End-to-End 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 long been modeled as a multi-step pipelined process. Such a paradigm, unfortunately, often suffers from two main challenges: 1) the entities are expanded in multiple separate steps, which tends to introduce noisy entities and results in the semantic drift problem; 2) it is hard to exploit the high-order entity-pattern relations for entity set expansion. In this paper, we propose an end-to-end bootstrapping neural network for entity set expansion, named BootstrapNet, which models the bootstrapping in an encoder-decoder architecture. In the encoding stage, a graph attention network is used to capture both the first- and the high-order relations between entities and patterns, and encode useful information into their representations. In the decoding stage, the entities are sequentially expanded through a recurrent neural network, which outputs entities at each stage, and its hidden state vectors, representing the target category, are updated at each expansion step. Experimental results demonstrate substantial improvement of our model over previous ESE approaches.

Introduction

Bootstrapping is a classical technique for entity set expansion (ESE) (Riloff and Jones 1999), as well as many other information extraction tasks such as relation extraction (Batista, Martins, and Silva 2015; Gupta, Roth, and Schütze 2018) and event extraction (Liao and Grishman 2010). Bootstrapping has long been modeled as a multi-step pipelined process. Starting from several seed instances, a bootstrapping ESE system expands new instances by iteratively matching patterns using expanded entities, evaluating and selecting new patterns, matching entities using selected patterns, and eventually, expanding new entities based on the quality evaluation (see example in Figure 1).

However, such a multi-step pipeline paradigm often suffers from two challenges.

First, the entities are expanded in multiple separate steps, which tends to introduce noisy entities and results in the semantic drift problem (Curran, Murphy, and Scholz 2007). In bootstrapping, the expansion results in previous steps are used to boost the successive expansions; and in the pipeline paradigm, the expanded entities from previous steps are directly used as the golden entities to guide the next expansion (Riloff and Jones 1999; Curran, Murphy, and Scholz 2007; Gupta and Manning 2014; Yan et al. 2019), which may cause the expansion process drifting to other categories as not all expanded entities are correct. For example, in Figure 1, when expanding the geopolitical entities (GPE), the noisy entity "NASA" selects a general pattern "* said", resulting in selection of other noisy entities ("Ford" and "Allen") in the subsequent steps. To resolve the semantic drift problem, we argue that the expansion steps should be tightly coupled so that: 1) more information from previous expansion steps can be better exploited, e.g., the confidence scores of entities; 2) future expansion results can be used as feedback for filtering out noisy entities.

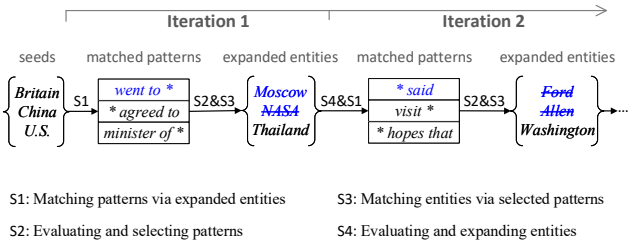


Figure 1: An example of the bootstrapping process by unfolding the bootstrapping iterations to a sequence. Patterns and entities in blue are top scored ones to be selected. Entities with strikethroughs are noisy entities.

Second, it is hard to exploit the high-order entity-pattern relations for entity set expansion. Due to the lack of extra supervision, previous studies mainly used the entity-pattern matching statistics for entity/pattern evaluation (Riloff and Jones 1999; Curran, Murphy, and Scholz 2007; Shi et al. 2014; Zupon et al. 2019). However, these methods only exploit the first-order entity-pattern relations, while ignoring the useful information from the high-order relations, which have been proven useful in many information extraction tasks (Riedel et al. 2013; Chen et al. 2006).

