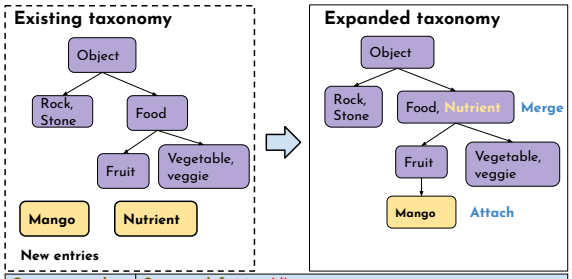
TEAM: A multitask learning based Taxonomy Expansion approach for Attach and Merge

Bornali Phukon[1]*, **Anasua Mitra**[2]*, **Sanasam Ranbir Singh**[1,2], **Priyankoo Sarmah**[1]
[1]Centre for Linguistic Science and Technology
[2]Department of Computer Science and Engineering
Indian Institute of Technology Guwahati, Assam, India
{mamunp, anasua.mitra, ranbir, priyankoo}@iitg.ac.in

Abstract

Taxonomy expansion is a crucial task. Most of the taxonomy expansion approaches are of two types, *attach* and *merge*. In a taxonomy like WordNet, both merge and attach are integral parts of the expansion operations, but the majority of studies consider them separately. This paper proposes a novel multi-task learning-based deep learning method known as *Taxonomy Expansion with Attach and Merge (TEAM)* that performs both the merge and attach operations. This is the first study that integrates both the merge and attach operations in a single model to the best of our knowledge. The proposed models have been evaluated on three separate WordNet taxonomies, viz., Assamese, Bangla, and Hindi. From the various experimental setups, it is shown that TEAM outperforms its state-of-the-art counterparts for attach operation and also provides highly encouraging performance for the merge operation.



Concept- words Synset (s)	Concept definition (d)
Object	A tangible and visible entity.
Rock, Stone	A lump or mass of hard consolidated mineral matter.
Food	Any substance that can be metabolized by an organism to give energy and build tissue.
Vegetable,Veggie	Edible seeds or roots or stems or leaves or bulbs or tubers or non sweet fruits of any of numerous herbaceous plant
Fruit	The ripened reproductive body of a seed plant.
Nutrient	A substance used by an organism to survive, grow, and reproduce.
Mango	A mango is an edible stone fruit.

Figure 1: Example of WordNet taxonomy expansion with attach and merge operations to include new terms *"Mango"* and *"Nutrient"*.
"Mango" is a specific concept of *Fruit* not present in the existing WordNet. Hence, a new concept node is created in the taxonomy by attaching it to its generic concept *Fruit* . As *"Nutrient"* refers to the same concept as *"Food"*, no new concept is created. *"Nutrient"* is merged with the existing concept *"Food"*.

1 Introduction

Taxonomy, such as the WordNet, is a crucial resource for developing NLP related technologies, as it plays a vital role in various text processing tasks such as information retrieval, information extraction, text classification, summarization, etc. (Pang et al., 2008; Allan et al., 1998; Singhal et al., 2001) (Miller, 1998). As most of the WordNets are manually curated, it often suffers from the problem of limited coverage. Therefore, an automatic taxonomy expansion is a crucial problem to handle the above issue. For taxonomy expansion, WordNet in particular, may need two types of operations; (i) *merge*, where a new concept [1] is merged to an existing node, and (ii) *attach*, where a new concept is inserted as a new node. Figure 1 illustrates these two operations where the word *Mango* is inserted as a new concept with the attach operation, and the word *Nutrient* is inserted as a new synonymy in an existing concept with the merge operation.

Though both of these operations are integral parts of a WordNet taxonomy expansion, all of the existing studies on taxonomy expansion have considered expansion with either attach operation (Schlichtkrull and Alonso, 2016; Vedula et al., 2018; Shen et al., 2020; Yu et al., 2020b; Zhang et al., 2021; Takeoka et al., 2021; Liu et al., 2021) or merge operation (Nakashole et al., 2012; Nguyen et al., 2017; Nakashole et al., 2012; Qu et al., 2017; Boteanu et al., 2018; Wang et al., 2019; Fei et al., 2019), but not together. Realizing the need to apply both the operation, SemEval-2016:task 14 (Semantic taxonomy enrichment) Jurgens and Pilehvar (2016) includes a call for expansion with both attach and merge operations. However, none of the submissions incorporate both operations in a single model.

*Equal contributions.
[1]Concept is a basic building block of WordNet, which refers a definition with associated synonym words

Findings of the Association for Computational Linguistics: NAACL 2022, pages 366 - 378
July 10-15, 2022

Motivated by the above observations, in this study, we propose an integrated deep learning-based method, namely, *Taxonomy Expansion with Attach and Merge* (TEAM), which performs both the attach and merge operations in a multitask-learning framework. Though most of the existing studies consider the expansion a regression problem (Shen et al., 2020; Yu et al., 2020b; Zhang et al., 2021), considering that our method performs both the attach and merge operation in a single model, it can also be considered a classification task. As a result, we propose two versions of TEAM, namely, TEAM-RG: *Regression*, and TEAM-CL: *Classification* to perform with explicit and implicit rankings. The proposed models have been evaluated on three different WordNet taxonomies, viz., Assamese, Bangla, and Hindi. From the various experimental setups, it is observed that the proposed TEAM-RG and TEAM-CL outperform their baselines counterparts for attach operation, and also obtained encouraging performance for merge operation as well. The major contributions of the paper are summarized as follows:

- A multi-task learning based taxonomy expansion framework *TEAM* is jointly trained to perform both the *Attach* and *Merge* operations. To the best of our knowledge, it is the first integrated model to perform both the *Attach* and *Merge* operations in a single model.
- Two variants of TEAM, namely *TEAM-Regression (RG)* and *TEAM-Classification (CL)* are proposed.

2 Taxonomy Expansion - Attach and Merge

In this study, we have considered WordNets as our target taxonomies. A WordNet may be defined by a collection of concepts connected by various semantic relationships such as *hypernymy, hyponymy, troponymy*, etc., where each concept is further defined by a set of attributes such as *definition, synonyms, examples*, etc (Bhattacharyya, 2010). In this study, we have considered only the hypernymy relation and the *definition* and *synonymy* attributes.

