Taxonomy Enrichment without candidates

Timotei Ardelean, Andreea Dogaru, Gabriel Rozzonelli, Alsu Vakhitova
Skolkovo Institute of Science and Technology, Moscow, Russia

Abstract

We present a new approach in the area of Taxonomy Enrichment, which leverages highly-available pre-trained resources to extract information about recent and domain-specific terms and integrates them into existing taxonomic structures. A novel formulation is proposed in order to facilitate the use of such automated systems in real-life scenarios. Specifically, the task is to predict new lemmas, without a set of possible candidates, given only their relationships with neighboring nodes (hypernyms and hyponyms of the first and second order).

Related resources

- Video: YouTube
- Presentation: Google Drive
- Notebook: Google Colab
- Code: GitHub Repository

1 Introduction and motivation

The existence of structured lexical resources is of prime importance for the NLP community. Indeed, lexicon datasets like WordNet (Miller, 1998) and all its other declensions are used for a various set of NLP tasks, due to them being extensive and well-structured. However, these static lexicons are often rather old and may lack adaptiveness related to the inherent dynamic nature of languages, missing some of today's common terms.

Hence, updating those already existing resources seems to be of a certain interest, as well as an obvious cheaper alternative to creating new lexicons from scratch. However, feeding those lexical databases manually is rather expensive in terms of both time and human resources. Thus, developing systems in order to automatize this process appears to be more than relevant.

In parallel, the last breakthroughs in the area of language models allow to address the task of masked token prediction with confidence. Models such as BERT (Devlin et al., 2018) and its multiple variants have proven to perform well at this very task. With this consideration in mind, one can wonder if the predictive capabilities of such models can serve the purpose of finding new entries to be added to existing lexical resources.

In this report, we present our work on taxonomy enrichment without candidates. After reviewing some of the existing methods for the taxonomy enrichment task in general, we present a novel approach based on knowledge-enabled language representation models and graph attention networks to infer both new lemmas and their positions in the hierarchies of said structured datasets. Following descriptions of our experiments, we present our results and discuss the limitations of such a method.

2 Related work

Automatizing the manual process of creating taxonomies is a high-interest area for Natural Language Processing researchers. The proposed methods can be divided into two main groups. The first one, Taxonomy Construction (Induction), generates new taxonomies without relying on existing ones. It consists of two predominant steps:

1. Extract term relations from text corpora.
2. Organize the derived relations into a hierarchical structure.

Two distinct approaches can be employed for the first stage:

- Pattern-based methods (Agichtein and Gravano, 2000; Hearst, 1992; Jiang et al., 2017; Nakashole et al., 2012; Panchenko et al., 2016; Roller et al., 2018) that leverage pre-defined

lexico-syntactic patterns (*e.g.* "is-a" relations) to detect hypernymy-hyponymy pairs.

- Distributional methods that use clustering algorithms (Alfarone and Davis, 2015; Shang et al., 2020; Zhang et al., 2018) to identify topic-indicative terms based on the distributional hypothesis (Harris, 1954), or through the binary classification of the hypernymy relation using term embeddings (Baroni et al., 2012; Chang et al., 2018; Fu et al., 2014; Luu et al., 2016; Roller et al., 2014).

The second stage creates a graph with the identified concepts (nodes, relations) and processes it using specific graph optimization algorithms to obtain the hierarchic taxonomy structure (Bansal et al., 2014; Cocos et al., 2018; Gupta et al., 2017; Navigli et al., 2011; Velardi et al., 2013).

Our work is part of the second type of approach for automatizing the creation of taxonomies, Taxonomy Enrichment (Expansion). This task receives less attention compared to the previous one. The methods usually start from a base taxonomy and extend it by integrating a set of candidate words in the best positions. Each new term is merged to an existing synset if it conveys the same concept. Otherwise, if the concept is more specific, a synset node is created and attached to an existing one in a hypernymy-hyponymy relation.

Apart from the base taxonomy and candidate terms, current methods also use additional input data to find the best fit for the orphan words, which is difficult to obtain in real-life scenarios. The dataset created for SemEval-2016 Task 14 (Jurgens and Pilehvar, 2016) provides term definitions, so the competing models are dependent on them (Anke et al., 2016; Tanev and Rotondi, 2016). Other works take advantage of query logs for search engines taxonomies (Wang et al., 2014), external paraphrase datasets (Plachouras et al., 2018), large corpora for generating Poincaré embeddings (Aly et al., 2019), or features extracted from Wiktionary and pre-trained word embeddings (Nikishina et al., 2020).

Similar to our work, TaxoExpan (Shen et al., 2020) also considers a subgraph of the taxonomy and graph-position enhanced neural networks. However, their approach is candidate-based, so it is not directly applicable for our candidate-free setup.

Taxonomies and knowledge graphs can both be represented as graph/tree structures, which means that Taxonomy Enrichment tasks could be solved using similar methods. In the past few years, different variants of Graph Neural Networks are being developed with Graph Convolutional Networks (GCN) (Kipf and Welling, 2016) being one of them. GCNs are considered as one of the basic Graph Neural Networks variants. GCNs perform similar to Convolutional Neural Networks (CNNs) operations where the model learns the features by inspecting neighboring nodes. The major difference between CNNs and GNNs is that CNNs are specially built to operate on regular (Euclidean) structured data, while GNNs are the generalized version of CNNs where the numbers of nodes connections vary and the nodes are unordered (irregular on non-Euclidean structured data). GCNs can be categorized into 2 major algorithms: Spatial Graph Convolutional Networks (Spurek et al., 2019) and Spectral Graph Convolutional Networks (Kipf and Welling, 2016).

Inspired by Vaswani et al. (2017) a further advancement of GNN emerged - Graph Attention Networks (GAT) (Velickovic et al., 2017). The GAT layer expands the basic aggregation function of the GCN layer, assigning different importance to each edge through the attention coefficients. By stacking such layers, GATs enable (implicitly) specifying different weights to different nodes in a neighborhood, without requiring any kind of costly matrix operation (such as inversion) or depending on knowing the graph structure upfront. In this way, Velickovic et al. (2017) address several key challenges of spectral-based graph neural networks simultaneously, and make their model readily applicable to inductive as well as transductive problems.

Furthermore, recent advancement of language representation (LR) models motivated other architectures that leverage LR power. One of the most popular language representation models is called BERT (which stands for Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018). Unlike other language representation models (Peters et al., 2018; Radford and Narasimhan, 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task specific

architecture modifications. Liu et al. (2019) noted that pre-trained language representation models, such as BERT, capture a general language representation from large-scale corpora, but lack domain-specific knowledge. They proposed a knowledge-enabled language representation model (K-BERT) with knowledge graphs (KGs), in which triples are injected into the sentences as domain knowledge. Moreover, in order to control the amount of knowledge incorporated into the sentence, they have introduced such modifications as soft-position and visible matrix. Another important combination of graphs and LR models is Graph-BERT (Zhang et al., 2020). Different from existing GNNs, Graph-BERT works well in deep architectures and does not suffer from the common problems with other GNNs. Based on a batch of linkless subgraphs sampled from the original graph data, Graph-BERT can effectively learn the representations of the target node with the extended graph-transformer layers.

3 Methodology

3.1 Dataset

The dataset used in our project was generated from the WordNet (Miller, 1998). Each entry consists of a subgraph, which includes a synset, its hypernyms, second-order hypernyms, co-hypernyms, hyponyms, second-order hyponyms and co-hyponyms. Moreover, for each synset a list of lemmas was provided. The aforementioned relations are illustrated in Figure 1. The central synset is masked and the models are tasked to predict candidates using the remained subgraph.

Furthermore, a subgraph is "flattened" into several sequences: token IDs, level IDs, synset IDs, lemma IDs and highway. Token IDs are produced by BERT tokenizer that is applied to each lemma in all synsets. The tokenization is performed on a subword level, thus some lemmas might be split into multiple tokens. Level IDs represent the position relative to a central node each synset has within a subgraph. Synset IDs indicate that a token belongs to a particular synset and are unique for each synset in a subgraph. Lemma IDs indicate that a token belongs to a particular lemma and are unique for each lemma in a subgraph. Highway is a boolean sequence that indicates which tokens belong to a synset name. For example, synset *education.n.05* consists of three lemmas: *education*, *training*, *breeding*. Tokens that result from tokenization of lemma *education* will be marked as a highway (True), while tokens for *training*, *breeding* won't be (False).

