statistics. $|\mathcal{N}|$ and $|\mathcal{E}|$ are the number of nodes and edges in the existing taxonomy. $|\mathcal{D}|$ indicates the taxonomy depth.

Dataset	$\|\mathcal{N}\|$	$\|\mathcal{E}\|$	$\|\mathcal{D}\|$
MAG-CS	24,754	42,329	6
MAG-Psychology	23,187	30,041	6
WordNet-Verb	13,936	13,408	13
WordNet-Noun	83,073	76,812	20

Here, $\mathcal{L}_i$ is loss function of choice. We choose binary cross entropy loss for simplicity. Take the primal loss as an example, it is formulated as:

$$\mathcal{L}_p = -\frac{1}{|\mathbb{D}|} \sum_{(\mathbf{X}_i, y_i) \in \mathbb{D}} y_i \log(s_p(\mathbf{X}_i)) + (1 - y_i) \log(1 - s_p(\mathbf{X}_i)) \tag{13}$$

Experiments

Experimental Setup

Dataset. We study the performance of TMN on four large-scale real-world taxonomies.

- **Microsoft Academic Graph (MAG).** We evaluate TMN on the public Field-of-Study (FoS) Taxonomy in Microsoft Academic Graph (MAG) (Sinha et al. 2015). It contains over 660 thousand scientific concepts and more than 700 thousand taxonomic relations. Following (Shen et al. 2020), we construct two datasets which we refer to as **MAG-Psychology** and **MAG-CS** based on the subgraph related to the "Psychology" and "Computer Science" domain, respectively. We compute a 250-dimension word word2vec embedding on a related paper abstracts corpus.
- **WordNet.** Based on WordNet 3.0, we collect verbs and nouns along with the relations among them to form two datasets which we refer to as **WordNet-Verb** and **WordNet-Noun**, respectively. The reason for limiting our

Table 2: Overall results on four datasets. We run all methods five times and report the averaged result with standard deviation. Note that smaller MR indicates better model performance. For all other metrics, larger values indicate better performance. We highlight the best two models in terms of the average performance under each metric.

Method	**MAG-CS**							
	MR	MRR	Recall@1	Recall@5	Recall@10	Prec@1	Prec@5	Prec@10
Closest-Position	9466.670	0.093	0.012	0.034	0.051	0.054	0.029	0.022
Single Layer Model	1025.245 ± 22.827	0.153 ± 0.002	0.030 ± 0.001	0.074 ± 0.001	0.105 ± 0.002	0.130 ± 0.004	0.064 ± 0.001	0.046 ± 0.001
Multiple Layer Model	1110.340 ± 64.987	0.156 ± 0.002	**0.032** ± 0.001	**0.080** ± 0.001	0.108 ± 0.002	**0.140** ± 0.004	**0.069** ± 0.000	0.047 ± 0.001
Bilinear Model	3373.772 ± 7.473	0.026 ± 0.000	0.000 ± 0.000	0.003 ± 0.000	0.006 ± 0.000	0.001 ± 0.000	0.002 ± 0.000	0.003 ± 0.000
Neural Tensor Network	**769.830** ± 2.416	**0.171** ± 0.003	0.023 ± 0.001	0.073 ± 0.001	**0.112** ± 0.002	0.099 ± 0.004	0.063 ± 0.001	**0.049** ± 0.001
TaxoExpan	1523.904 ± 52.982	0.099 ± 0.002	0.004 ± 0.001	0.027 ± 0.002	0.049 ± 0.001	0.017 ± 0.003	0.023 ± 0.001	0.021 ± 0.000
ARBORIST	1142.335 ± 19.249	0.133 ± 0.004	0.008 ± 0.001	0.044 ± 0.003	0.075 ± 0.003	0.037 ± 0.004	0.038 ± 0.003	0.033 ± 0.001
TMN	**639.126** ± 35.849	**0.204** ± 0.005	**0.036** ± 0.003	**0.099** ± 0.006	**0.139** ± 0.006	**0.156** ± 0.011	**0.086** ± 0.005	**0.060** ± 0.003

Method	**MAG-Psychology**							
	MR	MRR	Recall@1	Recall@5	Recall@10	Prec@1	Prec@5	Prec@10
Closest-Position	5201.604	0.168	0.030	0.072	0.107	0.062	0.029	0.022
Single Layer Model	435.548 ± 4.057	0.350 ± 0.002	0.090 ± 0.003	0.209 ± 0.002	0.274 ± 0.003	0.183 ± 0.006	0.085 ± 0.001	0.056 ± 0.001
Multiple Layer Model	**297.644** ± 8.097	**0.413** ± 0.002	**0.110** ± 0.001	**0.265** ± 0.004	**0.334** ± 0.002	**0.224** ± 0.003	**0.108** ± 0.001	**0.068** ± 0.000
Bilinear Model	2113.024 ± 9.231	0.032 ± 0.000	0.000 ± 0.000	0.001 ± 0.000	0.003 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.001 ± 0.000
Neural Tensor Network	299.004 ± 6.294	0.380 ± 0.004	0.066 ± 0.004	0.207 ± 0.002	0.291 ± 0.004	0.134 ± 0.008	0.084 ± 0.001	0.059 ± 0.001
TaxoExpan	728.725 ± 2.096	0.253 ± 0.001	0.015 ± 0.001	0.092 ± 0.001	0.163 ± 0.001	0.031 ± 0.001	0.038 ± 0.000	0.033 ± 0.000
ARBORIST	547.723 ± 20.165	0.344 ± 0.012	0.062 ± 0.009	0.185 ± 0.011	0.256 ± 0.013	0.126 ± 0.018	0.076 ± 0.004	0.052 ± 0.003
TMN	**212.298** ± 3.051	**0.471** ± 0.001	**0.141** ± 0.001	**0.305** ± 0.004	**0.377** ± 0.002	**0.287** ± 0.001	**0.124** ± 0.001	**0.077** ± 0.000