In order to be able to apply the proposed model, we first transform the original WordNet taxonomy into an experimental intermediate taxonomy (*directed unweighted acyclic graph*) $\mathcal{T} = (V, E)$ where V represents the set of concepts and E represents the set of hypernymy relations between the concepts. A concept $v \in V$ is further defined by a tuple $v = (d_v, s_v)$ where d_v represents the *definition* of the concept, and s_v represents the set of associated synonyms. An edge $e \in E$ represents a hypernymy relation from a parent concept v_p to its child concept v_{ch} and is denoted as $e : (v_p \xrightarrow{\text{hyper}} v_{ch})$. The taxonomy $\mathcal{T}$ is arranged in a hierarchical manner with directed edges in E, as shown in Figure 1. Given the taxonomy $\mathcal{T}$ and a query concept $q = (d_q, s_q)$, the *attach* and the *merge* expansion operations are defined below.

Attach (A) — An attach operation is performed when the concept q is not present in $\mathcal{T}$. The objective of the attach operation is to identify the best matching parent node in taxonomy network known as anchor concept $a = (d_a, s_a)$, and insert a new concept q with an edge $e : (a \xrightarrow{\text{hyper}} q)$. In a taxonomy network, a parent node represents a more generic concept of its children. After an attach operation i.e., insertion of q in $\mathcal{T}$ under the anchor a, the expanded taxonomy is updated as follows.

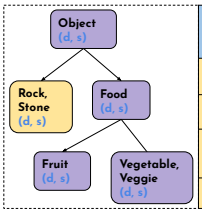
$$\mathcal{T} = (V \cup \{q\}, E \cup \{e\}) \quad (1)$$

Merge (M) — A merge operation is performed when an equivalent concept $a = (d_a, s_a)$ of the query q (i.e., $d_a \equiv d_q$) is already present in $\mathcal{T}$, but the synset s_q is not present in a (i.e., $s_q \cap s_a = \emptyset$). The objective of the merge operation is to identify the best matching concept $a = (d_a, s_a)$, known as the anchor concept, in the taxonomy network $\mathcal{T}$ and add the synset s_q to s_a. It neither creates a new node nor adds a new edge. It only updates the synset of the anchor concept. After the merge operation, the updated anchor concept in the expanded taxonomy can be expressed as follows.

$$a = (d_a, s_a \cup s_q) : a \in V \quad (2)$$

3 Proposed Methods

Our objective is to develop an integrated model that performs both attach and merge operations for taxonomy expansion. Since we have two tasks to unify in a single model, we resort to a multi-task learning framework known as **T**axonomy **E**xpansion Framework with ***Attach*** and ***Merge*** (TEAM). This joint learning objective facilitates information flow so that the two tasks can aid each other. Also, we are interested in deciding which expansion operation is to perform given a triplet (expansion task classification) and retrieving the ranked list of candidates



Query	Anchor	Label	Operation
Rock(d, "rock")	Object (d, s)	True	Attach
Rock(d, "rock")	Fruit (d,s)	False	
Rock(d, "rock")	Stone (parallel_d, s)	True	Merge
Rock(d, "rock")	Fruit(d, s)	False	

Figure 2: Example of training dataset generation. The table shows positive and negative training instances corresponding to the query concept *"Rock"* for both operations *Attach* and *Merge*.

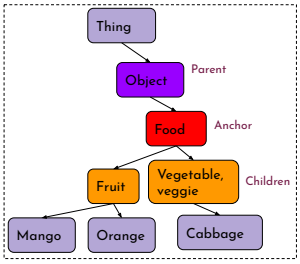


Figure 3: Ego tree of the anchor node *"Food"*. 1-hop ego-tree is extracted around the anchor *"Food"*. The color-codes distinguish various roles w.r.t the anchor node *"Food"*, eg., Deep Purple: Grand-parent, Red: Anchor/ Parent, Orange: Childrens

(ranking) as prospective anchors to associate the query with. For this *first-of-its-kind* novel taxonomy expansion task, we propose two versions of TEAM, namely *TEAM–Regression* (TEAM-RG) and *TEAM–Classification* (TEAM-CL) — where we show that using either regression or classification learning objectives, this task can be accomplished.

3.1 Training dataset generation

Given a transformed taxonomy $\mathcal{T}$ (as described in Section 2), we generate a training dataset for building the model as follows. The training samples are defined by a 3-tuple $< q, a, label >$, where q is the query, a is the potential anchor, and *label* is associated class, i.e., true/false (1/0). We randomly select a set of nodes in $\mathcal{T}$ as a set of queries [2], and generate the training samples for the attach and the merge operations separately as follows.

Attach (A) — We first remove the query nodes from the $\mathcal{T}$. For each query $q = (d_q, s_q)$, we consider its parent as anchor node $a = (d_a, s_a)$ and generate positive sample $< q, a, TRUE >$. We then randomly pick up $\mathcal{N}$ number other nodes $a' = (d_{a'}, s_{a'})$, and generate N negative samples $< q, a', FALSE >$. Thus, for a given query node q, we extract one positive and $\mathcal{N}$ negative samples.

Merge (M) — For each of the randomly selected query node $x = (d_x, s_x >)$ in $\mathcal{T}$, we generate the following positive training sample $< q, x, True >$ where $q = (d_x, s_q), s_q \subset s_x$ is the query and $x = (d_x, s_x - s_q)$ is the anchor.The s_q is a randomly selected synonym in s_x. Unlike attach, for generating the training sample for the query q, we only remove the query synset s_q from the anchor synset s_x i.e., $s_x = s_x - s_q$, and, not the node. Like attach, we randomly pick up $\mathcal{N}$ number other nodes a', and generate N negative samples $< q, a', FALSE >$.

[2] As we consider the same query set for both attach and merge experiments, nodes with at least two synonyms are considered.

Figure 2 illustrates the generation of the training samples from a taxonomy.

3.2 TEAM-Regression (TEAM-RG)

The proposed *TEAM-RG* works in two tiers process. Given a training input sample $< q, a, c >$, it first generates encoding of the query q and the anchor a. It then merges to a shared layer to produce two different multi-tasking dense networks; one for merge and another for attach, as shown in figure 4.D.

For learning embedding of the anchor concept from the taxonomy network and the query concept from the associated attributes, we consider the publicly available Fasttext pre-trained embedding available at `https://fasttext.cc/docs/en/crawl-vectors.html`.

Processing of the query concept: As mentioned in Section 2, a query concept consists of its definition and the associated synset i.e., $q = (d_q, s_q)$. The definition is a piece of text describing the concept, and the synset is a synonym associated with the query concept. . The two embeddings are then concatenated to represent the query.