To address the above two problems, this paper proposes a

*Corresponding authors.

neural network for ESE, named BootstrapNet, which models bootstrapping in an end-to-end manner via an encoder-decoder architecture. Specifically, we use the entity-pattern bipartite graph constructed from the corpus as our model input. In the encoding stage, a multi-layer graph attention network (Veličković et al. 2018) is used to encode correlation information between entities and patterns, which can naturally exploit both the first- and the high-order relations in the constructed bipartite graph. Self-attention mechanism (Vaswani et al. 2017) is used to avoid noisy relations. Comparing to the traditional bootstrapping methods, our model can effectively capture both the first- and the high-order relations from the bipartite graph, which is more informative than using only the first-order evidence (i.e., the entity-pattern matching statistics). In the decoding stage, a recurrent neural network (RNN) is used to sequentially expand entities. The RNN's outputs are new entities, and its hidden state vectors are used as category embeddings. At each step, the expanded entities are used to update the category embeddings, and new entities are expanded based on their similarity to the updated category embeddings. Comparing to the traditional bootstrapping methods, our model can effectively exploit various sources of useful information in the expanded results, such as the uncertainty of the expanded entities and the correlations between expanded entities, and such information is conveniently used on-the-fly. Furthermore, our model can leverage the successive expansion results as the feedback for previous expansion steps via the backpropagation through time (BPTT) learning.

To learn our BootstrapNet, we devise a multi-view based learning algorithm, which can efficiently learn our model using a small set of seed entities. Experimental results show that our method outperforms the traditional bootstrapping methods and can significantly reduce the semantic drift.

Our contributions are:

- We propose to model the entire bootstrapping process in an encoder-decoder architecture. To our best knowledge, this is the first attempt to fully model the whole bootstrapping process using a single neural network.
- We propose the BootstrapNet, which can capture both the first- and the high-order relations using a graph neural network as the encoder, and sequentially, rather than separately, expand new entities using a recurrent neural network as the decoder.
- We design an efficient learning algorithm for BootstrapNet with only several seeds as the supervision signals.

Entity-Pattern Bipartite Graph

Given several seed entities, an ESE bootstrapping system often expands entities based on the <entity, context pattern> relations extracted from a given corpus. For example, an <entity, context pattern> entry could be <Moscow, went to *>.

To effectively leverage the high-order information between entities and patterns, we construct an entity-pattern bipartite graph using all <entity, context pattern> entries extracted from the corpus as the input to the BootstrapNet (see Figure 2(a)). In this graph, each entity is linked to the patterns associating with it in the corpus. Therefore, by exploiting multi-hop paths in this bipartite graph, we can obtain the high-order relations between entities and patterns, which is effective in aggregating evidence for evaluating entities and patterns. Formally, our entity-pattern bipartite graph is defined as a tuple $G =< V, E, L >$, where:

1. $V = V_n \cup V_p$ consists of a set of entity nodes (V_n) and a set of pattern nodes (V_p);
2. E is the set of edges that link an entity node to a pattern node if the pattern occurs around the entity;
3. L is the set of labels of seed entities.

BootstrapNet

This section describes the proposed BootstrapNet—an end-to-end bootstrapping neural network for ESE, which employs the encoder-decoder architecture. Specifically, the BootstrapNet contains the following two main components:

- **BootstrapEncoder** is a graph neural network-based encoder which takes the bipartite graph as the input, and encodes the first- and the high-order relations between entities and patterns into the learned node representations.
- **BootstrapDecoder** is an RNN-based decoder which takes the above node representations as the inputs, sequentially generates new entities, and updates the entity category representations.

Figure 2(b) illustrates the framework of BootstrapNet. Components within BootstrapNet are introduced in detail in the following sections.

BootstrapEncoder

Given the constructed bipartite graph, the BootstrapEncoder captures the correlations between entities and patterns, and encodes the correlation information as node representations. The node representations are then fed into the BootstrapDecoder to generate new entities.