Method	**WordNet-Verb**							
	MR	MRR	Recall@1	Recall@5	Recall@10	Prec@1	Prec@5	Prec@10
Closest-Position	34778.772	0.144	0.011	0.045	0.075	0.020	0.016	0.013
Single Layer Model	2798.243 ± 61.384	0.140 ± 0.009	0.029 ± 0.005	0.065 ± 0.006	0.093 ± 0.008	0.044 ± 0.007	0.019 ± 0.002	0.014 ± 0.001
Multiple Layer Model	2039.213 ± 240.577	0.227 ± 0.020	0.050 ± 0.006	0.120 ± 0.009	0.160 ± 0.015	0.075 ± 0.009	0.036 ± 0.003	0.024 ± 0.002
Bilinear Model	1863.915 ± 5.685	0.175 ± 0.001	0.012 ± 0.001	0.054 ± 0.000	0.096 ± 0.001	0.017 ± 0.001	0.016 ± 0.000	0.015 ± 0.000
Neural Tensor Network	**1599.196** ± 18.409	**0.255** ± 0.003	**0.051** ± 0.002	**0.125** ± 0.006	**0.176** ± 0.005	**0.076** ± 0.003	**0.038** ± 0.002	**0.027** ± 0.001
TaxoExpan	1799.939 ± 4.511	0.227 ± 0.002	0.024 ± 0.001	0.095 ± 0.001	0.140 ± 0.002	0.036 ± 0.002	0.029 ± 0.000	0.021 ± 0.000
ARBORIST	1637.025 ± 4.950	0.206 ± 0.011	0.016 ± 0.004	0.073 ± 0.011	0.116 ± 0.011	0.024 ± 0.006	0.022 ± 0.003	0.018 ± 0.002
TMN	**1445.801** ± 27.209	**0.304** ± 0.005	**0.072** ± 0.003	**0.163** ± 0.005	**0.215** ± 0.001	**0.108** ± 0.005	**0.049** ± 0.002	**0.032** ± 0.000

Method	**WordNet-Noun**							
	MR	MRR	Recall@1	Recall@5	Recall@10	Prec@1	Prec@5	Prec@10
Closest-Position	5601.033	0.136	0.017	0.044	0.074	0.025	0.013	0.011
Single Layer Model	3260.415 ± 79.776	0.177 ± 0.010	0.025 ± 0.003	0.072 ± 0.006	0.103 ± 0.006	0.043 ± 0.005	0.025 ± 0.002	0.018 ± 0.001
Multiple Layer Model	**2801.500** ± 143.579	0.175 ± 0.005	0.029 ± 0.001	0.077 ± 0.002	0.106 ± 0.003	0.051 ± 0.002	0.027 ± 0.001	0.018 ± 0.000
Bilinear Model	3498.184 ± 3.586	0.176 ± 0.001	0.012 ± 0.000	0.052 ± 0.001	0.095 ± 0.001	0.020 ± 0.000	0.018 ± 0.000	0.017 ± 0.000
Neural Tensor Network	2808.900 ± 79.415	0.215 ± 0.007	**0.034** ± 0.002	**0.093** ± 0.003	**0.133** ± 0.004	**0.060** ± 0.004	**0.033** ± 0.001	**0.023** ± 0.001
TaxoExpan	3188.935 ± 17.461	0.209 ± 0.000	0.017 ± 0.000	0.074 ± 0.000	0.125 ± 0.001	0.030 ± 0.001	0.026 ± 0.000	0.022 ± 0.000
ARBORIST	2993.341 ± 114.749	**0.217** ± 0.005	0.021 ± 0.001	0.073 ± 0.002	0.125 ± 0.002	0.036 ± 0.001	0.025 ± 0.001	0.022 ± 0.000
TMN	**1647.665** ± 15.370	**0.270** ± 0.006	**0.039** ± 0.002	**0.111** ± 0.006	**0.167** ± 0.005	**0.068** ± 0.004	**0.039** ± 0.002	**0.029** ± 0.001

choice to only verbs and nouns is that only these parts of speech have fully-developed taxonomies in WordNet (Jurgens and Pilehvar 2016). We obtain the 300-dimension fasttext embeddings[2] as initial feature vectors.

For each dataset, we randomly sample 1,000 nodes for validation and another 1,000 for test. Then we build the initial taxonomy using remaining nodes and associated edges. Notice that new edges will be added into initial taxonomy to avoid the taxonomy from breaking into multiple directed acyclic graphs. Table 1 lists the statistics of these four datasets.

Evaluation Metrics. As our model returns a rank list of candidate positions for each query concept, we evaluate its performance using the following ranking-based metrics.

- **Mean Rank (MR)** measures the average rank position of a query concept's true positions among all candidates. For queries with multiple positive edges, we first calculate the rank position of each individual edge and then take the average of all rank positions.
- **Mean Reciprocal Rank (MRR)** calculates the reciprocal rank of a query concept's true positions. We follow (Ying et al. 2018) and scale the original MRR by a factor 10 to amplify the performance gap between different methods.
- **Recall@k** is the number of query concepts' true positions ranked in the top k, divided by the total number of true positions of all query concepts.
- **Precision@k** is the number of query concepts' true positions ranked in the top k, divided by the total number of queries times k.

Compared Methods. To the best of our knowledge, we are the first to study taxonomy completion task and there is no directly comparable previous method. Thus, we adapt the following related methods to our problem setting and compare TMN with them:

1. **Closest-Position**: A rule-based method which ranks can-

[2] We use the wiki-news-300d-1M-subword.vec.zip version on official website.

didate positions based on the cosine similarity:

$$s(n_q, n_p, n_c) = \frac{\text{cosine}(\mathbf{x}_p, \mathbf{x}_q) + \text{cosine}(\mathbf{x}_c, \mathbf{x}_q)}{2}$$

2. **Single Layer Model**: A model that scores tuple by a standard single layer neural network which inputs the concatenation of the concept embeddings.
3. **Multiple Layer Model**: An extension of Single Layer Model that replaces the single layer neural network with multiple layer neural network.
4. **Bilinear Model** (Sutskever, Salakhutdinov, and Tenenbaum 2009; Jenatton et al. 2012): It incorporates the interaction of two concept embeddings through a simple and efficient bilinear form.
5. **Nerual Tensor Network** (Socher et al. 2013): It incorporates Single Layer Model with a bilinear tensor layer that directly relates the two concept embeddings across multiple dimensions and a bias vector.
6. **TaxoExpan** (Shen et al. 2020): One state-of-the-art taxonomy expansion framework which leverages position-enhanced graph neural network to capture local information and InfoNCE loss(Oord, Li, and Vinyals 2018) for robust training.
7. **ARBORIST** (Manzoor et al. 2020): One state-of-the-art taxonomy expansion model which aims for taxonomies with heterogeneous edge semantics and optimizes a large-margin ranking loss with a dynamic margin function.

Notably, except for the rule-based method Closest-Position, other baselines are learning-based method and designed for one-to-one matching. Thus we concatenate the embeddings of candidate's constituting concepts as candidate embedding to fit our one-to-pair setting. For fair comparison, we replace the GNN encoder of TaxoExpan with initial feature vector to align with other compared methods. There are other recently-proposed taxonomy expansion methods, *e.g.*, HiExpan (Shen et al. 2018b) and STEAM (Yu et al. 2020). We do not include them as baselines because they leverage external sources, *e.g.*, text corpus, to extract complicated features, while TMN and other baselines only take initial feature vectors as input.