Processing of the anchor concept: For generating the encoding of the anchor concept, we exploit the proximity structure of the nodes in the taxonomy $\mathcal{T}$. For a given anchor node $a \in \mathcal{T}$, we first extract its ego-tree from the taxonomy. An ego tree $\mathcal{T}_a : (V_a, E_a)$ of a node a in the taxonomy $\mathcal{T}$ is a sub-tree that comprises the node a and its k-hop neighborhood nodes. In this study, we considered $k = 1$, i.e., the anchor node, its parent node, and all its children nodes. Figure 3 illustrates an example of an ego tree. A similar approach has also been used in (Wang et al., 2021; Yu et al., 2020b; Zhang et al., 2021; Shen et al., 2020) studies. To obtain the embedding of the anchor concept, we further apply graph embedding as described below.

3.2.1 Embedding Ego-tree

Ideally, we should be able to use any graph embedding method to obtain the embedding of the anchor node. As the objective is to incorporate the positional information of the parent and children node in the ego tree, we use the Graph Attention Network (GAT) proposed in Taxo-Expan (Shen et al., 2020). This GAT is a special type of graph neural network (GNN) (Kipf and Welling, 2016) with a neighborhood-based attention mechanism. The details of GAT and its difference from GNN are given in Section B of Appendix. Thus we used position enhanced GAT to obtain the node embeddings of an anchor's ego tree.

We summarize the tree by applying an activation function over the average of the embedding vectors of all nodes in the ego-tree as given in equation 3 to define the encoding of the anchor node.

$$\bar{\mathcal{T}}_a = \sigma\Big(\frac{1}{|V_a|}\sum_{x \in V_a} \bar{x}\Big) \tag{3}$$

where $\sigma(.)$ is an activation function. We have considered *Sigmoid* function in this study.

3.2.2 Multi-task Learning

Once we obtain the embeddings of the anchor and query concepts, the concatenated vector is subjected to a shared dense layer and then build two multi-task layers to perform the merge and attach operations as shown in Figure 4.D. Given a query concept and its true anchor concept with $\mathcal{N}$ false anchor concepts, the task is to design a regression-based ranking model such that the true concept is ranked higher than the $\mathcal{N}$ false concepts. This objective should be realized for all the queries in the training dataset.

Given the embedding vectors of anchor $\bar{a}$ and query $\bar{q}$ as learned above, we first estimate similarity between the two using a bi-linear model proposed in (Gutmann and Hyvärinen, 2010). It learns the discrimination between q and a through a learnable bi-linear scoring matrix $B \in \mathbb{R}^{|\bar{q}|\times|\bar{a}|}$ via a function $\mathcal{D} : \mathbb{R}^{|\bar{q}|\times|\bar{a}|} \mapsto \mathbb{R}$ as follows.

$$\mathcal{D}(q, a) = \sigma(\bar{q}^T B \bar{a}) \tag{4}$$

Here σ is sigmoid non-linearity. The output of this matching module is a probability estimate indicating the strength of association between the query and anchor. Now, considering the query concept q and its associated $\mathcal{N}+1$ anchor concepts, we estimate the probability of being the correct anchor using InfoNCE loss proposed in (Oord et al., 2018). Let $\mathcal{X}$ be a set of query concepts and their respective $\mathcal{N}+1$ anchor nodes (one positive and $\mathcal{N}$ negative). An element of $x_q \in \mathcal{X}$ for a given query q consists of $\{(q, a, 1), (q, a'_1, a'_2, ..., a'_{\mathcal{N}}, 0)\}$, where a is the positive anchor, and a' are the negative anchors of q. InfoNCE estimates loss function using an average probability of being true anchor node across the dataset $\mathcal{X}$ as follows.

$$\mathcal{L}_{A/M} = -\frac{1}{|\mathcal{X}|}\sum_{x_q \in \mathcal{X}} log\frac{\mathcal{D}(\bar{q}, \bar{a})}{\sum_{v \in \mathcal{M}(q)} \mathcal{D}(\bar{q}, \bar{v})} \tag{5}$$

where $\mathcal{M}(q)$ denotes the set of both positive and negative anchors of q. As mentioned earlier, the loss defined in Equation 5 is estimated separately for attach and merge operations. Therefore, we generate two different training datasets for attach and merge, and estimate $\mathcal{L}_A$ and $\mathcal{L}_M$ separately using respective datasets, The final model loss is defined as $\mathcal{L} = \mathcal{L}_A + \mathcal{L}_M$ — considering both the operations attach and merge.

3.3 TEAM-Classification (TEAM-CL)

Figure 4E shows the schematic diagram of the *TEAM-CL*. We use the identical representations for query q and candidate anchors a_A, a_M as described for TEAM-RG. We also adopt the same position-enhanced graph propagation and read-out modules as described in Section 3.2.1 for learning anchor $a = (d_a, s_a)$ concept representation. Once we obtain the query and anchor representations, we model the strength of association of an input query and the candidate anchors based on their features to predict the expansion task i.e., merge M or attach A. The matching module, a multi-layer perceptron (MLP) based classifier, takes the features of query $\bar{q} \in \mathbb{R}^{|\bar{q}|}$ and anchor $\bar{a} \in \mathbb{R}^{|\bar{a}|}$, and generates a contextualized pair representation $\bar{k} = [\bar{q} \oplus (\bar{q} - \bar{a}) \oplus (\bar{q} \times \bar{a}) \oplus \bar{a}]$ (assuming $|\bar{q}| = |\bar{a}|$). Here, $\oplus$ denotes concatenation. The anchor a can be any of the attach or merge candidates (a_A/a_M). A three-way classifier is learned to produce the categorical probability distribution over the training samples for *Merge (M)*, *Attach (A)* and *No-operation (N)* — three classes ($|\mathcal{Z}| = 3$) of operations. If $\theta \in \mathbb{R}^{|\bar{k}|\times|\bar{\mathcal{Z}}|}$ be a learnable projection matrix that projects the contextualized pair embedding $\bar{k}$ to the label space $\mathcal{Z} \in \mathbb{R}^3$. The predictions are obtained as below,

$$\hat{Y} = \text{softmax}(\text{MLP}(\bar{k}; \theta)) \tag{6}$$

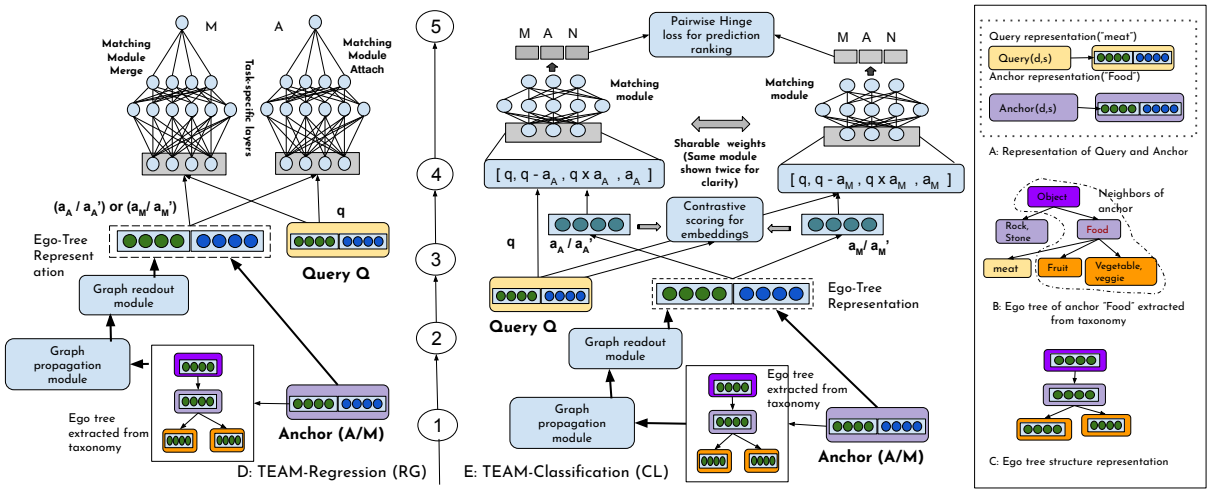


Figure 4: Taxonomy Expansion framework with Attach and Merge (TEAM) **D: TEAM-Regression-RG** — •1. (Query Q, Anchor, (N+1) Negative Anchors) are fed to the model, •2. Representation learning via shared graph propagation and readout modules, •3. Projection to shared hidden layers, •4. Task-specific matching modules with non-shareable weights, •5. Task-specific regression outputs. **E: TEAM-Classification (CL)** — •1. (Query Q, Anchor-A, Anchor-M, (N+1) Negative Anchors) are fed to the model, •2. Representation learning via shared graph propagation and readout modules, •3. Projection to shared hidden layers, •4. Simultaneous optimization of classification and ranking losses. •5.Three-way merge, attach, no-operation (M, A, N) prediction.
Explanation of used color-codes. Light-Purple: Anchor (A/M) nodes, Orange: Children of the anchor, Purple: Parent of the anchor. Green: Definition representation of anchors, Blue: Synset representation of anchors, Yellow: Query representation.

For two versions of TEAM, we chose two different kinds of matching models based on empirical performances to capture different kinds of embedding interaction in the latent space.

3.3.1 Multi-task Learning

Classification. Unlike in TEAM-RG, where we posit taxonomy expansion as a regression task with implicit ranking viz. discriminating true and false examples via InfoNCE loss, *in TEAM-CL, we simultaneously optimize for classification and explicit ranking objectives.* We obtain classification predictions from the matching module as described before. Given a training set $\mathcal{X}$, and a set of classes $\mathcal{Z}$ (M: Merge, A: Attach, N: No-operation), we optimize for the self-supervised cross-entropy loss over the task predictions $\hat{Y}$ given the ground-truth task-classes Y for an input query-anchor pair.

$$\mathcal{L}_C = -\frac{1}{|\mathcal{X}|}\sum_{i\in\mathcal{X}}\sum_{z\in\mathcal{Z}} Y_{iz} ln \hat{Y}_{iz} \tag{7}$$

Ranking. The classification objective can only learn and infer the confidence score of an operation (M/ A/ N) for a training sample. It fails to give us a reliable ranked list of prospective anchors-(A/M) given a query — since it does not learn the relative ranks of positive and negative anchors for a query. As illustrated in Figure 4, for a query q, (i) the ego-tree of anchor-A comprises of that query's parent's hierarchical neighborhood, and (ii) the ego-tree of anchor-M comprises of that query's replica's (same/similar definition with a missing portion of synset) hierarchical neighborhood. *Since a query q is very similar to both of its anchor-A and anchor-M's ego-trees – these operations are hardly distinguishable.* Thus, a model must accommodate a provision for directly comparing the prediction scores of M and A operations and learning a margin of separation between the scores. Here, we introduce two ranking objectives in the framework — (i) a contrastive objective to compare and contrast among a positive anchor and $\mathcal{N}$ negative anchors, (ii) a pair-wise hinge loss to learn a maximum margin between the M and A prediction scores.

Let, $dist(.)$ be a function to measure the distance between a query $\bar{q}$ and its true/ false anchor-(A/M) representations $(\bar{a_A}, \bar{a_A}')$, $(\bar{a_M}, \bar{a_M}')$. We use "slash" (/) to denote either. We intend to rank a positive query-anchor pair $(q, a_A/a_M)$ higher than $\mathcal{N}$ no of negative pairs $(q, a'_A/a'_M)$ by enforcing a group-wise contrastive loss using a margin λ as,

$$\mathcal{L}_{R1_A} = \frac{1}{\mathcal{X}}\sum_{i\in\mathcal{X}}\frac{1}{|\mathcal{N}(\bar{q}_i)|}\sum_{\bar{a_A}'_i\in\mathcal{N}(\bar{q}_i)} \max(0, \lambda - m + m')$$
$$m = -\text{ dist}(\bar{q}_i - \bar{a_A}_i),\ m' = -\text{ dist}(\bar{q}_i - \bar{a_A}'_i) \tag{8}$$

We can similarly compute the margin-based group-wise contrastive loss $\mathcal{L}_{R1_M}$ for the Merge (M).

Now, to distinguish between M and A operations, let, $f(\bar{k})$ be a function that projects the contextualized (q, a) embedding $\bar{k}$ in Equation 6, to

a hidden space $\mathbb{R}^h$. Here we introduce a margin-based hinge-loss on sample anchor pairs attach-merge $\langle a_A, a_M \rangle$ for a given query q via their contextualized vectors $\langle \bar{k_A}, \bar{k_M} \rangle$. If class labels of merge and attach are $M = 2, A = 1$, we ensure the prediction scores $\hat{Y}_{A/M} = f(\bar{k}_{A/M})$ for M and A are separated by a margin of λ.

$$\mathcal{L}_{R2} = \sum_{Y(\bar{k_A}) > Y(\bar{k_M})} \max(0, \lambda - f(\bar{k_M}) + f(\bar{k_A}))$$

Therefore, the final loss is, $\mathcal{L} = \mathcal{L}_C + \mathcal{L}_{R1_A} + \mathcal{L}_{R1_M} + \mathcal{L}_{R2}$ — considering both margin-based group-wise contrastive loss and pairwise hinge loss comprising the overall ranking loss.

3.4 Model Inference

We follow Taxo-Expan's (Shen et al., 2020) evaluation strategy for inferring the best candidate anchor a given a query q. We use our classification objective to decide which operation among merge, attach, or no-operation (M, A, N) to perform when q is given. *i)* For TEAM-RG, we augment a classification layer on top of the task-specific regression layers. Given a query q and a set of candidate anchors a, we obtain the merge and attach regression scores and choose the best value along with the corresponding operation as the apt operation to perform. *ii)* For TEAM-CL choosing which operation to perform is obtained based on the three-way prediction scores, given $< q, a, (0/1) >$ as input. Since both of our proposed frameworks optimize for ranking loss, i.e., discriminates true candidate pairs from the negative ones — we get a ranked list of candidate anchors a while matching each of them with q via respective matching modules.

4 Experiments and Results

Here we give you an overview of our experiment settings and provide the detailed reproducibility information in the Sections C, D, E of the Appendix.

Datasets. Table 1 shows the basic statistics of three WordNet taxonomies used in this study. The taxonomy networks are extracted from Assamese, Bengali, and Hindi WordNets, respectively.

	Nodes	Edges	Max-in degree	Max-out degree	leaf nodes
Assamese WordNet	8466	8363	1	525	7072
Bengali WordNet	26007	25815	1	924	22847
Hindi Wordnet	28242	28016	1	951	24737

Table 1: Dataset Statistics

Metrics. We use Mean Rank (MR), Hit@k, and Mean Reciprocal Rank (MRR) to evaluate the ranks of the retrieved results obtained from different models, for the test queries. Like Taxo-Expan (Shen et al., 2020) evaluation strategy, we scale the MRR score by a factor of 10 to highlight the discrepancy of the performances among different methods. Further, we use Accuracy, Micro/ Macro F1, Precision, Recall, and F-Scores to evaluate a method's prediction capability to decide which operation among merge (M), attach (A), and no-operation (N) needs to be performed.

Baselines. We choose two most recent benchmark SOTA taxonomy-expansion frameworks TaxoExpan (Shen et al., 2020) and Triplet Matching Network(TMN) (Zhang et al., 2021) as the competing methods. As Taxo-Expan and TMN outperform SemEval-2016 (Shen et al., 2020; Zhang et al., 2021), we have not included SemEval-2016 as baseline in this study. In terms of learning objective, Taxo-Expan is similar to ours. It uses ego-tree-based anchor features for matching query features in a regression-based setting. TMN captures fine-grained relationship dynamics of query and anchor concepts using channel-wise gating mechanism-based attention learning.

All datasets and our model implementations are available at: https://github.com/barnal/TEAM

5 Results

Here we report the classification and ranking results of the competing methods. We also compare and contrast among the variants of our TEAM framework. Apart from the two versions of the TEAM, namely, TEAM-RG and TEAM-CL, we have task-specific model variants specified as — attach–A, merge–M and merge+attach–MA. Here, (attach+merge) means simultaneously optimizing for both the tasks.

5.1 Ranking Results

In Table [2], we show the performance of the competing methods in terms of (best) ranking scores. We see similar trends for all taxonomies in the sub-tables. When considering only attach operation and the test ranking scores, we see TEAM-RG clearly beats Taxo-Expan by a large margin of $(196.87, 487.75, 470.87)$ in MR, by a margin of

	Assamese WordNet-Noun				Bengali WordNet-Noun				Hindi WordNet-Noun			
Methods	Micro_MR	Hit@1	Hit@3	MRR	Micro_MR	Hit@1	Hit@3	MRR	Micro_MR	Hit@1	Hit@3	MRR
TEAM-RG(A)	**144.92**	0.27	**0.42**	**0.67**	**191.51**	**0.17**	**0.36**	**0.86**	**177.85**	0.28	**0.43**	**0.67**
TEAM-CL(A)	189.75	0.16	0.28	0.57	277.01	0.05	0.18	0.50	220.98	0.13	0.29	0.54
Taxo-Expan(A)	341.81	0.07	0.11	0.29	679.26	0.03	0.04	0.10	648.72	0.04	0.08	0.14
TMN(A)	203.28	**0.28**	0.41	0.63	319.36	0.10	0.15	0.69	246.45	**0.31**	0.25	0.61
TEAM-RG(M)	**1.27**	**0.95**	**0.98**	**1.00**	**2.04**	**0.92**	**0.98**	**1.00**	**5.38**	**0.83**	**0.88**	**0.95**
TEAM-CL(M)	6.44	0.71	0.82	0.93	11.06	0.61	0.75	0.89	9.80	0.64	0.71	0.91
TEAM-RG(MA)	**73.34**	**0.61**	**0.70**	**0.83**	**59.81**	**0.69**	**0.80**	**0.85**	**91.62**	**0.63**	**0.71**	**0.81**
TEAM-CL(MA)	99.00	0.37	0.50	0.74	144.38	0.33	0.46	0.70	113.39	0.32	0.47	0.73

Table 2: Ranking results for test queries

$(0.2, 0.14, 0.24)$ in Hit@1 and $(0.31, 0.32, 0.37)$ in Hit@3 for Assamese, Bengali and Hindi WordNet respectively. We see TEAM-CL though performing competitively but is outperformed by TEAM-RG by a margin of $(44.82, 86.01, 43.04)$ in MR, by a margin of $(0.11, 0.12, 0.28)$ in Hit@1 and $(0.14, 0.18, 0.43)$ in Hit@3 respectively for Assamese, Bengali and Hindi WordNet. In TEAM-RG(M), we obtain near-perfect MRR scores. This is because the definitions are already present in training set for the query concepts with known definitions (test sample drawn from the base taxonomy). The score of 1 indicates the ability of the proposed method TEAM-RG(M) to correctly identify the appropriate anchor nodes for the merge operation. TMN gives better performance than Taxo-Expan owing to its useful attention mechanism. But, Team-RG(A) outperforms TMN in all the metrics except Hits@1. We only compare Taxo-Expan results for the attach since it is originally proposed for the attach operation. In the (merge-M) and (merge+attach-MA) section of the tables also, we see that TEAM-RG outperforms TEAM-CL on all three WordNet taxonomies. We attribute this huge performance improvement of TEAM-RG to InfoNCE based training — as it simultaneously provides pseudo-supervision from the negative examples while optimizing for the task-specific regression layers.

5.2 Classification Results

In Table [3], We observe similar trends on all three WordNet taxonomies. Since Taxo-Expan is a regression-based algorithm proposed for only attach operation in taxonomy expansion task — we could not obtain its classification performance. Therefore, we only consider variants of our frameworks as competing methods. As described in the sub-section 3.4, using a classification layer on top of the regression layer in TEAM-RG, we obtain classification performances for the attach, merge operations along with both (attach+merge) operations. Whereas obtaining classification performance for TEAM-CL is straightforward since this is already a classification framework.

When comparing TEAM-RG (attach) and (merge) variants — we see, unlike ranking results where ranking results of merge operation were always better than the attach operation, here the classification results of merge operation are inferior to attach operation. It means that the RG variant learns better ranking as compared to CL variants, but they fail to distinguish M and A – the operation to perform. This is expected since we do not provide a scheme here to contrast M and A operations — which is the motivation for our CL variant framework.

When comparing TEAM-RG and TEAM-CL for (merge+attach), we see TEAM-CL gives better classification scores using test queries except for Macro-F1 scores. TEAM-RG gives the best performance for the Macro-F1 score for the test cases. This essentially means that class-wise prediction performances are inferior for TEAM-CL. This is expected behavior since, in each batch of the training sample, we include a substantially large number ($\mathcal{N}$) of negative examples with class-label (N–No operation). We design our training samples like this so that the contrastive loss is better approximated. Nevertheless, it leads to a class-imbalance issue in our three-way classification setup, i.e., a large number of samples with N class labels as compared to the other M/ A class labels. Thus, TEAM-CL biases its prediction towards the N class, leading to poorer Macro-F1 scores than TEAM-RG.

To summarize, we observe that TEAM-RG gives the best ranking performances, whereas TEAM-CL gives the best classification performances. TEAM-

	Assamese WordNet						Bengali WordNet						Hindi WordNet					
Methods	Acc	Mi-F1	Ma-F1	Prec.	Recl	F-Sc	Acc	Mi-F1	Ma-F1	Prec.	Rec.l	F-Sc	Acc.	Mi-F1	Ma-F1	Prec.	Recl	F-Sc
TEAM-RG(A)	0.97	0.97	0.49	0.95	0.97	0.96	0.98	0.98	**0.50**	0.97	0.98	0.97	0.90	0.90	0.47	0.81	0.90	0.85
TEAM-RG(M)	0.81	0.81	0.45	0.66	0.82	0.73	0.29	0.29	0.29	0.57	0.29	0.48	0.55	0.55	0.30	0.23	0.15	0.39
TEAM-RG(MA)	0.88	0.88	0.88	0.89	0.88	0.88	0.51	0.51	0.36	0.96	0.71	0.85	0.53	0.53	0.43	0.82	0.53	0.62
TEAM-CL(MA)	**1.00**	**1.00**	**0.50**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.48	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.50**	**1.00**	**1.00**	**1.00**

Table 3: Classification results for test queries

CL performs poorly in Macro-F1 since it presumably suffers from class-imbalance issues owing to the style of training sample generation. Frameworks with multi-task learning strategy (TEAM-RG and TEAM-CL) outperform frameworks (Taxo-Expan) designed to perform a single task — which is motivated by the fact that simultaneously optimizing for multiple tasks provides self-supervision to each other, resulting in better performances.

5.3 Expansion of Assamese WordNet Taxonomy with Out-Of-Vocabulary (OOV) words

Assamese WordNet-Noun				
Methods	Micro_MR	Hit@1	Hit@3	MRR
TEAM-RG(A)	**65.30**	**0.33**	**0.53**	**0.80**
TEAM-CL(A)	240.47	0.02	0.16	0.06
Taxo_Expan(A)	386.30	0.05	0.05	0.11
TEAM-RG(M)	**170.79**	**0.07**	**0.14**	**0.38**
TEAM-CL(M)	331.93	0.03	0.12	0.29

Table 4: Ranking result for out-of-vocabulary words

To investigate the effectiveness of the proposed models, we employ the models for expanding a WordNet with OOV words. For this, first, we find out-of-vocabulary words, i.e., words that are not present in Assamese WordNet. Second, we manually identify true anchors of respective out-of-vocabulary words with associated operations (Attach/Merge) in Assamese WordNet. We evaluate the predicted results of the proposed model against the manually identified true anchors. Since we can either perform an A or M operation with OOV words and not both, we do not predict expansion tasks for OOV words using any of the MA variants of our proposed frameworks. Table 4 shows the ranking performance of the model in predicting true anchors for attach and merge expansion operations. We see a similar trend of prediction ranking as seen with the test set in our earlier experiments. TEAM-RG gives the best performance in both expansion operations. The detailed analysis of results is in Section F of the Appendix.

6 Related Works

Existing methods for taxonomy expansion can be divided into two categories: relying on alignment between multiple taxonomies [Ruiz-Casado et al. (2005), Toral et al. (2008), Ponzetto and Navigli (2009), and Yamada et al. (2011)] or relying on machine learning-based rating sub-graphs. Further, the latter category can be divided into two sub-categories (1) by expanding synonymy relations/Merge (2) by expanding hypernymy relations/Attach. Synonymy-based taxonomy expansion leverages synonymy relations of the taxonomy. Given a seed taxonomy, the distributional approach discovers synonyms by representing strings with their distributional feature and learning a classifier to predict the relation between strings [(Nakashole et al., 2012), (Wang et al., 2019), (Fei et al., 2019)]. Most of the recent taxonomy expansion approaches are based on hypernymy expansion. These methods attempt to determine the attachment position by scoring between several nodes. Recently numerous methods have been proposed to solve this problem(Shen et al., 2018), (Shen et al., 2020), (Yu et al., 2020b), (Zhang et al., 2021), (Liu et al., 2021). (Shen et al., 2020). Hence, all the existing taxonomy expansion approaches expand a taxonomy either by merge operation(synonymy expansion) or by attach operation(hypernymy expansion). However, particular to WordNet expansion it is an integrated task of Merge and attach operation. We are the first to study the problem of taxonomy expansion using both the Attach and Merge taxonomy expansion operations in a single model. A detailed related study is in Section A of the Appendix.

7 Conclusion

In this paper, we proposed an integrated framework called *Taxonomy Expansion with Attach and Merge (TEAM)* for expanding taxonomy with attach and merge operations together. We built two multi-task learning-based variants of TEAM, namely, TEAM-Regression and TEAM-Classification, which solve the taxonomy expansion problem as regression and

classification, respectively. Our proposed methods learned to predict the taxonomy expansion operation (merge, attach, or no-operation) to perform and provided a ranked list of candidates. We evaluated the effectiveness of TEAM on WordNet taxonomies of three distinct languages, viz., Assamese, Bangla, and Hindi. In various experimental setups, the proposed TEAM-RG and TEAM-CL outperformed its state of the art for attach operation and provided a highly encouraging performance on merge operation. We had also investigated the performance of the proposed model with out-of-vocabulary concepts.

In the future, we plan to investigate the response of the proposed model with different types of taxonomies and WordNet of different languages. Another future research possibility can be to explore the response of this model using advanced contextual encoders.

References

James Allan, Ron Papka, and Victor Lavrenko. 1998. On-line new event detection and tracking. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–45. ACM.

Pushpak Bhattacharyya. 2010. Indowordnet. In *In Proc. of LREC-10*. Citeseer.

Adrian Boteanu, Adam Kiezun, and Shay Artzi. 2018. Synonym expansion for large shopping taxonomies. In *Automated Knowledge Base Construction (AKBC)*.

Hongliang Fei, Shulong Tan, and Ping Li. 2019. Hierarchical multi-task word embedding learning for synonym prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 834–842.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

David Jurgens and Mohammad Taher Pilehvar. 2015. Reserating the awesometastic: An automatic extension of the wordnet taxonomy for novel terms. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1459–1465.

David Jurgens and Mohammad Taher Pilehvar. 2016. Semeval-2016 task 14: Semantic taxonomy enrichment. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 1092–1102.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Zichen Liu, Hongyuan Xu, Yanlong Wen, Ning Jiang, HaiYing Wu, and Xiaojie Yuan. 2021. Temp: Taxonomy expansion with dynamic margin loss through taxonomy-paths. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3854–3863.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Distinguishing antonyms and synonyms in a pattern-based neural network. *arXiv preprint arXiv:1701.02962*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Simone Paolo Ponzetto and Roberto Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *IJCAI*, volume 9, pages 2083–2088.

Michael Poprat, Elena Beisswanger, and Udo Hahn. 2008. Building a biowordnet using wordnet data structures and wordnet's software infrastructure–a failure story. In *Software engineering, testing, and quality assurance for natural language processing*, pages 31–39.

Meng Qu, Xiang Ren, and Jiawei Han. 2017. Automatic synonym discovery with knowledge bases. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 997–1005.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. 2005. Automatic assignment of wikipedia encyclopedic entries to wordnet synsets. In *International Atlantic Web Intelligence Conference*, pages 380–386. Springer.

Michael Schlichtkrull and Héctor Martínez Alonso. 2016. Msejrku at semeval-2016 task 14: Taxonomy enrichment by evidence ranking. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 1337–1341.

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 *Proceedings of The Web Conference 2020*, pages 486–497.

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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2180–2189.

Amit Singhal et al. 2001. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808, Sydney, Australia. Association for Computational Linguistics.

Kunihiro Takeoka, Kosuke Akimoto, and Masafumi Oyamada. 2021. Low-resource taxonomy enrichment with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2747–2758.

Antonio Toral, Rafael Muñoz, and Monica Monachini. 2008. Named entity WordNet. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Nikhita Vedula, Patrick K Nicholson, Deepak Ajwani, Sourav Dutta, Alessandra Sala, and Srinivasan Parthasarathy. 2018. Enriching taxonomies with functional domain knowledge. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 745–754.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Jin Wang, Chunbin Lin, Mingda Li, and Carlo Zaniolo. 2019. An efficient sliding window approach for approximate entity extraction with synonyms. In *EDBT*, pages 109–120.

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 *Proceedings of the Web Conference 2021*, pages 3291–3304.

Ichiro Yamada, Jong-Hoon Oh, Chikara Hashimoto, Kentaro Torisawa, Jun'ichi Kazama, Stijn De Saeger, and Takuya Kawada. 2011. Extending WordNet with hypernyms and siblings acquired from Wikipedia. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 874–882, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Jiale Yu, Yongliang Shen, Xinyin Ma, Chenghao Jia, Chen Chen, and Weiming Lu. 2020a. Synet: Synonym expansion using transitivity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1961–1970.

Yue Yu, Yinghao Li, Jiaming Shen, Hao Feng, Jimeng Sun, and Chao Zhang. 2020b. Steam: Self-supervised taxonomy expansion with mini-paths. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1026–1035.

Jieyu Zhang, Xiangchen Song, Ying Zeng, Jiaming Shen, Yuning Mao, Lei Li, et al. 2021. Taxonomy completion via triplet matching network. *arXiv preprint arXiv:2101.01896*.

A Related Works

Expansion by resource alignment: In the first category of studies, Poprat et al. (2008) first attempted to automatically expand a WordNet with biomedical terminology; however, they were unable in developing the resource. Ruiz-Casado et al. (2005), Toral et al. (2008), Ponzetto and Navigli (2009), and Yamada et al. (2011) exploit structured information in Wikipedia to expand WordNet with new synsets. Snow et al. (2006) leverage distributional similarity techniques for WordNet expansion. Jurgens and Pilehvar (2015) enrich the existing WordNet taxonomy using an additional resource, Wiktionary, to extract sense data based on information in the term concepts.

Synonymy Expansion: Synonymy expansion in a taxonomy leverages synonymy relations to enrich a taxonomy with new concepts. Approaches for synonymy expansion can be divided in to two categories: (1) Distributional based approach (Wang et al., 2019), (Fei et al., 2019) (2) Pattern-based approach (Nguyen et al., 2017), (Nakashole et al., 2012). Given a seed taxonomy, the distributional approach discovers synonyms by representing strings with their distributional feature and learning a classifier to predict the relation between strings. However, in the pattern-based approach, consider the sentences mentioning a pair of synonymous strings and learn some textual patterns from these sentences, which are further used to discover more synonyms. Qu et al. (2017) proposed an approach that integrates both the categories. Boteanu et al. (2018) focus on the problem of expanding taxonomies with synonyms for applications in which entities are complex concepts arranged into taxonomies designed to facilitate browsing the product catalog on amazon.com. They first generate synonymy candidates for each node in the taxonomy and then filter synonymy candidates using a binary classifier. Yu et al. (2020a) study a task of synonym expansion using transitivity named SYNET, which leverages both the contexts of two synonymy pairs.

Hypernymy expansion : Jurgens and Pilehvar (2016) formulated a task of synonymy expansion, where it is proposed to enrich the WordNet taxonomy by performing two operations for each new concept. The first action is *Attach*, where a new concept is treated as a new synset and is attached as a hyponym of one existing synset in WordNet, and the second action is *Merge*, where a new concept is merged into an existing synset. The best solution proposed by Schlichtkrull and Alonso (2016) included only the attach operation. Later solutions for attaching, as in Shen et al. (2020), adopted self-supervision and tried to exploit the information of nodes in the seed taxonomy to perform node pair matching. On the other hand, Yu et al. (2020b) resorted to classification along mini paths in the taxonomy. In contrast, in our current approach, we have incorporated both the attach and merge operations.

B Graph Propagation Module (in details)

Graph Neural Network (GNN) allows us to transform and propagate node features as messages to learn structure-aware node representations. The EXTRACT() mechanism extracts the messages from a target node and its neighborhood, which is later on combined based on a chosen ATTENTION() mechanism by the AGGREGATE() operation. Next, the aggregated message is propagated to the rest of the graph. Studies apply various aggregation strategies to combine the propagated and extracted messages from the target node's neighborhood based on the importance of each node in the neighborhood towards that target node. GCN (Kipf and Welling, 2016) and GAT (Veličković et al., 2017) are popular GNN frameworks.

GCN uses $N(*)$ neighborhood-based normalization constant to calculate the importance ($att_{v\rightarrow u}$) of node v towards the target node u without considering the participating nodes' features as follows.

$$H_u^l = \sigma\Big(\sum_{\forall v \in \tilde{N(u)}} att_{v\rightarrow u}^{l-1} W^{l-1} H_v^{l-1}\Big) \quad (9)$$

$$\begin{aligned} \text{ATTENTION}_{\text{GCN}}(v \rightarrow u) &: \\ att_{v\rightarrow u}^{l-1} &= \frac{1}{\sqrt{|\tilde{N(u)}||\tilde{N(v)}|}} \\ \text{EXTRACT}_{\text{GCN}}(v) &: W^{l-1} H_v^{l-1} \\ \text{AGGREGATE}_{\text{GCN}}(*) &: \sigma(*) \end{aligned}$$

where σ is non-linear activation function, W^l is a projection matrix for a GNN layer l and $\tilde{N(*)}$ is a node's extended neighborhood structure including the node itself (i.e., including self-loop edges).

GAT uses the same message extraction and aggregation strategies as above except for the fact that it uses attentive aggregation strategies that consider both the participating nodes' features as well as the

neighborhood information, as follows.

$$\text{ATTENTION}_{\text{GAT}}(v \rightarrow u) : att^{l-1}_{v \rightarrow u} = \underset{\forall v \in N(u)}{\text{SOFTMAX}}\Big(c^{l-1}(W^{l-1}H^{l-1}_{u} \oplus W^{l-1}H^{l-1}_{v})\Big)$$

where c^{l-1} is a learnable parameter to approximate the importance of node v towards u ($att^{l-1}_{v \rightarrow u}$) based on their interaction in the latent space in l layer-wise manner, W is the layer-wise projection matrix, and $\oplus$ denotes concatenation.

C Dataset Statistics

Table 1 shows the basic statistics of the datasets that we consider in this study. The taxonomy networks are extracted from Assamese, Bengali, and Hindi WordNets, respectively. The Assamese WordNet dataset consists of 8466 of noun concepts and 8463 edges. The network has the maximum in-degree path of 1, implying that each concept node has only one parent node or predecessor. The network consists of 7072 leaf nodes. Bengali WordNet dataset contains 26007 of noun concepts and 25815 edges and has the maximum in-degree path of 1. The network consists of 7072 leaf nodes. Hindi WordNet dataset consists of 28242 of noun concepts and 28016 edges. It contains 24737 leaf nodes.

D Ranking and Classification Metrics

We use an array of performance metrics from the domain of classification and ranking to evaluate the competing methods' performances. Among the ranking metrics, we use Mean Rank (MR), Hit@k (k=1, 3), and Mean Reciprocal Rank (MRR) to judge how well a competing method performs in producing a ranked list of candidate anchors given a test query and a taxonomy expansion operation to carry-out – merge or attach.

- **Mean Rank:** It calculates the average rank of true anchors among all the candidate anchors with respect to the matching scores, given a query.
- **Hit@k:** It calculates the number of times a true anchor appears in the top k positions when matched with a test query.
- **Mean Reciprocal Rank(MRR):** The Mean Reciprocal Rank is used to assess the ranking quality of the true anchor. The reciprocal rank can be computed by finding and inversing the rank of a true anchor in the predicted anchors' list of each query. MRR is averaged over all queries.

Further, we use Accuracy, Micro/ Macro F1, Precision, Recall, and F-Scores as classification metrics for deciding given a test query and an initial taxonomy tree, which operation among merge (M), attach (A), and no-operation (N) is to be performed.

- **Accuracy:** It summarizes the performance of the classification model as the fraction of the number of true tasks predicted over the total number of ground-truth tasks for a set of queries.
- **Precision:** It calculates the fraction of true-positive predicted expansion task classes among the total number of true-positive and false-negative task classes.
- **Recall:** It calculates the fraction of true-positive predicted expansion task classes among all the relevant ground-truth task classes.
- **F-Score:** The harmonic mean of precision and recall. It is also known as F1-Score.
- **Micro/ Macro F1 :** The Macro F1 computes F1-Score for each class (merge M/ attach A) independently but averages the final score by treating each expansion task-class as equally contributing. However, Micro F1 computes the F1-Score for each query sample in the training set and therefore aggregates the contributions of all expansion task classes to compute the final average metric.

E Evaluation Strategy

We obtain the initial feature vector for train and test concepts using pre-trained subword-aware Fasttext embeddings. For each concept, we generate its definition embedding by averaging the embedding of each word in its textual definition. We employ PyTorch and DGL framework [3] to load and train embeddings. In TEAM, we use a two-layer position-enhanced GAT where the first layer (of size 300) has four attention heads and the second layer (of size 600) has one attention head. We use 50-dimension position embeddings for both layers and apply dropout with the rate of 0.1 on the input

[3] https://github.com/dmlc/dgl

feature vectors. We use Adam optimizer with an initial learning rate of 0.001.

F Ranking result for out-of-vocabulary words

In the case of attach expansion, TEAM-RG beats state-of-the-art Taxo-expan by a large margin of $(321, 0.31, 0.48, 0.69)$ in MR, Hit@1, Hit@3, MRR, respectively. However, our proposed frameworks are seen not to perform so well in merge M operation as compared to the attach A operation. Intuitively, this is because, for OOV words, we use a set of manually collected paraphrase definitions of the OOV words to match them with the candidate anchor concepts in the existing taxonomy. Whereas for actually training our model, we have used the same definitions in the replica nodes. That is, we have used the same definition in the original anchor concept and in the input query-concept with mutually exclusive synset information. Thus, in this case-study, the paraphrase-based definition matching deems challenging for our learning model resulting in poorer results for M operation. We believe we can always eliminate this drawback by using a description generation tool (Wang et al., 2021) to generate different definitions of the same concept nodes and train our learning model in a more powerful way.