The dataset is split into three parts: train (63900 samples), validation (7099 samples) and test (3375 samples). In order to prevent data leakage it is ensured that the splits do not intersect.

3.2 Evaluation metrics

The generated candidates are compared against the true candidates from the existing taxonomy. We use standard metrics for Information Retrieval like Precision@k, Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR):

$$Precision@k = \frac{\text{relevant items @k}}{\text{recommended items @k}}, \quad (1)$$

where k is the number of candidates at a certain step;

$$AP = \frac{\sum_{k=1}^{n}(P(k) \times rel(k))}{\text{number of relevant documents}}$$

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q},$$

where $P(k)$ is the precision at cut-off k in the list, $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant document, zero otherwise, and Q is the number of queries;

$$MRR = \frac{1}{|Q|}\sum_{i}^{|Q|}\frac{1}{rank_i}, \quad (2)$$

where Q is the sample of queries, $rank_i$ is the first position of the relevant candidate in the ranked list for the query i.

3.3 Models

To solve the task of Taxonomy Enrichment we propose to rely on a pretrained masked language model, namely, BERT (Devlin et al., 2018). Since the model was trained on large corpora it has knowledge about many recent terms that we may want to add to an existent, incomplete, taxonomy. Specifically, we try to predict lemmas which fit a certain position in the taxonomic tree. The task we are aiming to solve is significantly harder compared to traditional candidate based approaches that often also use definitions and other contexts. Formally, we want to a learn a function $F(V(node))$, where $V(node)$ is the vicinity of a node (formed by known synsets) in the taxonomic tree, which

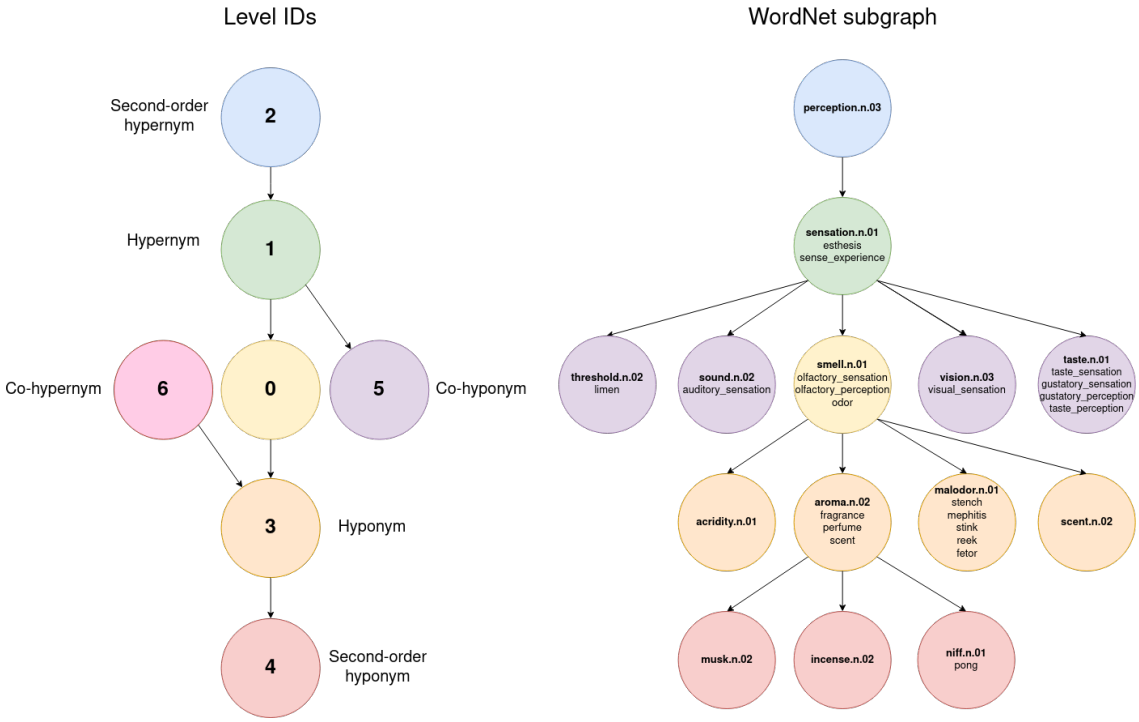

Figure 1: Dataset entry structure. The first graph presents the level IDs corresponding to each of the possible relations between the central node and the neighboring nodes. The second graph is a real example extracted from WordNet taxonomy.

outputs a ranking of possible lemmas which fit the query position. This formulation allows extending synsets in the taxonomic tree with new lemmas and also creating new synsets as leafs in the taxonomy. We represent this function using a neural network with learnable weights $F(V(node); \theta)$ and suggest different architectures trained with the objective of minimizing the classification loss. The experiments highlight two models: KBERT and KBERT-GAT which will be described in the following.

Since we propose a new formulation, there is no established baseline with which we can compare our models. Therefore, we adapt a solution based on a fixed vocabulary to the candidate-free taxonomy enrichment setting and use it as our baseline.

3.3.1 Baseline

In order to obtain generic information about words, the Baseline solution uses pretrained embeddings. There are many options available which will give different levels of performance. As the Baseline model predicts full lemmas directly, relying on a fixed vocabulary, it is possible that it doesn't contain some of the ground truth answers. Considering this, it is generally better to use a vocabulary as large as possible with informative embeddings (large dimensionality).

The Baseline uses an input similar with all other models, as described in section 3.1. However, instead of using the sub-word BERT tokenizer, a simple word splitting tokenization is employed. The corresponding embeddings are then processed by several Transformer Encoder layers (Vaswani et al., 2017). To ease the task of the network, the model doesn't output word probabilities directly. Instead, the pretrained embeddings are leveraged again and the model is tasked to predict features in this space, corresponding to the query position. To obtain lemmas, all words in the vocabulary are ranked based on their cosine similarity with the predicted vector. While this method can be applied to the proposed task, it has the inherent limitation of a fixed vocabulary and linear complexity scaling with its size.

3.3.2 KBERT

The KBERT solution relies on a BERT (Devlin et al., 2018) language representation model, and inspired by the work of Liu et al. (2019), we incorporate additional knowledge in the network input. We choose to use the lemmas of the synsets that are not part of the highway as "domain knowledge". Similar to the original formulation, we adapt the visible matrix to limit the impact of knowledge, preventing the knowledge noise issue. The visible

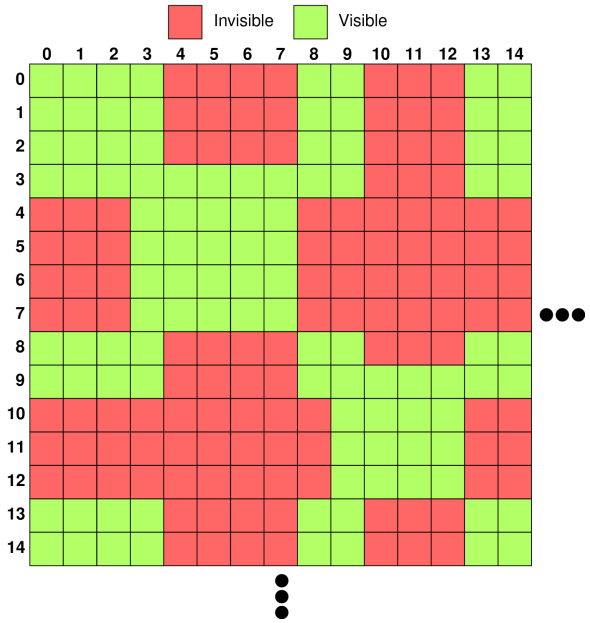

Figure 2: Visible matrix used by the KBERT model in the Multi-Head Attention layers. The matrix corresponds to the input data illustrated in figure 3. A visible entry means that the respective tokens can attend to one another, while an invisible entry prevents the attention between the respective tokens.

matrix is used in the Multi-Head Attention layer, limiting the visibility as follows: a token can attend to the tokens that are part of lemmas in the same synset, and if it is on the highway, it can also attend to the other tokens on the highway.

The model takes as input the data in the format described in section 3.1. The network first computes embeddings as illustrated in Figure 3. Token embeddings are taken from the pre-trained BERT model and freezed during training. We use the **BERT$_{\text{BASE}}$** configuration with 12 layers and hidden size of 768. The visible matrix specific to K-BERT can easily be built using the synset IDs and highway inputs and applied to the pre-trained model. An example of such matrix is illustrated in figure 2.

During training, we randomly choose one of the target lemmas of the central node and replace its tokens with the special `[MASK]` token. Then, we only compute the loss on the masked tokens. During inference, we iteratively predict lemmas with $1, 2, \ldots L$ `[MASK]` tokens and select the top k most probable outputs. For multiple masked tokens, a beam search approach is used to identify the best outputs.

3.3.3 KBERT-GAT

This architecture essentially extends the KBERT solution described in a previous section. The key difference is that the final prediction head is replaced with a graph-attention network (GAT) inspired by Velickovic et al. (2017).

The first steps are similar to ones in KBERT. The model takes the same input, builds visible matrix and computes input embeddings as illustrated in Figure 3.

Furthermore, the graph visible matrix is calculated. The assumed graph visibility principles are: every lemma sees other lemmas inside one synset, only highway lemmas see other highway lemmas, with following rules (for the reference of levels' order see Figure 1):

- Level 1 and level 0 are visible to each other;
- Level 1 and level 2 are visible to each other;
- Level 1 and level 5 are visible to each other;
- Level 3 and level 6 are visible to each other;
- Level 3 and level 4 are visible to each other;
- Level 3 and level 0 are visible to each other.

The graph visible matrix is used in the Multi-Head Graph Attention Layer. Basically, this matrix can be viewed as a rethought adjacency matrix.

As opposed to the original implementation of Velickovic et al. (2017) we had to perform a significant change in the architecture. Initially, GATs were not designed to accommodate batched data. Therefore, usage of our dataset has introduced a significant computational overhead to the multi-head attention on the final (prediction) layer of the network. To resolve this problem we have decided to replace the final prediction layer of GAT with a simple linear layer while carrying out the rest of the prediction process in the same way as for KBERT solution described in a previous section.

4 Results and discussion

We performed several experiments with the described models. This section describes the results of these experiments and compares the different approached.

For the baseline model we tested different different embeddings to observe how they affect the performance of the system. In Table 1 is shown a comparison between three such pretrained word embeddings.

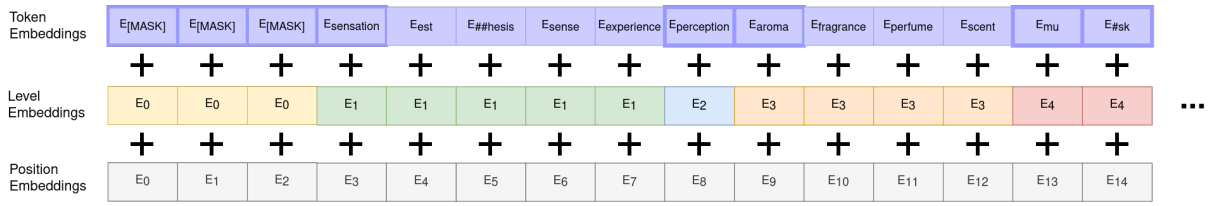

Figure 3: Input embeddings. Similar to the original BERT model, we apply sum between the individual embeddings.

Embedding	Vocab Size	Lemma Coverage	Precision@10	MRR	MAP
fasttext-wiki-300	999K	0.382	0.0003	0.00094	0.00095
glove-wiki-300	400K	0.338	0.0058	0.02587	0.02575
glove-twitter-200	1193K	0.235	0.0002	0.00169	0.00169

Table 1: Comparison between different pretrained word embeddings for the baseline model. Vocab size represents the number of tokens with trained vectors, Lemma Coverage equals the portion of ground truth lemmas that must be predicted which are contained in the vocabulary, Metrics were discussed in section 3.2.

We observe that the dataset on which the embeddings were trained is also important. Even though glove-twitter has a significantly larger vocabulary, it covers a smaller percent of the lemmas in our dataset compared to glove-wiki. This is explained by the fact that terms of interest are less likely to appear in colloquial contexts. Note that even though fasttext can potentially be applied to out-of-vocabulary words, the baseline method still needs a fixed set of vectors for ranking. In this sense, the baseline can be considered a bridge between candidate based and candidate free methods as the set of candidates is the entire vocabulary.

For the KBERT model, we explored how the knowledge gained during pre-training can be transferred to our task by replacing the encoding layers and/or the classification head of the model with randomly initialized ones, and comparing the performance. Tabel 2 shows that the pretraining of the encoder layer and classification head plays an important role in model performance and ability to generalize.

For the KBERT-GAT model we have performed several experiments. As was mentioned before, the GAT part is very computationally expensive (even with the simplification in the architecture). Thus, the first experiments were dedicated to exploring the limits to which we could stretch the memory. Without alteration in the architecture, KBERT-GAT would not go any further that batch size 1 and one attention head. Fortunately, the simplification allowed to extend the model to four attention heads with hidden size 32 that would accommodate batch size 32 during training. These parameters were used in all further iterations. Next, we focused on testing the small variations within the same architecture. Namely, we compared two settings: pre-trained $\text{BERT}_{\text{BASE}}$ as an encoder with frozen embeddings and BertForMaskedLM as an encoder with trainable embeddings. The result are presented in a Table 3. One could observe that the first setting is better in terms of MAP and MRR, while the second settings has a significantly better Precision@k score. We admit that the scores are low and we suppose that the problem lies in the fact that we try to learn a classification model where some of the classes may not even appear in the train set. Even with the pretrained parts, we learn the classification head from scratch and it can't learn something generic enough. However, the exact reason for the low scores is still yet to be determined.

5 Conclusion

We first observe that the baseline proved to be a strong candidate, yielding reasonable results on the test split in terms of MRR and Precision. However, its inherent limitation of a fixed vocabulary suggests that it's wise to direct efforts to more flexible solutions.

The experiments performed using KBERT highlight the importance of pretraining. Furthermore, by freezing the head of the classification model we prevent a shift of the weights to overfit the training set.

We observe that using GAT as the prediction

Encoder	Head	Precision@10	MRR	MAP
-	-	0.00028	0.0010	0.0010
+	-	0.00030	0.0011	0.0011
+	+	0.00038	0.0011	0.0011
+	+*	0.00038	0.0014	0.0014

Table 2: Numeric results for the KBERT model as evaluated by the metrics described in section 3.2. Legend: - trained from scratch, + pre-trained, * freezed.

Base Model	Embeddings	Precision@10	MRR	MAP
BERT	Frozen	0.00018	0.00097	0.00097
BertForMaskedLM	Trainable	0.00025	0.00094	0.00094

Table 3: Numeric results for the KBERT-GAT model as evaluated by the metrics described in section 3.2.

head doesn't improve the metrics despite the increase in representation capacity of the network. This suggests that it might be too difficult to learn the weights of the GAT from scratch and it harms the ability of the model to generalize. Therefore, finding a way to regularize training and incorporate external knowledge is a feasible direction for development that might considerably boost the performance of the models (both KBERT and KBERT-GAT).

While the absolute values of the results appear low, they are far from a random choice. Considering the difficulty of the newly proposed task, our initial results are worthwhile and prove that the task of candidate-free taxonomy enrichment is within reach.

6 Contribution

6.1 Timotei Ardelean

Wrote the code infrastructure for training, validation (metrics) and experiment management. Implemented the baseline model and performed experiments with it. Contributed to the following sections of the report: Methodology, Results and discussion, Conclusion. Contributed to presentation.

6.2 Andreea Dogaru

Wrote the code for the KBERT model and performed experiments with it. Implemented multi-token prediction for BERT-based models. Contributed to the following sections of the report: Related Work, Methodology, Results and discussion. Contributed to presentation.

6.3 Gabriel Rozzonelli

Wrote the code for the data pre-processing and wrapping. Contributed to the following sections of the report: Introduction and motivation. Contributed to presentation.

6.4 Alsu Vakhitova

Wrote the code for the KBERT-GAT model and performed experiments with it. Contributed to the following sections of the report: Related Work, Methodology, Results and discussion. Contributed to presentation.

7 3rd party code list

The implementation of KBERT-GAT takes some inspiration from the Pytorch implementation of the Graph Attention Network (GAT) model: Github repository.

References

Eugene Agichtein and L. Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *DL '00*.

Daniele Alfarone and J. Davis. 2015. Unsupervised learning of an is-a taxonomy from a limited domain-specific corpus. In *IJCAI*.

Rami Aly, Shantanu Acharya, Alexander Ossa, Arne Köhn, Chris Biemann, and Alexander Panchenko. 2019. Every child should have parents: a taxonomy refinement algorithm based on hyperbolic term embeddings. In *ACL*.

Luis Espinosa Anke, Francesco Ronzano, and Horacio Saggion. 2016. Taln at semeval-2016 task 14: Semantic taxonomy enrichment via sense-based embeddings. In *SemEval@NAACL-HLT*.

Mohit Bansal, David Burkett, Gerard de Melo, and D. Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *ACL*.

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

Haw-Shiuan Chang, ZiYun Wang, L. Vilnis, and A. McCallum. 2018. Distributional inclusion vector embedding for unsupervised hypernymy detection. In *NAACL*.

Anne Cocos, Marianna Apidianaki, and Chris Callison-Burch. 2018. Comparing constraints for taxonomic organization. In *NAACL-HLT*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, H. Wang, and T. Liu. 2014. Learning semantic hierarchies via word embeddings. In *ACL*.

Amit Gupta, R. Lebret, Hamza Harkous, and K. Aberer. 2017. Taxonomy induction using hypernym subsequences. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*.

Z. Harris. 1954. Distributional structure. *WORD*, 10:146–162.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*.

Meng Jiang, Jingbo Shang, T. Cassidy, Xiang Ren, Lance M. Kaplan, T. Hanratty, and Jiawei Han. 2017. Metapad: Meta pattern discovery from massive text corpora. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

David Jurgens and Mohammad Taher Pilehvar. 2016. Semeval-2016 task 14: Semantic taxonomy enrichment. In *SemEval@NAACL-HLT*.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2019. K-BERT: enabling language representation with knowledge graph. *CoRR*, abs/1909.07606.

Anh Tuan Luu, Yi Tay, S. C. Hui, and S. Ng. 2016. Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In *EMNLP*.

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

Ndapandula Nakashole, G. Weikum, and Fabian M. Suchanek. 2012. Patty: A taxonomy of relational patterns with semantic types. In *EMNLP-CoNLL*.

R. Navigli, P. Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *IJCAI*.

Irina Nikishina, Alexander Panchenko, V. Logacheva, and Natalia V. Loukachevitch. 2020. Studying taxonomy enrichment on diachronic wordnet versions. In *COLING*.

Alexander Panchenko, Stefano Faralli, E. Ruppert, Steffen Remus, Hubert Naets, Cedric Fairon, Simone Paolo Ponzetto, and Chris Biemann. 2016. Taxi at semeval-2016 task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling. In *SemEval@NAACL-HLT*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Vassilis Plachouras, Fabio Petroni, Timothy Nugent, and Jochen L. Leidner. 2018. A comparison of two paraphrase models for taxonomy augmentation. In *NAACL-HLT*.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

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

Stephen Roller, Douwe Kiela, and Maximilian Nickel. 2018. Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *ACL*.

Jingbo Shang, Xinyang Zhang, Liyuan Liu, Sha Li, and Jiawei Han. 2020. Nettaxo: Automated topic taxonomy construction from text-rich network. *Proceedings of The Web Conference 2020*.

J. Shen, Zhihong Shen, Chenyan Xiong, Chunxin Wang, Kuansan Wang, and Jiawei Han. 2020. Taxoexpan: Self-supervised taxonomy expansion with position-enhanced graph neural network. *Proceedings of The Web Conference 2020*.

Przemyslaw Spurek, Tomasz Danel, Jacek Tabor, Marek Smieja, Lukasz Struski, Agnieszka Slowik, and Lukasz Maziarka. 2019. Geometric graph convolutional neural networks. *CoRR*, abs/1909.05310.

Hristo Tanev and A. Rotondi. 2016. Deftor at semeval-2016 task 14: Taxonomy enrichment using definition vectors. In *SemEval@NAACL-HLT*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

P. Velardi, Stefano Faralli, and R. Navigli. 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39:665–707.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *CoRR*, abs/1710.10903.

J. Wang, C. Kang, Yi Chang, and Jiawei Han. 2014. A hierarchical dirichlet model for taxonomy expansion for search engines. *Proceedings of the 23rd international conference on World wide web*.

C. Zhang, Fangbo Tao, Xiusi Chen, J. Shen, Meng Jiang, Brian M. Sadler, M. Vanni, and Jiawei Han. 2018. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *CoRR*, abs/2001.05140.