Parameter Settings. For learning-based methods, we use Adam optimizer with initial learning rate 0.001 and ReduceLROnPlateau scheduler[3] with ten patience epochs. During model training, the batch size and negative sample size is set to 128 and 31, respectively. We set k, *i.e.*, the dimension of internal feature representation, to be 5. For TMN, we simply set $\lambda_1 = \lambda_2 = \lambda_3 = 1$ to avoid heavy hyperparameter tuning.

Experimental Results

Overall Performance. Table 2 presents the results of all compared methods on the four datasets. First, we find that learning-based methods clearly outperform rule-based Closest-Position method. Second, there is no baseline that could consistently outperform others in all taxonomies, which indicates the diversity of taxonomies of different domains and the difficulty of taxonomy completion task. Third, ARBORIST and TaxoExpan do not work well in taxonomy completion, which indicates that methods carefully designed for taxonomy expansion task will struggle in taxonomy completion task. Finally, our proposed TMN has the overall best performance across all the metrics and defeats the second best method by a large margin.

Table 3: Results on taxonomy expansion task.

Method	MAG-Psychology			
	MR	MRR	Recall@10	Prec@1
TaxoExpan	175.730	0.594	0.477	0.153
ARBORIST	119.935	0.722	0.629	0.258
TMN	69.293	0.740	0.646	0.329
Method	**WordNet-Verb**			
	MR	MRR	Recall@10	Prec@1
TaxoExpan	642.694	0.410	0.319	0.098
ARBORIST	608.668	0.380	0.277	0.067
TMN	464.970	0.479	0.379	0.132

Performance on Taxonomy Expansion. As taxonomy expansion being a special case of our novel taxonomy completion task, we are curious about how TMN performs on previous task. Thus, we compare TMN with ARBORIST and TaxoExpan on taxonomy expansion task[4]. The results are presented in Table 3. Notice that ARBORIST and TaxoExpan is trained directly on taxonomy expansion task, while TMN is trained solely on taxonomy completion task. From the results, we can see TMN outperforms the others in both dataset with a large margin, which indicates that TMN is able to solve taxonomy expansion task better than previous state-of-the-arts.

Table 4: Ablation analysis on MAG-Psychology and WordNet-Verb datasets.

Method	MAG-Psychology			
	MR	MRR	Recall@10	Prec@10
TMN w/o CG	265.729	0.385	0.298	0.061
TMN w/o s_1&s_2	258.382	0.458	0.368	0.075
TMN w/o s_1	269.058	0.471	0.382	0.123
TMN w/o s_2	229.306	0.474	0.381	0.078
TMN w/o s_3	342.021	0.326	0.213	0.043
TMN	212.298	0.471	0.377	0.077
Method	**WordNet-Verb**			
	MR	MRR	Recall@10	Prec@10
TMN w/o CG	1578.120	0.260	0.178	0.027
TMN w/o s_1&s_2	1497.843	0.278	0.200	0.030
TMN w/o s_1	1530.192	0.293	0.219	0.033
TMN w/o s_2	1586.740	0.290	0.207	0.031
TMN w/o s_3	1604.802	0.248	0.159	0.024
TMN	1445.801	0.304	0.215	0.032

Ablation Study. We conduct the ablation studies on two representative datasets **MAG-Psychology** and **WordNet-Verb**, and the results are presented in Table 4. The results show that without any of the key components of TMN, *i.e.*, auxiliary

[3] https://pytorch.org/docs/stable/optim.html\#torch.optim.lr_scheduler.ReduceLROnPlateau

[4] We sample validation/test set from leaf nodes for taxonomy expansion task.

scorers (s_1, s_2 and s_3) and channel-wise gating mechanism (CG), the overall performance will degrade by different extends, which indicates the effectiveness of the components.

Table 5: Efficiency Analysis.

Dataset	# of candidate pairs	Avg. running time per query (s)
MAG-CS	153,726	0.131
MAG-Psychology	101,077	0.067
WordNet-Verb	51,159	0.055
WordNet-Noun	799,735	0.870

Efficiency Analysis. At the training stage, our model uses $|\mathcal{N}^{(0)}|$ training instances every epoch and thus scales linearly to the number of concepts in the existing taxonomy. At inference stage, because the cardinality of candidate pairs is $|\mathcal{N}^{(0)}|^2$ without any restriction, for each query concept, we need to calculate $|\mathcal{N}^{(0)}|^2$ matching scores, one for every candidate pair. However, in practical, as we restrict the valid candidate pairs to be $\langle$ancestor, descendant$\rangle$ concept pairs in existing taxonomy, the number of candidates need to be considered is substantially reduced and therefore the inference efficiency is largely improved. Also, the inference stage can be further accelerated using GPU. We list the number of valid candidate pairs and the average running time per query during inference stage of all datasets in Table 5. From the table, we can see the number of valid candidate pairs is no more than ten times of $|\mathcal{N}^{(0)}|$ and thus the inference stage is quite efficient.

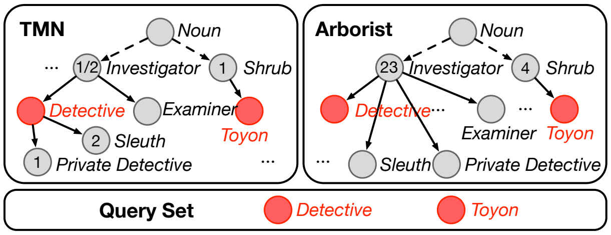


Figure 3: Case Study. The grey concepts are existing concepts while the red ones are queries needed to be inserted. The dash lines mean an omitting of some internal nodes and the solid lines indicate real edges in taxonomy. The numbers inside nodes are the ranking position output by the model. We can see TMN recovers the true positions for both internal and leaf concepts.

Case Study. We illustrate the power of TMN via two real query concepts "*Detective*" and "*Toyon*" of **WordNet-Noun** in Fig.3. For internal concept "*Detective*", TMN ranks the true positions $\langle$"*Investigator*", "*Private Detective*"$\rangle$ at top 1 and $\langle$"*Investigator*", "*Sleuth*"$\rangle$ at top 2, while Arborist can only rank the true parent "*Investigator*" at top 23. For leaf concept "*Toyon*", TMN recovers its true parent "*Shrub*" but Arborist ranks "*Shrub*" at top 5. We can see that TMN works better than baseline in terms of recovering true positions.