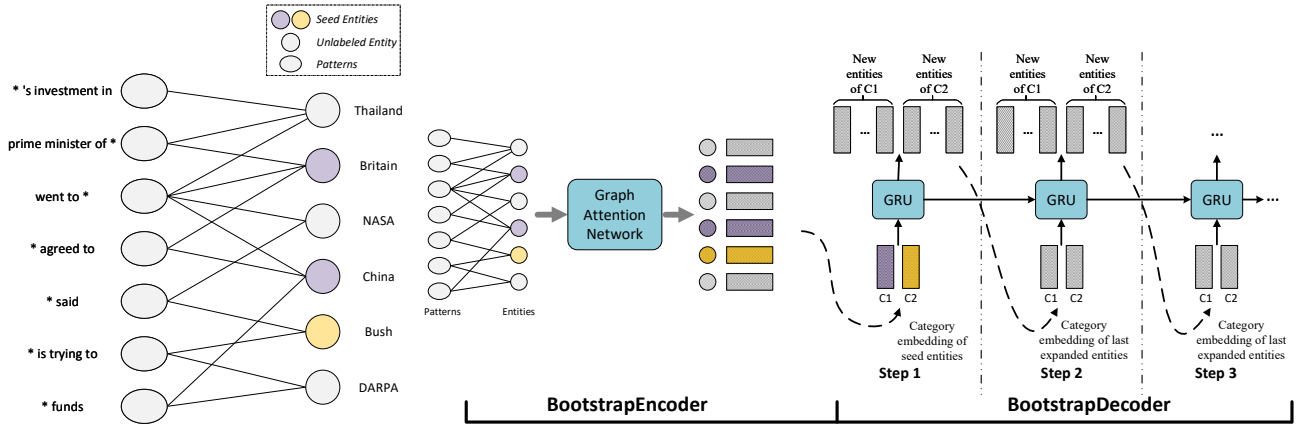
To this end, we use a multi-layer graph attention network (Veličković et al. 2018), which takes the constructed bipartite graph as the input and updates node representations by aggregating neighboring representations from the previous layer.

Specifically, for each node v_i, the BootstrapEncoder learns its l-th layer representation using two parts. The first gets the information from its representation in the previous layer, i.e.,

$$\alpha_i^l = W_v^l s_i^{l-1} \tag{1}$$

where W_v^l is the projection matrix to be learned, and s_i^{l-1} is the node representation in layer $l-1$ (For s_i^0, we encode each node in the bipartite graph using an initial representation described in the end of this subsection). The second aggregates the information from its neighboring nodes:

$$\beta_i^l = \sum_{j \in \mathcal{N}(i)} a_{i,j}^l W_v^l s_j^{l-1} \tag{2}$$



(a) An example of entity-pattern bipartite graph. Circles in different colors are the seeds belonging to the different categories.

(b) Overall framework of BootstrapNet.

Figure 2: The input and the overall framework of BootstrapNet.

where $\mathcal{N}(i)$ is the set of neighboring nodes of node v_i, $a_{i,j}$ is the normalized weight computed by the attention mechanism.

Consequently, the final updated representation of node v_i in the l-th layer is the combination of α_i^l and β_i^l as follows:

$$s_i^l = \sigma(\alpha_i^l + \beta_i^l) \tag{3}$$

where σ is a non-linear activation function, e.g., the ReLU. By recursively aggregating the information from neighboring nodes, the encoder is capable of capturing multi-hop information contained in the graph and therefore can exploit high-order relations for entity set expansion.

Attention. The normalized weight in Eq. (2) is crucial for filtering out noises when aggregating neighboring information since not all context patterns are representative to the semantics of an entity. For example, "*prime minister of* *" is a strong indicator for GPE, while the general pattern "* *wants to*" is a weak indicator of an entity's category. Therefore, we compute the normalized weight using the following self-attention score, for a node v_i:

$$\begin{aligned} a_{i,j}^l &= \frac{\exp(e_{i,j}^l)}{\sum_{k\in\mathcal{N}(i)}\exp(e_{i,k}^l)} \\ e_{i,j}^l &= f(W_a^l h_i^l, W_a^l h_j^l) \end{aligned} \tag{4}$$

where f is a shared attention mechanism function $f : \mathbb{R}^F \times \mathbb{R}^F \mapsto \mathbb{R}$ to compute attention coefficients, W_a^l is the projection matrix to be learned.

Initial representation. In this paper, we initialize the representations of entities/patterns using the average embeddings of tokens in them. Because both entities and patterns are short *n*-grams in our experiments, average token embedding is an effective initialization[1]. We use the pre-trained GloVe (Pennington, Socher, and Manning 2014) to embed the tokens.

[1]For long patterns and entities, other contextualized initialization can be applied, e.g., BERT (Devlin et al. 2019). We herein adopt average embedding for its simplicity and equivalent performance according to our pilot experiment results.

BootstrapDecoder

After encoding each node using the BootstrapEncoder, the BootstrapDecoder exploits a recurrent neural network to sequentially expand new entities, and the learned seed representations are used as its initial inputs.

Specifically, we regard the entity expansion process as a sequential entity generation process, and Gated Recurrent Unit (GRU) (Cho et al. 2014) is used as the sequential generation model. To capture the semantics of the target category and update it during the bootstrapping process, the GRU's hidden state is used to represent the entity categories. At each step, our GRU decoder takes previous expansion results as the inputs to update the hidden state (i.e., category representations) and generates new entities as the outputs according to their similarities to the updated hidden state.

Formally, the expansion of new entities at each bootstrapping step contains two stages—the category representation updating and the entity generation. At the category representation updating stage, the BootstrapDecoder takes the last expanded entity embeddings as the inputs and updates the hidden state (i.e., the category embeddings) as follows:

$$\begin{aligned} z_c^t &= \sigma(W_z \cdot s_c^t + U_z \cdot h_c^{t-1}) \\ r_c^t &= \sigma(W_r \cdot s_c^t + U_r \cdot h_c^{t-1}) \\ \bar{h}_c^t &= \sigma(W \cdot s_c^t + r_c^t \cdot U \cdot h_c^{t-1}) \\ h_c^t &= z_c^t \odot h_c^{t-1} + (1 - z_c^t) \odot \bar{h}_c^t \end{aligned} \tag{5}$$

where h_c^{t-1} is the updated hidden state vector of category c after the step $t-1$, and s_c^t is the average embedding of the expanded entities belonging to category c in the step $t-1$(we set s_c^t to an all-zero vector if there is no entity belonging to c). The initial hidden state is set to all-zero.

At the entity generation stage, the BootstrapDecoder generates new entities for each category based on the cosine

similarity of the unexpanded entities to the updated category representations. Specifically, at the t-th decoding step, an unexpanded entity v_i is labeled as the category with the highest probability computed as:

$$p(c|i) = \frac{g(c,i)}{\sum_{c' \in C} g(c',i)}$$
$$g(c,i) = 0.5 + 0.5 * sim(h_c^t, s_i) \quad (6)$$

where $sim(h_c^t, s_i)$ is the cosine similarity of category c's updated representation h_c^t and an entity s_i, C is the set of all categories, $g(*)$ is a normalization function to scale the cosine similarity to $[0, 1]$. After that, for those entities labeled as the same category, only the top N entities with the highest probabilities are expanded.

Model Learning

One challenge for our BootstrapNet is how to train it using a few seed entities as the initial supervision. Because the seed entities are fed as the first-step input of the BootstrapDecoder, there is no other supervision for evaluating BootstrapDecoder's successive outputs. Therefore, most semi-supervised learning algorithms on graph neural network are not applicable to our model.

To address the above issues, we propose to optimize our BootstrapNet with a multi-view based algorithm inspired by Qu, Bengio, and Tang (2019). Specifically, we first construct an auxiliary neural network, named BootstrapTeacher, which shares the same encoder architecture with the BootstrapNet, but directly connects the encoder to a multilayer perceptrons (MLP) classifier to classify each entity. The BootstrapTeacher does not consider the long-term dependencies between entity expansion decisions. Therefore, the BootstrapTeacher and the BootstrapNet can be regarded as two different views of entity set expansion: the BootstrapNet models a sequential expanding process considering the influence of current expansion on successive expansion; while the BootstrapTeacher models a non-sequential expanding process in which the classification results depend only on the individual entity representations.

Specifically, we design an iterative model learning algorithm containing two phases (see Alg. 1) as follows:

- **BootstrapTeacher learning phase** learns the BootstrapTeacher parameters using the seed entities and the labeled expanded entities by BootstrapNet.
- **BootstrapNet learning phase** learns the BootstrapNet parameters using labeled entities returned by BootstrapTeacher.

In the following, we explain the two phases in detail.

BootstrapTeacher learning phase. To learn the BootstrapTeacher, we use both the seed entities and the expanded entities returned by BootstrapNet as the supervision, and let the BootstrapTeacher maximize the following objective:

$$O_S = \sum_{n \in L} \log q(y_n|x_n) + \sum_{n' \in L'} \log q(\hat{y}_{n'}|x_{n'}) \quad (7)$$

where $q(*|x)$ is the category distribution predicted by BootstrapTeacher, L is the set of seed entities, y_n is the label of a seed entity, L' is the set of expanded entities returned by the BootstrapNet, and $\hat{y}_{n'}$ is the label of an expanded entity returned by the BootstrapNet. At the beginning of model learning, L' is set to be empty.

Algorithm 1 Optimization Algorithm

Input: A bipartite graph G, and seed entities with category labels (L, y_L)
Output: Expanded entities for each category

```
Construct BootstrapTeacher with the BootstrapEncoder followed by an MLP classifier
Pre-train BootstrapTeacher according to Eq. (9)
while convergence criteria not met do
    Copy Encoder's parameters in BootstrapTeacher to BootstrapNet
    Annotate unlabeled entities with BootstrapTeacher
    Update BootstrapNet with Eq. (10)
    Expand new entities with labels using BootstrapNet
    Update BootstrapTeacher with Eq. (9)
end while
Expand new entities using the learned BootstrapNet.
```

Because the seed entities are sparse comparing to the number of unexpanded entities, the learned BootstrapTeacher tends to be underfitting. Inspired by Zupon et al. (2019), and based on the intuition that the pattern and entity embeddings are similar to their neighbors but dissimilar to their unrelated patterns or entities, we leverage the graph structure as a regularizer to the learning procedure, and let the BootstrapTeacher maximize the following unsupervised learning objective:

$$O_U = \sum_e [\sum_{e^-} \log(\sigma(-V_e^T V_{e^-})) + \sum_{p^+} \log(\sigma(V_e^T V_{p^+})) + \sum_{p^-} \log(\sigma(-V_e^T V_{p^-}))] \quad (8)$$

where e is an entity node in the graph, e^- is the entity multi-hop away from e, p^+ is the pattern directly linked to entity e, p^- is the pattern multi-hop away from e in the graph. By adding Eq. 7 and 8, we obtain the overall objective:

$$O_{teacher} = \gamma O_S + (1 - \gamma) O_U \quad (9)$$

BootstrapNet learning phase. After optimizing the BootstrapTeacher, all unlabeled entities are labeled using the BootstrapTeacher. Then the BootstrapNet is trained on these pseudo-labeled unexpanded entities by maximizing the following objective:

$$O_{net} = \sum_{t=1}^{T} \sum_{n \in Node(t)} \log p(\bar{y}_n|x_n) \quad (10)$$

where T is the maximal decoding step of BootstrapNet, $Node(t)$ is the set of expanded entities at the t-th decoding step, $p(*|x_n)$ is the category distribution for node n returned by the BootstrapNet, $\bar{y}_n$ is the node n's label returned

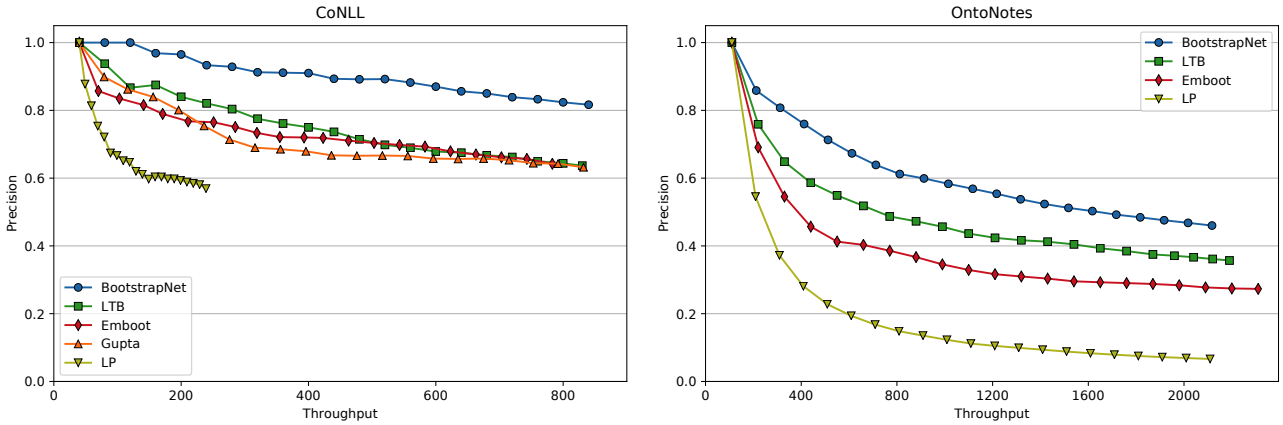


Figure 3: Overall results on CoNLL and OntoNotes.

by BootstrapTeacher. To accelerate the optimization process, the encoder of BootstrapNet will share the same parameters with that of BootstrapTeacher after the BootstrapTeacher learning phase, and it will not be tuned during the BootstrapNet learning phase.

After the BootstrapNet learning phase, BootstrapNet expands a set of entities with their categories as labels, which are used in the subsequent BootstrapTeacher learning phase.

Experiments

Experimental Setup

Datasets: We use two datasets, CoNLL and OntoNotes, constructed by Zupon et al. (2019). CoNLL is constructed from the CoNLL 2003 shared task dataset (Tjong Kim Sang and De Meulder 2003), which contains 4 entity types. OntoNotes is constructed from the OntoNotes datasets (Pradhan et al. 2013) without numerical categories, which finally contains 11 entity types. Zupon et al. (2019) use the *n*-grams of the size up to 4 tokens on either side of an entity as the patterns and filter out some patterns. Table 1 demonstrates the basic statistics of these two datasets [2].

Dataset	# Categories	# Entities	# Patterns	# Links
CoNLL	4	5,522	8,477	13,916
OntoNotes	11	19,984	33,985	67,229

Table 1: Dataset statistics.

Baselines. In this paper, we use the following methods as our baselines:

- **LP**: this is the label propagation method which uses the seed entities as labeled ones and co-occurrence counts of the entities and patterns as the entity features, and expands top entities with the lowest entropy at each iteration[3].
- **Gupta** (Gupta and Manning 2014): this method is a classical bootstrapping system that iteratively evaluates and selects patterns, and scores new entities by a learned entity classifier[4].
- **Emboot** (Zupon et al. 2019): this method follows Gupta and Manning (2014), but learns custom embeddings for entities and patterns at each iteration, which are used to guide the entity classifier.
- **LTB** (Yan et al. 2019): this method uses the MCTS algorithm to perform lookahead search for estimating delayed feedback and jointly learns an entity similarity function.

Settings. To evaluate these methods, we report the cumulative precision and throughput of expanded entities (The seed entities are treated as the initial throughput). For all baselines and our model, we manually select 10 seeds per category with the highest frequency in the datasets and run them for 20 bootstrapping iterations. At each bootstrapping iteration, we add 10 entities and 10 patterns to each category. The number of layers in BootstrapEncoder is set to 3.

To learn our model, we randomly select other 30 entities per category with their labels from each dataset as the development set, and leave the remaining entities as the test set. Source code is available online[5].

Overall Results

Figure 3 shows the overall results on CoNLL and OntoNotes. From this figure, we can see that:

- **BootstrapNet significantly outperforms baselines on two datasets**. On both CoNLL and OntoNotes, our proposed BootstrapNet can expand entities with higher preci-

[2]There are other larger datasets available. Due to the limited scalability of implementing graph neural networks on large scale graphs, we leave the experimentation on these datasets for our future work.

[3]LP is implemented using the scikit-learn package (https://scikit-learn.org/stable/modules/label_propagation.html).

[4]Due to the fact that the labels of its builtin Named Entity classifier do not match the labels in the OntoNotes, we do not run this method on the OntoNotes.

[5]https://github.com/lingyongyan/bootstrapnet

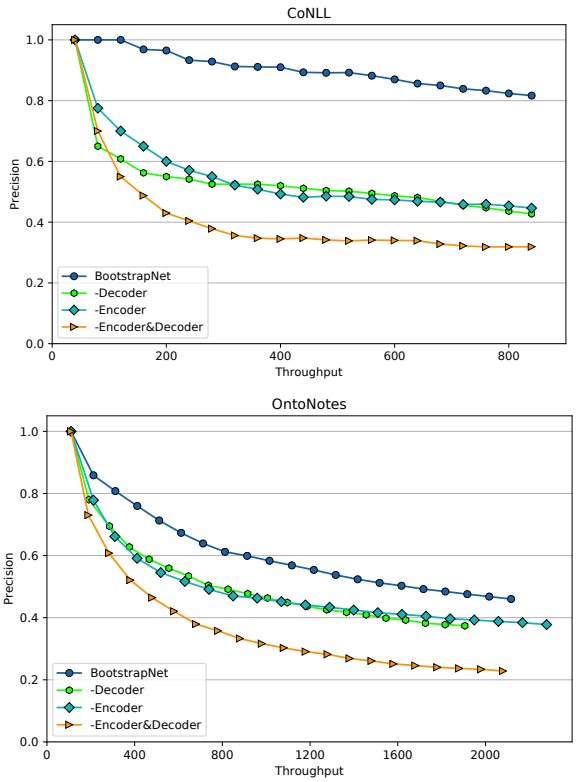


Figure 4: An ablation study of the BootstrapNet.

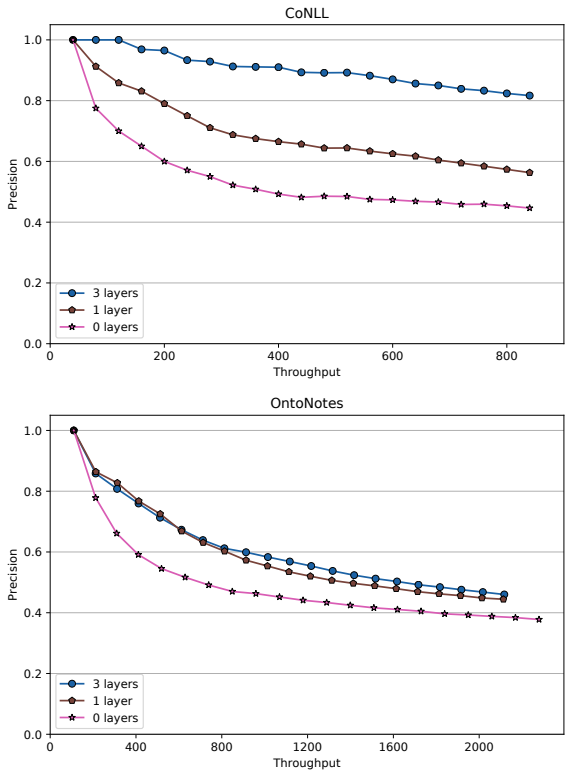


Figure 5: The performance of BootstrapNet with different numbers of layers in the BootstrapEncoder.

sion comparing to the baselines. Specifically, on CoNLL, the BootstrapNet expands 800 new entities with the precision higher than 80% after 20 iterations, while the precision of all baselines is less than 70%; on OntoNotes, the BootstrapNet expands around 2,000 new entities with the precision higher than 45% after 20 iterations, while the precision of all baselines is less than 40%.

- **BootstrapNet can significantly reduce the semantic drift problem in the bootstrapping technique**. Comparing to the baselines, the precision curve of the BootstrapNet decreases more smoothly with the increases of the throughput on both CoNLL and OntoNotes; while the baselines usually decrease drastically at the very beginning stages, especially on the OntoNotes dataset. The different decreasing curve slopes indicate that our model can introduce less noisy entities when performing more bootstrapping iterations and consequently reduce the semantic drift problem.

Detail Analysis

Ablation study of BootstrapNet. To further analyze the contribution of the encoder and the decoder to the final performance, we conduct ablation study on the two datasets (see Figure 4), where "-Encoder" denotes directly using initial node representations (i.e., GloVe) as the final node representations, "-Decoder" denotes iteratively selecting top entities that are most similar to seed entities' representation without RNN; "-Encoder&Decoder" means disabling both components. From Figure 4, we can find that the both components have considerable contributions to the final performance. Specifically, the precision of BootstrapNet decreases without the BootstrapEncoder or the BootstrapDecoder; and the performance of BootstrapNet further degrades if both components are dropped. The results indicate that the BootstrapEncoder for capturing the first- and the high-order relations and the BootstrapDecoder for modeling a sequential expansion process are both effective and beneficial to the final performance of our BootstrapNet.

Method	CoNLL		OntoNotes	
	10 Iters	20 Iters	10 Iters	20 Iters
GloVe	.882/400	.808/800	.521/1007	.431/2007
Random	.405/400	.385/800	.226/897	.206/1579

Table 2: Expansion quality comparison for two node initialization methods. "GloVe": initializing using GloVe vectors. "Random": randomly initializing. We report the precision (before the slash) and numbers of expanded entities (after the slash) at expansion iteration 10 and 20 (We only report the precision of expanded entities excluding the seeds).

Comparison of different layers of BootstrapEncoder. We further study the influence of layer numbers on our model's performance (see Figure 5)("0 layers" indicates directly using initial node representation as final node repre-

sentations, which is the same as the "-Encoder"). Among these methods, "0 layers" captures no relations between entities and patterns; "1 layer" captures the first-order relations between entities and patterns; "3 layers" captures both the first- and the high-order relations. From Figure 5, we can see that the performance of the BootstrapNet increases with more layers. