Global Bootstrapping Neural Network for Entity Set Expansion

Lingyong Yan[1,3], **Xianpei Han**[1,2,*], **Ben He**[3,1], **Le Sun**[1,2]
[1]Chinese Information Processing Laboratory [2]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
{lingyong2014, xianpei}@iscas.ac.cn, benhe@ucas.ac.cn, sunle@iscas.ac.cn

Abstract

Bootstrapping for entity set expansion (ESE) has been studied for a long period, which expands new entities using only a few seed entities as supervision. Recent end-to-end bootstrapping approaches have shown their advantages in information capturing and bootstrapping process modeling. However, due to the sparse supervision problem, previous end-to-end methods often only leverage information from near neighborhoods (local semantics) rather than those propagated from the co-occurrence structure of the whole corpus (global semantics). To address this issue, this paper proposes Global Bootstrapping Network (GBN) with the "pre-training and fine-tuning" strategies for effective learning. Specifically, it contains a global-sighted encoder to capture and encode both local and global semantics into entity embedding, and an attention-guided decoder to sequentially expand new entities based on these embeddings. The experimental results show that the GBN learned by "pre-training and fine-tuning" strategies achieves state-of-the-art performance on two bootstrapping datasets.

1 Introduction

Bootstrapping is a classical technique for entity set expansion (ESE), which starts from several seed entities of a specific category (e.g., {`London`, `Beijing`, `U.S.`} for GPE category) and then iteratively expands the entity set to cover more entities of the category (e.g., `Egypt` and `Harare`). Most previous ESE studies (Riloff and Jones, 1999; Curran et al., 2007; Yan et al., 2019) adopt the pipelined paradigm (see Figure 1a), which iteratively: evaluates patterns using seeds, matches and evaluates entities using patterns, adds top entities to the seed set. Such a pipelined paradigm makes it hard to represent the whole bootstrapping process as a single learnable model, and the implementation of ESE systems are often very ad hoc.

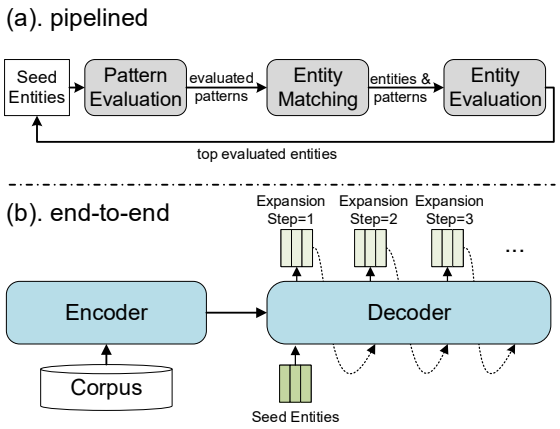


Figure 1: The illustration of the pipelined (a) and the end-to-end (b) bootstrapping paradigm.

Witnessed the drawbacks of the pipelined bootstrapping paradigm, recent studies start turning to the end-to-end paradigm. For instance, Yan et al. (2020) propose the first end-to-end bootstrapping neural network for ESE, which uses the encoder-decoder architecture (see Figure 1b): the encoder leverages and encodes the co-occurrence relations between entities and patterns into their embeddings; the decoder models bootstrapping as a sequential entity generation process, and the generated entities are used as expansion results. Compared with the pipeline paradigm, the end-to-end paradigm represents the whole bootstrapping process as a single learnable model and therefore is capable of leveraging more information and is more flexible.

One of the biggest challenges of end-to-end bootstrapping is how to learn it effectively since only very sparse supervision signals (i.e., several seed entities) are provided. In general, bootstrapping systems expand entities based on the entity-pattern duality assumption that "*similar entities will share similar patterns, and similar patterns will match*

*Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3705–3714
November 16 - 20, 2020.

similar entities". Based on this assumption, a bootstrapping network should be able to represent entities/patterns by leveraging both their near neighborhoods (local semantics) and the information propagated via the entity/pattern co-occurrence structure in the whole corpus (global semantics). Currently, using only several seeds as supervision signals, previous end-to-end bootstrapping models often only aggregate neighborhood information to represent entity/patterns, therefore the final representations of entities/patterns are mostly learned from their neighborhoods (short-sighted), rather than from global information (global-sighted). This raises a big issue because most entities are long-tail (Zipf, 1935), which will only match a limited number of patterns, and as a result, the local semantics cannot provide reliable and informative representations for effective bootstrapping (see Figure 2 for an example).

To address the sparse supervision problem, this paper proposes a new end-to-end bootstrapping neural network for ESE, called Global Bootstrapping Network (GBN), which can effectively capture the global information of a corpus via an augmented entity-pattern bipartite graph, and learn to leverage both the local and the global semantics for bootstrapping via effective pre-training and fine-tuning strategies. Our method is motivated by the recent success of the pre-training and fine-tuning strategy in addressing the sparse supervision challenges (Devlin et al., 2019; Hu et al., 2020).

Concretely, the Global Bootstrapping Network adopts the encoder-decoder architecture. The encoder is a global-sighted graph neural network, in which each layer aggregates rich information not only between directly linked entities and patterns but also the entities and patterns multi-hop away via augmented links. The decoder is an attention-guided RNN model, which efficiently generates expansion results based on the global-sighted entity representations. Compared with previous methods, GBN can also effectively aggregate the global information rather than only local neighborhood information, therefore it is more reliable even for the long-tail entities/patterns with sparse links.

To learn the GBN, we propose several pre-training and fine-tuning algorithms: 1) In the pre-training stage, we design both the self-supervised and the supervised pre-training strategies to learn the encoder in the GBN, which ensures the learned representations of entities/patterns will capture both the local and global semantics. 2) In the fine-tuning stage, based on the learned representation, we use a multi-view learning algorithm to fine-tune GBN to fit a specific bootstrapping task using only a few seeds.

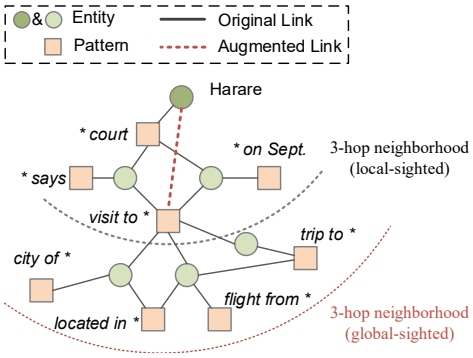


Figure 2: An example of local/global-sighted neighborhood with/without augmented links (we use a long-tail GPE–"Harare" as the center entity). By adding an augmented link, "Harare" can easily observe its global-sighted neighborhood such as the strong GPE patterns–"visit to *", "located in *", etc., and therefore it can be accurately expanded.

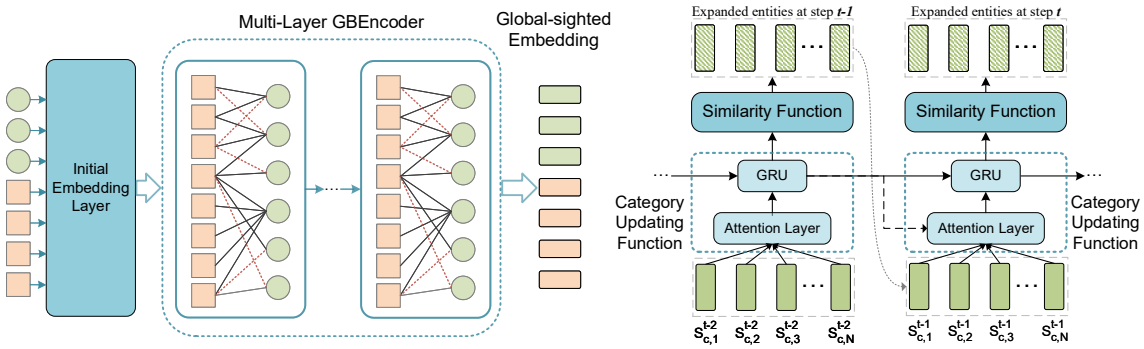
To summarize, the main contributions are:

1. We propose a new end-to-end bootstrapping neural network—GBN, to leverage the global-sighted information and encode the global-sighted information into entity embeddings.
2. We propose a novel pre-training and fine-tuning strategy for learning bootstrapping network with only sparse supervision signals. In pre-training, our method learns entity/pattern representations by effectively exploiting co-occurrence information in the corpus, and in fine-tuning, our method can be easily fitted to a specific bootstrapping task.

2 Entity-Pattern Bipartite Graph Construction

This section describes how to construct the entity-pattern bipartite graph, which captures the global structure of entity-pattern co-occurrences in the original ESE corpus. Furthermore, augmented links are added for long-tail entities/patterns.

Traditionally, entity-pattern duality is modeled as a set of individual $\langle entity, pattern \rangle$ entries (Riloff and Jones, 1999; Curran et al., 2007; Shi et al., 2014), e.g., {$\langle$Harare, * *court*$\rangle$, $\langle$London, *visit to* *$\rangle$, ...}. However, such a data model considers different $\langle entity, pattern \rangle$ entries



(a) GBEncoder (b) GBDecoder

Figure 3: The overall architecture of Global Bootstrapping Network.

independently, makes it hard to leverage the global co-occurrence structure.

To capture both the local and global semantics, we follow Yan et al. (2020) and use the entity-pattern bipartite graph. Concretely, the entities and patterns are represented as graph nodes, and an entity and a pattern will be linked if the pattern matches the entity in the corpus. Finally, the entity-pattern bipartite graph is formulated as a tuple $G =< V, E, S >$, where V is the node set of entities and patterns, E is the set of edges connecting entities and patterns, S is the set of seed entities with corresponding labels.

Graph augmentation. In real-world corpus, entities/patterns usually follow the long-tail distribution, therefore most entities/patterns have only a few links to others. Such a sparse neighborhood makes it challenging to effectively leverage both the local and global semantics (see Figure 2).