Related Work

Taxonomy Construction and Expansion. Automatic taxonomy construction is a long-standing task in the literature. Existing taxonomy construction methods leverage lexical features from the resource corpus such as lexical-patterns (Nakashole, Weikum, and Suchanek 2012; Jiang et al. 2017; Hearst 1992; Agichtein and Gravano 2000) or distributional representations (Mao et al. 2018; Zhang et al. 2018; Jin, Barzilay, and Jaakkola 2018; Luu et al. 2016; Roller, Erk, and Boleda 2014; Weeds, Weir, and McCarthy 2004) to construct a taxonomy from scratch. However, in many real-world applications, some existing taxonomies may have already been laboriously curated and are deployed in online systems, which calls for solutions to the taxonomy expansion problem. To this end, multitudinous methods have been proposed recently to solve the taxonomy expansion problem (Vedula et al. 2018; Shen et al. 2018b; Manzoor et al. 2020; Shen et al. 2020; Yu et al. 2020; Mao et al. 2020). For example, Arborist (Manzoor et al. 2020) studies expanding taxonomies by jointly learning latent representations for edge semantics and taxonomy concepts; TaxoExpan (Shen et al. 2020) proposes position-enhanced graph neural networks to encode the relative position of terms and a robust InfoNCE loss; STEAM (Yu et al. 2020) re-formulates the taxonomy expansion task as a mini-path-based prediction task and proposes to solve it through a multi-view co-training objective. However, all the existing taxonomy expansion methods aim for solving the one-to-one matching problem, *i.e.* to find the true parent/hypernym, which is incompatible to our novel one-to-pair matching problem induced by taxonomy completion task.

Auxiliary Learning. Auxiliary learning refers to a learning strategy that facilitates training of a primal task with auxiliary tasks (Ruder 2017; Shen et al. 2018a). Different from multitask learning, auxiliary learning only cares the performance of the primal task. The benefits of auxiliary learning have been proved in various applications (Standley et al. 2020; Tang et al. 2020; Trinh et al. 2018; Toshniwal et al. 2017; Hwang et al. 2020; Jaderberg et al. 2017; Odena, Olah, and Shlens 2017; Liu, Davison, and Johns 2019; Lin et al. 2019; Xiao et al. 2019). In most of these contexts, joint training with auxiliary tasks adds an inductive bias, encouraging the model to learn meaningful representations and avoid overfitting spurious correlations. Despite the numerous applications of auxiliary learning, its benefits on taxonomy construction remains less investigated. To our best knowledge, we are the first to leverage auxiliary learning to enhance taxonomy construction.

Conclusion

This paper studies taxonomy completion without manually labeled supervised data. We propose a novel TMN framework to solve the one-to-pair matching problem in taxonomy completion, which can be applied on other applications where one-to-pair matching problem exists. Extensive experiments demonstrate the effectiveness of TMN on various taxonomies. Interesting future work includes leveraging current method to cleaning the existing taxonomy, and incorporating feedback from downstream applications (*e.g.*, searching & recommendation) to generate more diverse (auxiliary) supervision signals for taxonomy completion.

Acknowledgement

Thanks the anonymous reviewers for their helpful comments and suggestions. We also thank Jingjing Xu, Hao Zhou, and Jiawei Han for their valuable discussions.

References

Agichtein, E.; and Gravano, L. 2000. Snowball: extracting relations from large plain-text collections. In *ACM DL*.

Aly, R.; Acharya, S.; Ossa, A.; Köhn, A.; Biemann, C.; and Panchenko, A. 2019. Every Child Should Have Parents: A Taxonomy Refinement Algorithm Based on Hyperbolic Term Embeddings. In *ACL*.

Hearst, M. A. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *COLING*.

Hua, W.; Wang, Z.; Wang, H.; Zheng, K.; and Zhou, X. 2017. Understand Short Texts by Harvesting and Analyzing Semantic Knowledge. *TKDE* .

Huang, J.; Ren, Z.; Zhao, W. X.; He, G.; Wen, J.-R.; and Dong, D. 2019. Taxonomy-Aware Multi-Hop Reasoning Networks for Sequential Recommendation. In *WSDM*.

Hwang, D.; Park, J.; Kwon, S.; Kim, K.; Ha, J.-W.; and Kim, H. J. 2020. Self-supervised Auxiliary Learning with Meta-paths for Heterogeneous Graphs. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 10294–10305.

Jaderberg, M.; Mnih, V.; Czarnecki, M. W.; Schaul, T.; Leibo, Z. J.; Silver, D.; and Kavukcuoglu, K. 2017. Reinforcement Learning with Unsupervised Auxiliary Tasks. *ICLR* .

Jenatton, R.; Roux, N. L.; Bordes, A.; and Obozinski, G. 2012. A Latent Factor Model for Highly Multi-Relational Data. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*.

Jiang, M.; Shang, J.; Cassidy, T.; Ren, X.; Kaplan, L. M.; Hanratty, T. P.; and Han, J. 2017. MetaPAD: Meta Pattern Discovery from Massive Text Corpora. In *KDD*.

Jin, W.; Barzilay, R.; and Jaakkola, T. S. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. In *ICML*.

Jurgens, D.; and Pilehvar, M. T. 2016. SemEval-2016 Task 14: Semantic Taxonomy Enrichment. In *SemEval@NAACL-HLT*.

Karamanolakis, G.; Ma, J.; and Dong, X. L. 2020. TXtract: Taxonomy-Aware Knowledge Extraction for Thousands of Product Categories. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics.

Lin, X.; Baweja, H.; Kantor, G.; and Held, D. 2019. Adaptive Auxiliary Task Weighting for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 4773–4784.

Lipscomb, C. E. 2000. Medical Subject Headings (MeSH). *Bulletin of the Medical Library Association* .

Liu, B. W.; Guo, W.; Niu, D.; Wang, C.; Xu, S.-Z.; Lin, J.; Lai, K.; and Xu, Y. W. 2019. A User-Centered Concept Mining System for Query and Document Understanding at Tencent. In *KDD*.

Liu, S.; Davison, A.; and Johns, E. 2019. Self-supervised generalisation with meta auxiliary learning. In *Advances in Neural Information Processing Systems*, 1677–1687.

Luu, A. T.; Tay, Y.; Hui, S. C.; and Ng, S.-K. 2016. Learning Term Embeddings for Taxonomic Relation Identification Using Dynamic Weighting Neural Network. In *EMNLP*.

Manzoor, E.; Li, R.; Shrouty, D.; and Leskovec, J. 2020. Expanding Taxonomies with Implicit Edge Semantics. In *Proceedings of The Web Conference 2020*, 2044–2054.

Mao, Y.; Ren, X.; Shen, J.; Gu, X.; and Han, J. 2018. End-to-End Reinforcement Learning for Automatic Taxonomy Induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2462–2472. Association for Computational Linguistics.

Mao, Y.; Tian, J.; Han, J.; and Ren, X. 2019. Hierarchical text classification with reinforced label assignment. In *EMNLP*.

Mao, Y.; Zhao, T.; Kan, A.; Zhang, C.; Dong, X. L.; Faloutsos, C.; and Han, J. 2020. Octet: Online Catalog Taxonomy Enrichment with Self-Supervision. In *KDD*.

Nakashole, N.; Weikum, G.; and Suchanek, F. M. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *EMNLP-CoNLL*.

Odena, A.; Olah, C.; and Shlens, J. 2017. Conditional Image Synthesis with Auxiliary Classifier GANs. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 2642–2651. International Convention Centre, Sydney, Australia: PMLR.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* .

Roller, S.; Erk, K.; and Boleda, G. 2014. Inclusive yet Selective: Supervised Distributional Hypernymy Detection. In *COLING*.

Ruder, S. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* .

Shen, J.; Karimzadehgan, M.; Bendersky, M.; Qin, Z.; and Metzler, D. 2018a. Multi-task learning for email search ranking with auxiliary query clustering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2127–2135.

Shen, J.; Shen, Z.; Xiong, C.; Wang, C.; Wang, K.; and Han, J. 2020. TaxoExpan: Self-Supervised Taxonomy Expansion with Position-Enhanced Graph Neural Network. In *Proceedings of The Web Conference 2020*, WWW '20.

Shen, J.; Wu, Z.; Lei, D.; Zhang, C.; Ren, X.; Vanni, M. T.; Sadler, B. M.; and Han, J. 2018b. HiExpan: Task-Guided Taxonomy Construction by Hierarchical Tree Expansion. In *KDD*.

Shen, J.; Xiao, J.; He, X.; Shang, J.; Sinha, S.; and Han, J. 2018c. Entity set search of scientific literature: An unsupervised ranking approach. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 565–574.

Sinha, A.; Shen, Z.; Song, Y.; Ma, H.; Eide, D.; Hsu, B.-J. P.; and Wang, K. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *WWW*.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, 926–934.

Standley, T.; Zamir, A. R.; Chen, D.; Guibas, L. J.; Malik, J.; and Savarese, S. 2020. Which Tasks Should Be Learned Together in Multi-task Learning? In *ICML*.

Sutskever, I.; Salakhutdinov, R.; and Tenenbaum, J. B. 2009. Modelling Relational Data Using Bayesian Clustered Tensor Factorization. In *Advances in neural information processing systems*, volume 22, 1821–1828.

Tang, L.; Chen, K.; Wu, C.; Hong, Y.; Jia, K.; and Yang, Z. 2020. Improving Semantic Analysis on Point Clouds via Auxiliary Supervision of Local Geometric Priors. *IEEE Transactions on Cybernetics* .

Toshniwal, S.; Tang, H.; Lu, L.; and Livescu, K. 2017. Multitask Learning with Low-Level Auxiliary Tasks for Encoder-Decoder Based Speech Recognition. In *Proc. Interspeech 2017*.

Trinh, T.; Dai, A.; Luong, T.; and Le, Q. 2018. Learning Longer-term Dependencies in RNNs with Auxiliary Losses. In *International Conference on Machine Learning*, 4965–4974.

Vedula, N.; Nicholson, P. K.; Ajwani, D.; Dutta, S.; Sala, A.; and Parthasarathy, S. 2018. Enriching Taxonomies With Functional Domain Knowledge. In *SIGIR*.

Vrandecic, D. 2012. Wikidata: a new platform for collaborative data collection. In Mille, A.; Gandon, F. L.; Misselis, J.; Rabinovich, M.; and Staab, S., eds., *WWW (Companion Volume)*, 1063–1064. ACM.

Weeds, J.; Weir, D. J.; and McCarthy, D. 2004. Characterising Measures of Lexical Distributional Similarity. In *COLING*.

Wu, W.; Li, H.; Wang, H.; and Zhu, K. Q. 2012. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD Conference*.

Xiao, S.; Ouyang, Y.; Rong, W.; Yang, J.; and Xiong, Z. 2019. Similarity Based Auxiliary Classifier for Named Entity Recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Yang, C.; Zhang, J.; and Han, J. 2020. Co-Embedding Network Nodes and Hierarchical Labels with Taxonomy Based Generative Adversarial Networks. In *ICDM*.

Yang, G. H. 2012. Constructing Task-Specific Taxonomies for Document Collection Browsing. In *EMNLP-CoNLL*.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*.

Yu, Y.; Li, Y.; Shen, J.; Feng, H.; Sun, J.; and Zhang, C. 2020. STEAM: Self-Supervised Taxonomy Expansion with Mini-Paths. In *KDD*.

Zhang, C.; Tao, F.; Chen, X.; Shen, J.; Jiang, M.; Sadler, B. M.; Vanni, M. T.; and Han, J. 2018. TaxoGen: Constructing Topical Concept Taxonomy by Adaptive Term Embedding and Clustering. In *KDD*.

Zhang, Y.; Ahmed, A.; Josifovski, V.; and Smola, A. J. 2014. Taxonomy discovery for personalized recommendation. In *WSDM*.

Supplementary Materials

Discussion of Simplified Problem

The simplified problem in Section defines $|\mathcal{C}|$ *independent* optimization problems and each problem aims to rank candidate positions of a new concept n_q. Therefore, we essentially reduce the more generic taxonomy completion problem into $|C|$ independent simplified problems and tackle it by inserting new concepts *one-by-one* into the existing taxonomy. As a result of the above reduction, possible interactions among new concepts are ignored and we leave it to the future work.

Edge Modification of taxonomy completion

It is worth noting that when a new concept n_q is inserted into existing edge $\langle n_p, n_c \rangle$ in taxonomy, this edge will be break into two new edges $\langle n_p, n_q \rangle$ and $\langle n_q, n_c \rangle$. Therefore, the initial edge set $\mathcal{E}^0$ will be modified, while in taxonomy expansion the final edge set $\mathcal{E}'$ is simply the union set of initial edge set $\mathcal{E}^0$ and newly-discovered edge set $\mathcal{R}$, which makes the $\mathcal{E}'$ a superset of $\mathcal{E}^0$.

Optimality of taxonomy completion

In streaming setting where incoming concepts should be processed one-by-one without interactions among them being considered, an optimal taxonomy completion model, *e.g.*, TMN, could recover the ground truth taxonomy regardless of the order of incoming concepts. However, an optimal taxonomy expansion model can only recover the ground truth taxonomy when the order of incoming concepts is optimized, which is unrealistic in online streaming setting. We illustrate this difference using a tony example as shown in Fig.4 and Fig.5. Notice that in the tony example, the optimal taxonomy expansion model can always find the best hypernym concepts in existing taxonomy and the optimal taxonomy completion model always outputs best ⟨hypernym, hyponym⟩ pairs.

Learning Algorithm

We summarize our self-supervised learning procedure in Algorithm 1.

Discussion on Other Taxonomy Expansion Methods

Except for ARBORIST (Manzoor et al. 2020), there are several other recently-proposed methods on taxonomy expansion: HiExpan (Shen et al. 2018b), STEAM(Yu et al. 2020) and OCTET (Mao et al. 2020). Here, we explain why we do not compare with them in detail. In our setting, only concept embedding is available, because it is easy to obtain in real-world online streaming setting. For example, one can use pre-trained language model to obtain contextualized embedding with only a few of sentences in hand. However, most of aforementioned taxonomy expansion models require external resources which are difficult to collect in online streaming setting. First, HiExpan consists of a set expansion module and a relation extraction module, both of which rely on an external text corpus to extract skip text patterns and to learn concept embedding. We argue that such a text corpus rich with useful skip text patterns is uneasy, if not infeasible, to obtain in online streaming setting. The same as STEAM, where an external corpus is needed to extract contextual features and lexical-syntactic features. The contextual features require two terms to occur in some sentences and lexical-syntactic features require the frequency of terms in the corpus. For OCTET, it is specifically designed for catalog data in e-commerce field where user queries and click logs are ready to be used to build query-item-taxonomy graph, which is not applicable to other domains.

Algorithm 1: Self-supervised learning of TMN

```
Input: A taxonomy $\mathcal{T}^0$; negative size $N$, batch size $B$;
       model $f(\cdot|\Theta)$.
Output: Learned model parameters $\Theta$.
Randomly initialize $\Theta$;
while $\mathcal{L}(\Theta)$ in Eq. (13) not converge do
    Enumerate nodes in $\mathcal{T}^0$ and sample $B$ nodes without
      replacement;
    $\mathbb{D} = \{\}$ # current batch of training instances;
    for each sampled node $n_q$ do
        Select one of its parents $n_p$ and one of its children
          $n_c$ to construct positive triple $\langle n_q, \langle n_p, n_c \rangle \rangle$
          Generate $N$ negative triples $\{\langle n_q, \langle n_p^l, n_c^l \rangle \rangle|_{l=1}^N\}$;
        $\mathbb{D} \leftarrow \mathbb{D} \cup$
          $\{\langle n_q, \langle n_p, n_c \rangle \rangle, \langle n_q, \langle n_p^1, n_c^1 \rangle \rangle, \ldots, \langle n_q, \langle n_p^N, n_c^N \rangle \rangle\}$;
    Update $\Theta$ based on $\mathbb{D}$.
Return $\Theta$;
```

Hyperparameter Effectiveness Study

We conduct additional experiments to study the effectiveness of six hyperparameters of TMN. The results can be found in Fig.6 and Fig.7. In the comparison among different chosen values for λ_1, λ_2, and λ_3, there is no significant change in terms of evaluation metrics. It can be concluded that our TMN model is robust to the weight hyperparameter of losses. We noticed that the increasing "Batch Size" can benefit the performance but overlarge batch deccreases the performance in the MAG-Psy dataset. In the study of other hyperparameter settings, we noticed marginal increase in the performance with the "k" and "Negative Sampling Size" go up. However, increasing such hyperparameters introduces additional computation cost.

Extended Case Study

Due to the page limit, we show more case studies in Figure 8. The queries we selected here are more representative and capture more aspects than the demo cases in the paper.

Hardware

Experiments were run on an Intel(R) Xeon(R) Platinum 8260 CPU at 2.40GHz with 24 cores and 30GB of main memory, as well as a Tesla V100-SXM2 GPU. Implementations were in PyTorch 1.4.0 / Python 3.

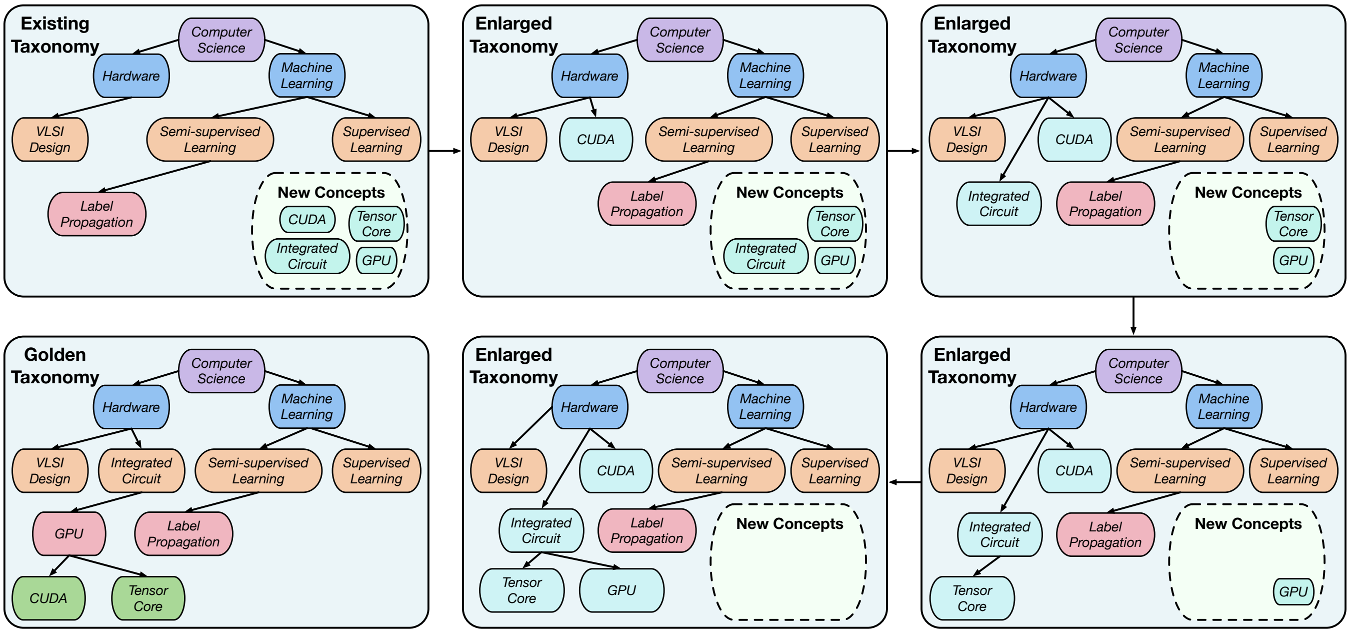


Figure 4: Illustration on sequentially applying optimal taxonomy expansion model to complete an existing taxonomy.

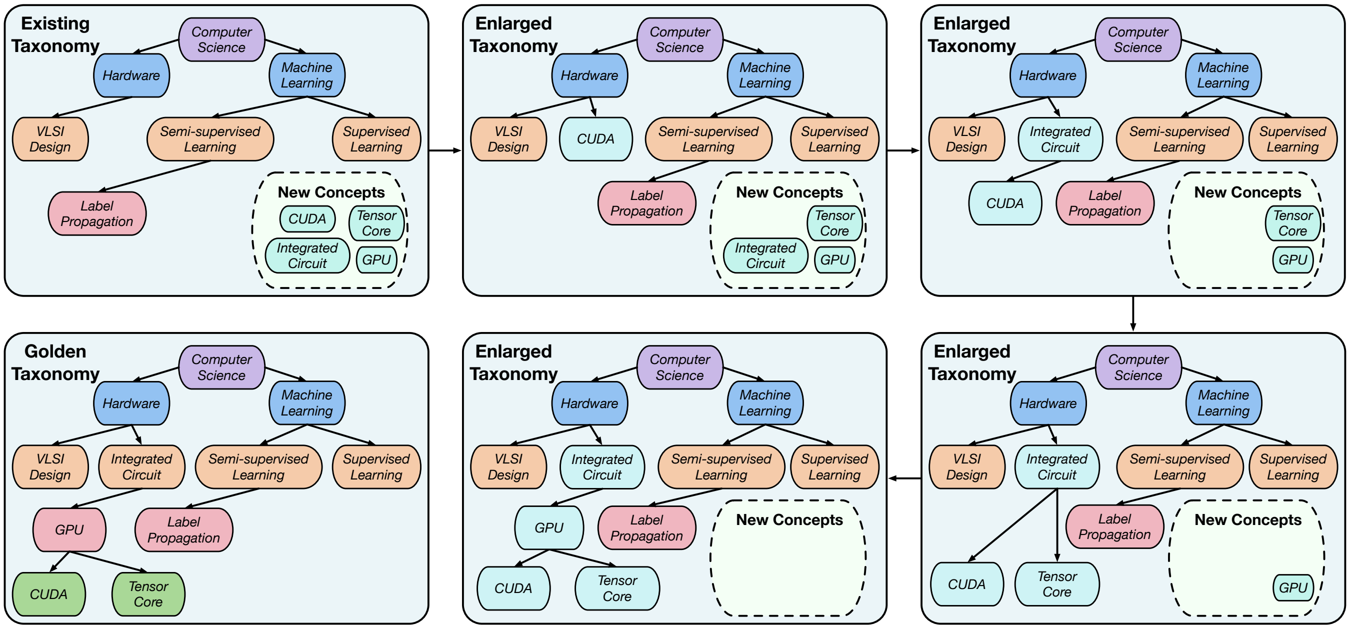
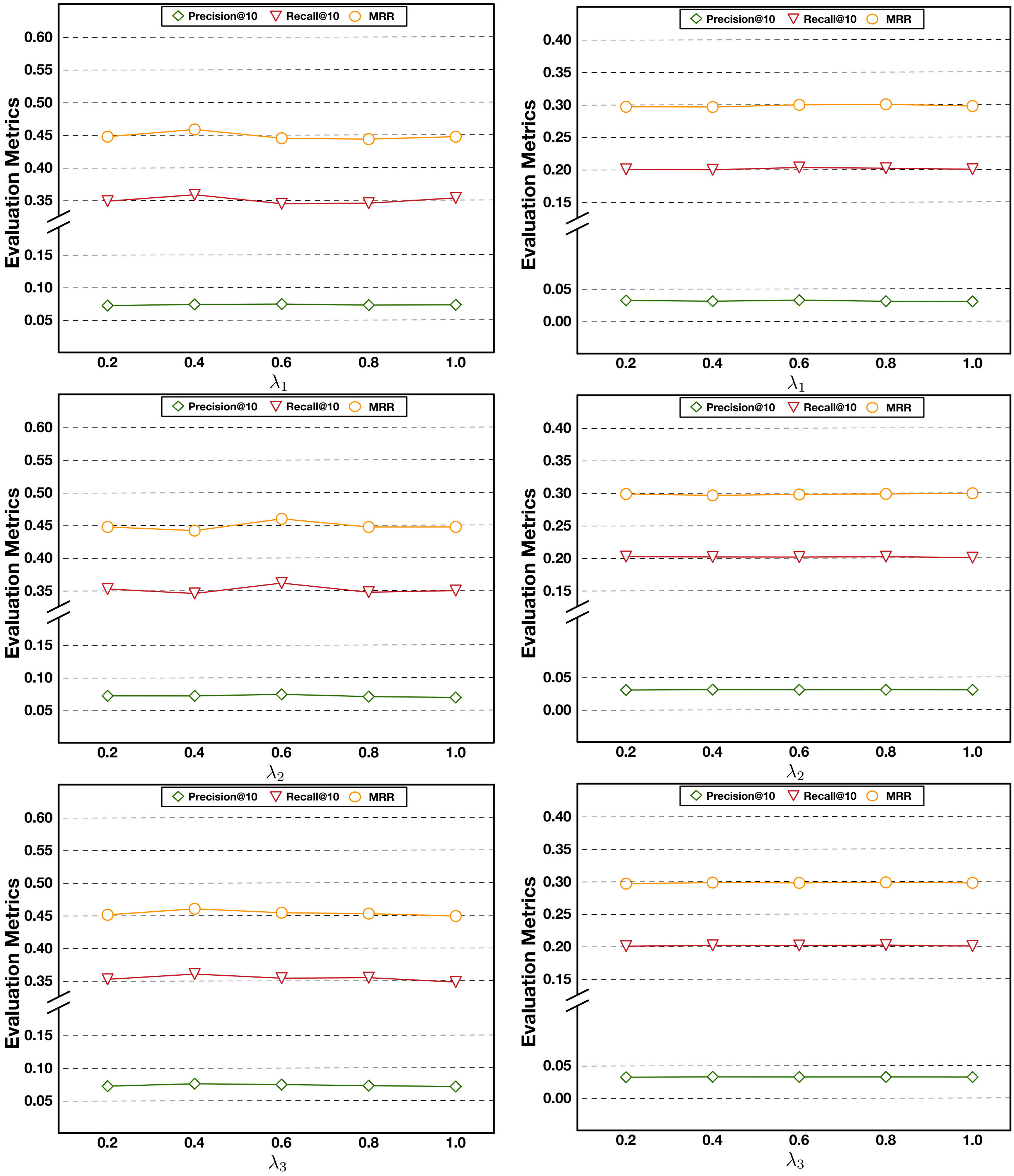


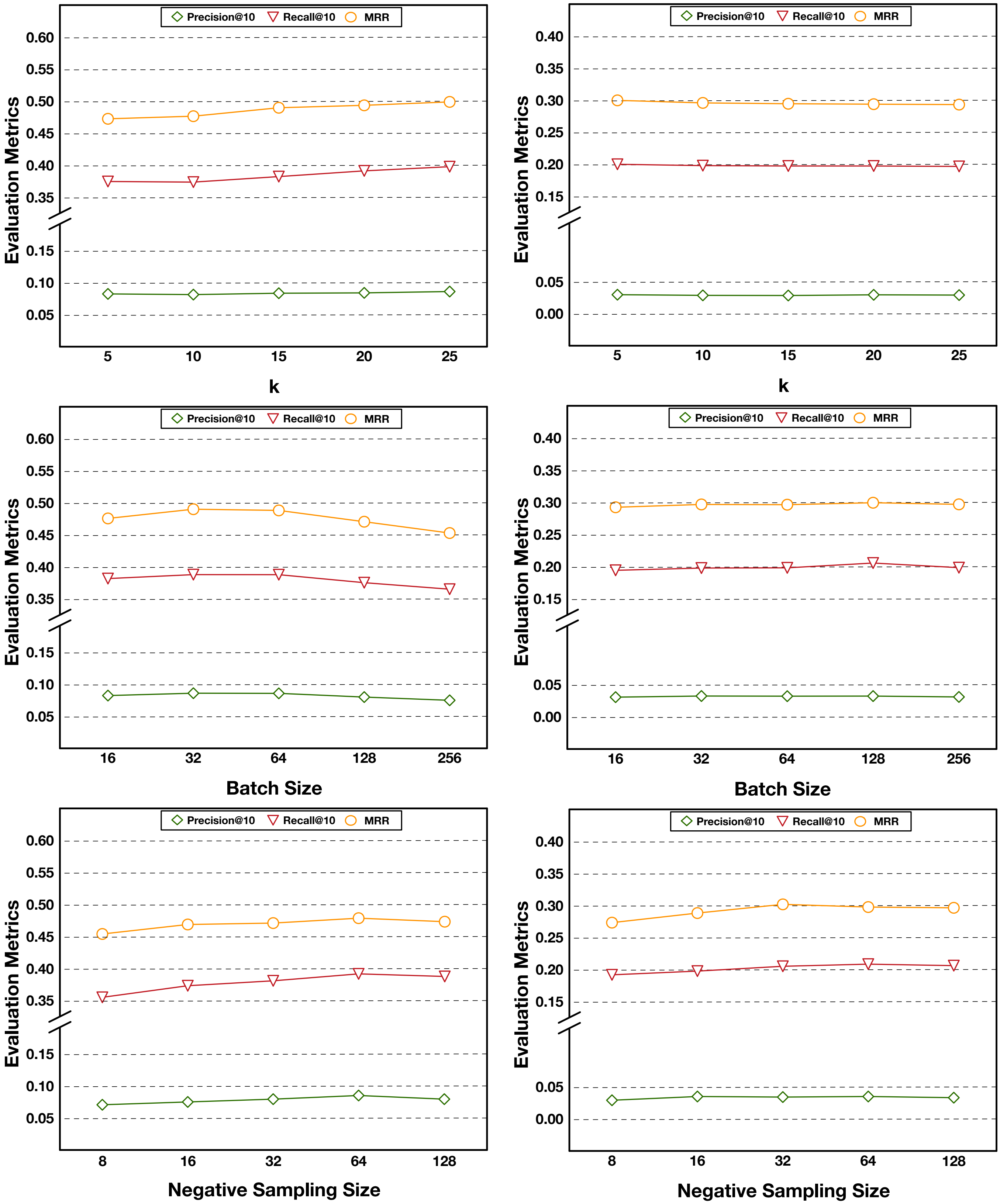
Figure 5: Illustration on sequentially applying optimal TMN model to complete an existing taxonomy.



(a) MAG-Psychology

(b) WordNet-Verb

Figure 6: Hyperparameter Effectiveness Study



(a) MAG-Psychology

(b) WordNet-Verb

Figure 7: Hyperparameter Effectiveness Study Continue

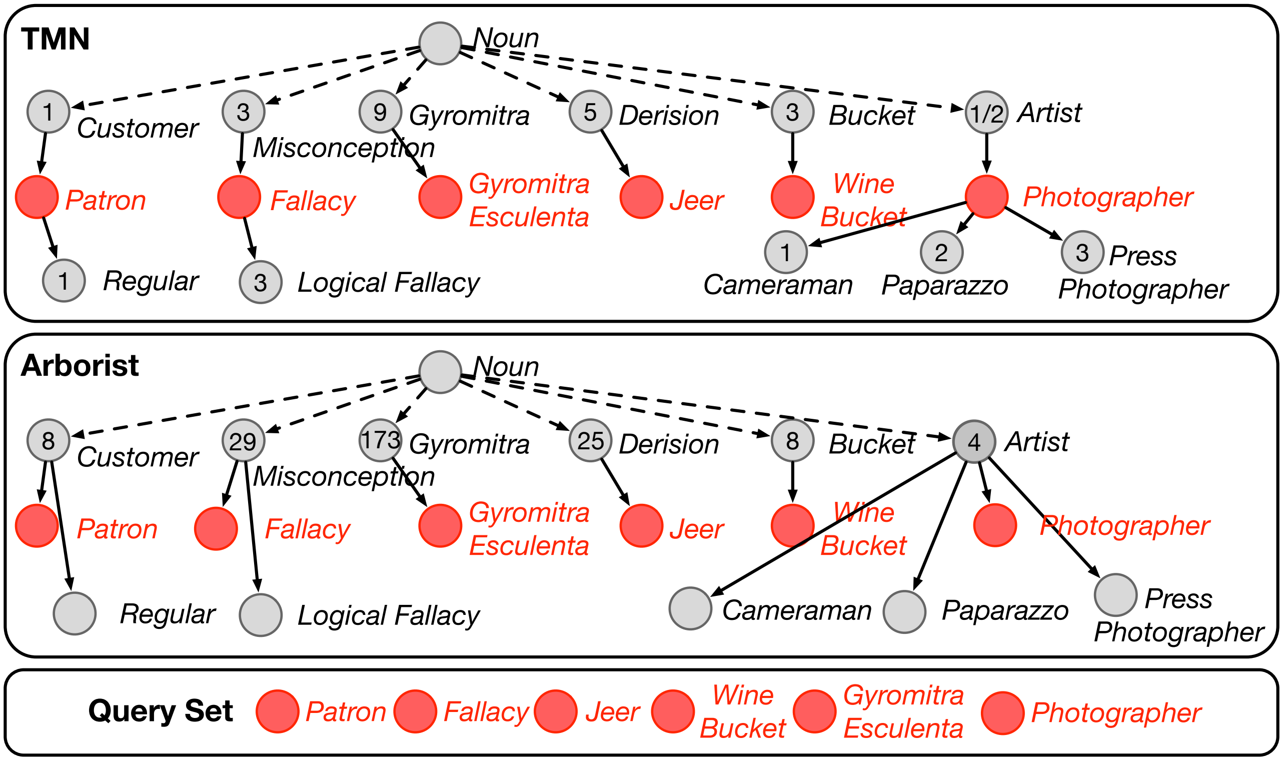


Figure 8: Extended Case Study.