To address this issue, we design to add augmented links to the constructed graph. Specifically, if there exist at least M paths $\leq K$ hops between an unlinked entity and pattern pair, we will add an augmented link between them (see Figure 2). In this paper, we set M and K both as 2 for efficiency and effectiveness[1].

3 Global Bootstrapping Network

This section describes the Global Bootstrapping Network (GBN), which adopts the encoder-decoder architecture (see Figure 3) and contains:

- **GBEncoder**: a global-sighted GNN encoder, which takes the augmented bipartite graph as the input and encodes both local and global semantics into entity/pattern embeddings.
- **GBDecoder**: an attention-guided RNN decoder, which iteratively generates new entities as expansion results based on their global-sighted embeddings.

3.1 GBEncoder

Given the entity-pattern bipartite graph, the GBEncoder embeds entities and patterns by leveraging both the local and the global semantics. The global-sighted embeddings can be further leveraged to perform the global-sighted entity set expansion.

Architecture. To capture both the local and global semantics, we use a multi-layer graph neural network, where each layer aggregates information from node neighborhood through both original links and augmented links as:

$$h_i^l = \sigma(h_i^{l-1} + \sum_{j \in N(i)} a_j^{l-1} h_j^{l-1}) \quad (1)$$

where i represents the i-th node to be updated, $N(i)$ is the set of nodes linked to the i-th node by both original and augmented links, h_i^l is the node representation after the l-th layer, a_j^{l-1} is the updating weight for neighboring node j, σ is a non-linear mapping function (this paper uses the ReLU).

Attention mechanism. The updating weights of Eq.1 is critical for finding out related patterns/entities and filtering out noises. To estimate it accurately, we use the attention mechanism:

$$a_j = \frac{exp(e_{i,j})}{\sum_{k \in N(i)} exp(e_{i,k})} \quad (2)$$
$$e_{i,k} = g(W^a h_i, W^b [h_k; d_k; t_k])$$

where $g(\cdot)$ is the scaled dot production-based attention function, W^a and W^b are learnable parameter matrices. To calculate the attention score, we use the following three features:

- Node feature h_k: the representation of node k from the last layer.

[1]Both M and K are set according to the pilot experiments: $M > 2$ will produce only a few augmented links, and $K > 2$ will result in the GPU memory overflow.

- Distance feature d_k: a learnable distance embedding. The distance of two nodes equals to: 1 if they are directly linked; 2 if they are linked by an augmented link.
- Link type feature t_k: a learnable link type embedding. This paper uses three link types: *before*, *middle* and *after*[2].

Node initialization. This paper initializes entity/pattern representations by their average token embeddings using pre-trained GloVe tables (Pennington et al., 2014). There are many other choices for initialization, such as CNN and BERT (Devlin et al., 2019). Based on the flying experiments, this paper adopts the average token embeddings for its simplicity and effectiveness.

Compared to previous end-to-end model (Yan et al., 2020), the GBEncoder is different in two aspects: 1). It can leverage more information between entities by introducing distance information and link type features; 2). It has a more global-sighted perceptual field by explicitly modeling augmented links and passing messages through them.

3.2 GBDecoder

Using the global-sighted entity embeddings from GBEncoder, the GBDecoder sequentially generates expanded entities using a recurrent neural network.

Specifically, based on the global-sighted embeddings, the GBDecoder is a GRU (Cho et al., 2014)-based model, where the GRU hidden state is used as the category embedding. The GBDecoder expands entities in the following bootstrapping schema:

1. At the very beginning, the seed entities are used to update the category embedding using the ***category updating function***.
2. The unexpanded entities are evaluated based on their similarities to the category embedding calculated by ***similarity function***, and the top ones are expanded.
3. The expanded entities are added to the seed set, and the category updating function will be used to update the category embedding.
4. Go to step 2 unless reaching the end iteration.

Attention-guided category updating function. To adaptively capture the target category semantics throughout the bootstrapping process, the GBDecoder updates the GRU hidden state (category embedding) using previous expanded entities at each step as the follows:

$$\begin{aligned} z_c^t &= \sigma(W_z \cdot s_c^t + U_z \cdot h_c^{t-1}) \\ r_c^t &= \sigma(W_r \cdot s_c^t + U_r \cdot h_c^{t-1}) \\ \bar{h}_c^t &= \sigma(W \cdot s_c^t + r_c^t \cdot U \cdot h_c^{t-1}) \\ h_c^t &= z_c^t \odot h_c^{t-1} + (1 - z_c^t) \odot \bar{h}_c^t \end{aligned} \tag{3}$$

where h_c^{t-1} is the hidden state vector of category c after step $t-1$ (h_c^0 is set to all-zero), and s_c^t is the embedding of the expanded entities of the last step.

To avoid introducing noises when updating the category embedding, it is crucial to filter out the noisy expansions from the last step. Therefore, we use the attention mechanism[3] to compute s_c^t:

$$\begin{aligned} s_c^t &= \sum_{i=1}^{N} \alpha_{c,i}^{t-1} \cdot s_{c,i}^{t-1} \\ \alpha_{c,i}^{t-1} &= \frac{\exp(g(h_c^{t-1}, s_{c,i}^{t-1}))}{\sum_j \exp(g(h_c^{t-1}, s_{c,i}^{t-1}))} \end{aligned} \tag{4}$$

where $s_{c,i}^{t-1}$ is the i-th expanded entity embedding of category c at step $i-1$, $g(\cdot)$ is a score function (this paper uses the scaled dot production). And we set s_c^t to all-zero if there is no expanded entity.

Similarity function. This paper calculates the similarity using the cosine similarity:

$$sim(v_i, h_c^t) = \frac{{v_i}^T h_c^t}{||v_i||_2 ||h_c^t||_2} \tag{5}$$

where v_i is the global-sighted embedding of entity i. And the top-N unexpanded entities with the highest similarity scores will be expanded at step t.

4 Learning GBN with Pre-training and Fine-tuning

In this section, we describe how to learn GBN effectively using the "pre-training and fine-tuning" (Devlin et al., 2019). In the pre-training stage, we adopt both the self-supervised and the supervised pre-training algorithms to pre-train the GBEncoder; in the fine-tuning stage, we adopt the multi-view learning algorithm to fine-tune both the GBEncoder and the GBDecoder. In this way, the sparse supervision problem can be effectively resolved.

4.1 Pre-training Strategies

The pre-training stage mainly aims to pre-train the GBEncoder to effectively capture both the local

[2] *before*, *middle* and *after* are corresponds respectively to the entity appearing before/within/after the pattern.

[3] Yan et al. (2020) use the average embedding of last expanded entities as s_c^t, which can be regarded as a special case of our method.

and global semantics from entity-pattern graphs.

Specifically, we want the GBEncoder to aggregate related information for all entities and patterns from their global-sighted neighborhoods while ignoring the noises. To this end, the GBEncoder should be able to effectively leverage as much information as possible from the dataset and task definition, including the inherent structural information within each dataset (**self-supervised**) and the labeled entity/pattern information within human-annotated datasets (**supervised**).

Self-supervised pre-training. The self-supervised pre-training strategies are designed to leverage the structural information included within the dataset and the task definition without the help of manually-labeled supervision signals. And we design the following learning algorithm for self-supervised pre-training:

- *Neighborhood learning*. This strategy mainly learns to discriminate the neighboring nodes of a certain node and the nodes many hops away from it. This is because the links between entities and patterns usually indicate relevance between them; on the opposite side, the long path between them usually indicates the irrelevance. Therefore, we want the learned entity and pattern embeddings more similar if they are neighbors than if they are many hops away. To this end, we try to maximize the following function:

$$CL(i) = \sum_{j \in N(i)} \frac{c_{i,j}}{c_{i,j} + \sum_{k \in N'(i)} c_{i,k}}$$
$$c_{i,j} = \exp\left(\frac{v_i^T v_j}{||v_i|| \cdot ||v_j||}\right) \quad (6)$$

  where v_i is the outputted embedding of node i, $N(i)$ is the set of directly linked nodes of node i, $N'(i)$ is the set of nodes at least n-hops way from i. In this paper, we set $n = 20$ following Yan et al. (2020).
- *Masked link prediction*. This strategy learns to predict the masked links between entities and patterns. Specifically, we randomly mask a fixed ratio r of existing links between entities and patterns in the bipartite graph; then we use the GBEncoder to encode the masked graph; finally, we use the following function to predict whether the link is masked between entity i and pattern j:

$$LP(i, j) = \sigma(g_{LP}([e_i; p_j])) \quad (7)$$

  where e_i and p_j are corresponding embeddings of entity i and pattern j outputted by GBEncoder, $g_{LP}(\cdot)$ is an MLP function. In this paper, we experimentally set r as 0.1. For training, we sample one negative link per masked link.

Algorithm 1 Multi-View Fine-Tuning Algorithm

Require: A bipartite graph G, seed entities (SEs)
1: Construct GBTeacher with the GBEncoder followed by an MLP classifier
2: Learn GBTeacher using SEs, and predict entity labels using the GBTeacher
3: **while** NOT reach the finish iteration **do**
4: Learn GBN using predicted entity labels and expand seeds using GBN
5: Learn GBTeacher using SEs and expanded entities, and predict entity labels using the GBTeacher
6: **end while**

Supervised pre-training. In addition, some datasets provide manually-labeled node types, which can be good supplementary supervision. And we exploit them using the following algorithm:

- *Node label prediction*. This strategy mainly learns to predict the given entity labels in the supervised datasets. Specifically, we use the following function to predict the entity labels:

$$TP(i) = \sigma(g_{TP}(e_i)) \quad (8)$$

  where $g_{TP}(\cdot)$ is another MLP function and $\sigma(\cdot)$ is the sigmoid activation function.

Note that, since the GBDecoder needs the seed entities (supervision signals) to start the bootstrapping process, which cannot be pre-trained unsupervisedly. Therefore, this paper does not pre-train but only fine-tunes it without loss of generality.

4.2 Fine-tuning via Multi-View Learning

After pre-training the GBEncoder, this paper fine-tunes both the GBEncoder and the GBDecoder using the multi-view learning algorithm proposed by Yan et al. (2020) on the bootstrapping dataset.

Specifically, this paper first constructs an auxiliary neural network to directly predict the entity labels, called GBTeacher, which contains a GBEncoder followed by an MLP classifier. Then we iteratively optimize the GBTeacher and GBN as the following steps (see Alg. 1):

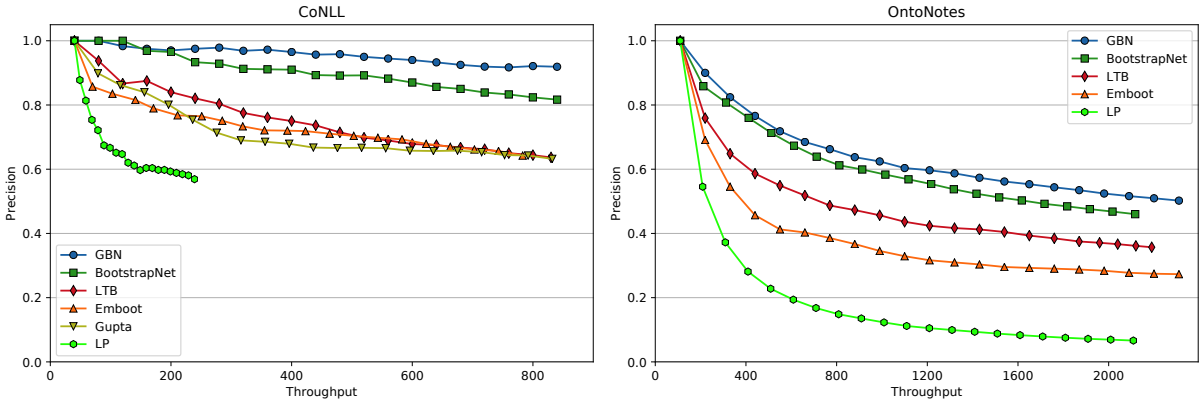


Figure 4: The precision-throughput curve on CoNLL and OntoNotes.

- **Learning GBTeacher**: this step uses the seed entities plus the labeled entities expanded by GBN to fit the GBTeacher.
- **Learning GBN**: this step uses the predicted entity labels by the optimized GBTeacher to fit the GBN.

5 Experiments

5.1 Experimental Setup

Datasets: Following Zupon et al. (2019) and Yan et al. (2020), we use CoNLL and OntoNotes datasets. CoNLL is constructed from the CoNLL 2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003), which contains 4 entity types. OntoNotes is a sparse dataset constructed from the OntoNotes datasets (Pradhan et al., 2013) but without numerical categories, which contains 11 entity types. The patterns are n-grams ($n \leq 4$).

As for the pre-training datasets, we use Wikigold (Balasuriya et al., 2009), GUM (Zeldes, 2017) and half of the DocRED (Yao et al., 2019) for supervised pre-training; we use the remaining half of the DocRED without labels for self-supervised pre-training[4].

Baselines. We use the following baselines:

1). LP[5]: this is the classical label propagation method, which propagates the seed labels to other entities based on the co-occurrence features.

2). Gupta (Gupta and Manning, 2014): this is a classical bootstrapping system that evaluates patterns and new entities by learning an entity classifier[6].

3). Emboot (Zupon et al., 2019): this method follows Gupta and Manning (2014), but learns custom word embeddings for entities and patterns, which are used to guide the entity classifier.

4). LTB (Yan et al., 2019): this method performs the lookahead search to capture more information for each entity using the MCTS algorithm.

5). BootstrapNet (Yan et al., 2020): this method uses an end-to-end model to capture information from entity/pattern neighborhoods and expand seeds without attention mechanism. In other words, this is the short-sighted baseline of our method on both model and learning algorithms.

Metrics.To evaluate these methods, we follow Zupon et al. (2019) to report the cumulative precision-throughput curve. And we also report the P@Iter.K[7](the precision after K-th expansion iterations, $K = 1, 10, 20$) and the corresponding MAP (the mean average precision).

Other Settings. Our pre-training strategy is to first perform the self-supervised pre-training and then the supervised pre-training on the pre-training datasets. After that, we fine-tune the GBN on the bootstrapping datasets.

For all methods, we run them 20 bootstrapping iterations and expand 10 entities per iteration. We set the layer number of the GBEncoder as 3, the learning rate as 1e-3. We implemented our model using PyTorch (Paszke et al., 2019) with the PyTorch Geometric extension (Fey and Lenssen, 2019). And all models are run on a single Nvidia TiTan RTX[8].

[4]Due to the limited scalability of implementing GNN, we split the DocRED into small ones (each contains $\leq 2,000$ documents).

[5]LP is implemented using the scikit-learn package.

[6]Because the labels of its builtin NER model mismatch the labels in the OntoNotes, we don't run it on the OntoNotes.

[7]Because we mainly study the precision at different iterations, P@Iter.K is more intuitive.

[8]Our codes and datasets are available at https://www.github.com/lingyongyan/bootstrapping_pre-train.

	CoNLL				OntoNotes			
	P@Iter.1	P@Iter.10	P@Iter.20	MAP	P@Iter.1	P@Iter.10	P@Iter.20	MAP
GBN	**1.000**	**0.953**	**0.915**	**0.956**	**0.800**	**0.556**	**0.477**	**0.611**
GBN_{-gs}	1.000	0.920	0.868	0.929	0.709	0.447	0.381	0.512
GBN_{-pt}	0.825	0.618	0.534	0.659	0.600	0.326	0.285	0.404

Table 1: The ablation study results of GBN. GBN_{-gs} is the model without global-sighted encoder; GBN_{-pt} is the model not learned by the "pre-training and fine-tuning" strategies.

5.2 Overall Results

Figure 4 shows the overall results on CoNLL and OntoNotes. From this figure, we can see that:

- **GBN significantly outperforms all baselines**. On both CoNLL and OntoNotes, the proposed GBN can expand entities with higher precision compared with the baselines. Specifically, on the CoNLL, GBN can expand 800 entities with the precision more than 90%, while the baselines can achieve at most 80%, the LP method even can not expand more than 300 entities; on the OntoNotes, GBN can also expand 2200 entities with the precision more than 47%, while the precisions of most other baselines are less than 40%, BootstrapNet can achieve around 45% precision in the end, but its final expanded entities are less than 2000.

- **End-to-end paradigm is promising for bootstrapping**. From the Figure 4, we can see that both two end-to-end models– GBN and BootstrapNet can achieve better performance than other pipelined methods in two aspects: compared with the pipelined methods, both end-to-end models can achieve significantly higher precision; the precision-throughput curves decrease more slightly with the increases of the throughput on CoNLL and OntoNotes datasets.

5.3 Detailed Analysis

Ablation study of GBN. To detailedly analyze the contribution of the global-sighted encoding and the "pre-training and fine-tuning" strategy, we conduct ablation study on the two datasets (see Table 1), where GBN_{-gs} replaces the global-sighted encoder in GBN with a simple graph attention network (Veličković et al., 2018); GBN_{-pt} denotes a variant of the GBN model that is not learned by "pre-training and fine-tuning" strategy but rather by the multi-view learning algorithm like Yan et al. (2020).

	P@Iter.1	P@Iter.10	P@Iter.20	MAP
GBN	**1.000**	**0.953**	**0.915**	**0.956**
$GBN_{\text{-self}}$	1.000	0.935	0.908	0.948
$GBN_{\text{-sup}}$	0.900	0.728	0.640	0.756
$GBN_{\text{-both}}$	0.825	0.618	0.534	0.659

(a) CoNLL

	P@Iter.1	P@Iter.10	P@Iter.20	MAP
GBN	**0.800**	**0.556**	**0.477**	**0.611**
$GBN_{\text{-self}}$	0.709	0.534	0.450	0.564
$GBN_{\text{-sup}}$	0.636	0.337	0.289	0.421
$GBN_{\text{-both}}$	0.600	0.326	0.285	0.404

(b) OntoNotes

Table 2: The performance of GBN with different pre-training strategies.

We can see that, without the global-sighted encoding, the final performance may decrease even with the "pre-training and fine-tuning" strategy. This indicates that our proposed global-sighted encoder can effectively capture global-sighted information than other encoder models. From Table 1, we can also see that, without using the "pre-training and fine-tuning" strategy, the performance of GBN decreases sharply. This verifies the importance of the "pre-training and fine-tuning" strategies for bootstrapping tasks. Furthermore, we found that "pre-training and fine-tuning" is critical for models with a large capacity: there is a large performance gap between GBN_{-pt} and GBN. Therefore we believe that the capacity of models should be consistent with its learning algorithms and supervision signals: an expressive model with a weak learning algorithm may not result in a strong performance.

Effect of different pre-training strategies. To further analyze the effect of different pre-training strategies, we conduct another ablation study by ablating the self-supervised pre-training strategies ($GBN_{\text{-self}}$), the supervised pre-training strategy ($GBN_{\text{-sup}}$) and both of them ($GBN_{\text{-both}}$). The results are shown in Table 2. We can see that: 1). Both the self-supervised pre-training strategies and

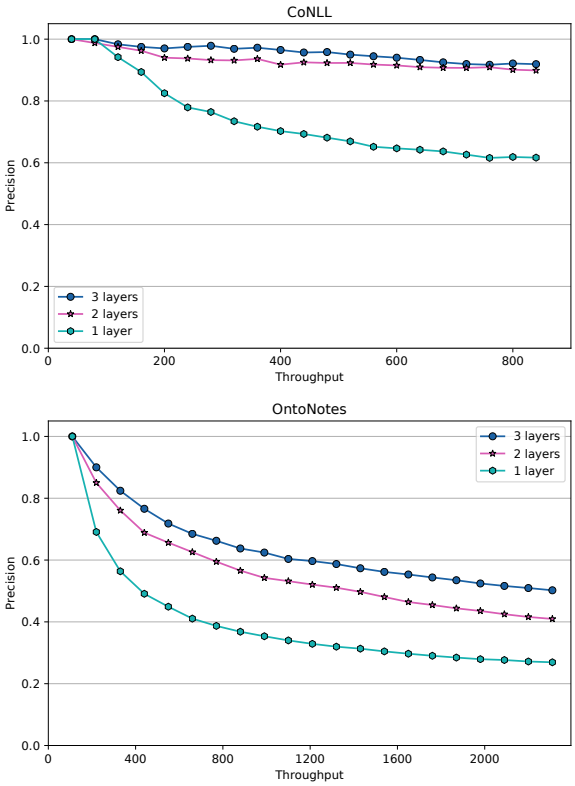


Figure 5: The performance of GBN with different layer numbers of GBEncoder.

the supervised pre-training strategy are effective for GBN's final performance. 2). Compared to the supervised pre-training strategy, self-supervised pre-training strategies obtain less performance improvement. This could be explained by the fact that the pre-training and the bootstrapping datasets are often with different structures, making it more difficult to capture structural information via self-supervised pre-training strategies.

Effect of Encoder layers. To analyze the effect of layer numbers of GBEncoder, we conduct experiments with different layer numbers (see Figure 5). From Figure 5, we can see that the performance of the GBN increases with more layers, which also indicates that the performance of bootstrapping methods for ESE can benefit from effectively capturing more global-sighted information, as more layers we used, more global-sighted information can be captured.

6 Related Work

Bootstrapping. Bootstrapping is a widely studied technique in IE (Riloff and Shepherd, 1997; Qadir et al., 2015; Gupta et al., 2018), as well as word sense disambiguation (Yoshida et al., 2010), entity translation (Lee and Hwang, 2013), model learning (Whitney and Sarkar, 2012), etc.

Currently, bootstrapping methods for ESE can be categorized into two paradigms: pipelined paradigm and end-to-end paradigm (Yan et al., 2020). The pipelined methods (Riloff and Jones, 1999; Collins and Singer, 1999) mainly leverage direct co-occurrence information, which will easily lead to the semantic drifting problem (Curran et al., 2007). To resolve this problem, many pipelined methods are proposed, e.g., mutual exclusive bootstrapping (Curran et al., 2007; McIntosh and Curran, 2008, 2009; Gupta et al., 2018), bootstrapping using negative seeds (Yangarber et al., 2002; Shi et al., 2014), lexical and statistical features (Liao and Grishman, 2010; Gupta and Manning, 2014), word embeddings (Batista et al., 2015; Gupta and Manning, 2015; Zupon et al., 2019), active learning (Berger et al., 2018), lookahead search (Yan et al., 2019), etc. Recently Yan et al. (2020) propose an end-to-end bootstrapping model and show its advantages in information leveraging and flexibility. Besides, there are some other studies that focus on the web-based ESE (Tong and Dean, 2008; Carlson et al., 2010; Chen et al., 2016), which heavily relies on the base search engine.

Pre-training and fine-tuning. The early pre-trained models on the ImageNet (Russakovsky et al., 2015) show its advantages in many CV tasks (Simonyan and Zisserman, 2014; Johnson et al., 2016; Huang et al., 2017; He et al., 2017). In NLP, the pre-training has also been proven its effectiveness on many tasks, including the early word vectors such as word2vec or Glove (Mikolov et al., 2013; Pennington et al., 2014) and recent language model pre-training such as Elmo (Peters et al., 2018), BERT (Devlin et al., 2019) and XLNet (Yang et al., 2020). Recently, Hu et al. (2020) also show the advantages of graph pre-training, which directly inspires our work.

7 Conclusions

In this paper, we propose the Global Bootstrapping Network (GBN) and effective "pre-training and fine-tuning" strategies to learn it. Specifically, we design global-sighted GBEncoder to capture both local and global semantics from the corpus and an effective attention-guided GBDecoder to adaptively expand new entities. To learn GBN, we design several pre-training and fine-tuning strategies. Experiments show that the proposed GBN together with "pre-training and fine-tuning" al-

gorithm significantly outperforms state-of-the-art methods. For future work, we want to design more effective "pre-training and fine-tuning" strategies and apply our model on other bootstrapping tasks.

Acknowledge

This research work is supported by the National Key Research and Development Program of China under Grant No.2017YFB1002104, the National Natural Science Foundation of China under Grants no. U1936207, Beijing Academy of Artificial Intelligence (BAAI2019QN0502), and in part by the Youth Innovation Promotion Association CAS(2018141).

References

Dominic Balasuriya, Nicky Ringland, Joel Nothman, Tara Murphy, and James R Curran. 2009. Named entity recognition in wikipedia. In *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources (People's Web)*, pages 10–18.

David S. Batista, Bruno Martins, and Mário J. Silva. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of EMNLP*, pages 499–504.

Matthew Berger, Ajay Nagesh, Joshua Levine, Mihai Surdeanu, and Helen Zhang. 2018. Visual Supervision in Bootstrapped Information Extraction. In *Proceedings of EMNLP*, pages 2043–2053, Brussels, Belgium.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2010. Coupled Semi-supervised Learning for Information Extraction. In *Proceedings of WSDM*, pages 101–110, New York, NY, USA.

Zhe Chen, Michael Cafarella, and H. V. Jagadish. 2016. Long-tail Vocabulary Dictionary Extraction from the Web. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 625–634, New York, NY, USA. ACM. Event-place: San Francisco, California, USA.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv:1409.1259 [cs, stat]*. ArXiv: 1409.1259.

Michael Collins and Yoram Singer. 1999. Unsupervised Models for Named Entity Classification. In *EMNLP*.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of PACLING*, volume 6, pages 172–180. Citeseer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, Minneapolis, Minnesota.

Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.

Pankaj Gupta, Benjamin Roth, and Hinrich Schütze. 2018. Joint Bootstrapping Machines for High Confidence Relation Extraction. In *Proceedings of NAACL-HLT*, pages 26–36, New Orleans, Louisiana.

Sonal Gupta and Christopher Manning. 2014. Improved Pattern Learning for Bootstrapped Entity Extraction. In *Proceedings of CoNLL*, pages 98–108, Ann Arbor, Michigan.

Sonal Gupta and Christopher D. Manning. 2015. Distributed Representations of Words to Guide Bootstrapped Entity Classifiers. In *Proceedings of NAACL-HLT*, pages 1215–1220, Denver, Colorado.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. *arXiv:1905.12265 [cs, stat]*. ArXiv: 1905.12265.

Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. 2017. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer.

Taesung Lee and Seung-won Hwang. 2013. Bootstrapping Entity Translation on Weakly Comparable Corpora. In *Proceedings of ACL*, pages 631–640.

Shasha Liao and Ralph Grishman. 2010. Filtered Ranking for Bootstrapping in Event Extraction. In *Proceedings of COLING*, pages 680–688.

Tara McIntosh and James R. Curran. 2008. Weighted Mutual Exclusion Bootstrapping for Domain Independent Lexicon and Template Acquisition. In *Proceedings of ALTA*, pages 97–105, Hobart, Australia.

Tara McIntosh and James R. Curran. 2009. Reducing semantic drift with bagging and distributional similarity. volume 1, pages 396–404. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of EMNLP*, pages 1532–1543, Doha, Qatar.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards Robust Linguistic Analysis using OntoNotes. In *Proceedings of CoNLL*, pages 143–152, Sofia, Bulgaria.

Ashequl Qadir, Pablo N. Mendes, Daniel Gruhl, and Neal Lewis. 2015. Semantic Lexicon Induction from Twitter with Pattern Relatedness and Flexible Term Length. In *Proceedings of AAAI*, pages 2432–2439.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI/IAAI*, pages 474–479.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. *arXiv preprint cmp-lg/9706013*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Bei Shi, Zhenzhong Zhang, Le Sun, and Xianpei Han. 2014. A probabilistic co-bootstrapping method for entity set expansion. In *Proceedings of COLING*, pages 2280–2290.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the HLT-NAACL*, pages 142–147.

Simon Tong and Jeff Dean. 2008. System and methods for automatically creating lists. US Patent 7,350,187.

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

Max Whitney and Anoop Sarkar. 2012. Bootstrapping via Graph Propagation. In *Proceedings of ACL*, pages 620–628.

Lingyong Yan, Xianpei Han, Ben He, and Le Sun. 2020. End-to-end bootstrapping neural network for entity set expansion. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Lingyong Yan, Xianpei Han, Le Sun, and Ben He. 2019. Learning to bootstrap for entity set expansion. In *Proceedings of EMNLP*, pages 292–301.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv:1906.08237 [cs]*. ArXiv: 1906.08237.

Roman Yangarber, Winston Lin, and Ralph Grishman. 2002. Unsupervised Learning of Generalized Names. In *Proceedings of COLING*, pages 1–7.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A Large-Scale Document-Level Relation Extraction Dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy. Association for Computational Linguistics. Citation Key Alias: yao_DocREDLargeScaleDocumentLevel_2019.

Minoru Yoshida, Masaki Ikeda, Shingo Ono, Issei Sato, and Hiroshi Nakagawa. 2010. Person Name Disambiguation by Bootstrapping. In *Proceedings of SIGIR*, pages 10–17. ACM.

Amir Zeldes. 2017. The gum corpus: creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

George Zipf. 1935. The psych-biology of language.

Andrew Zupon, Maria Alexeeva, Marco Valenzuela-Escárcega, Ajay Nagesh, and Mihai Surdeanu. 2019. Lightly-supervised Representation Learning with Global Interpretability. In *Proceedings of the Third Workshop on Structured Prediction for NLP*, pages 18–28.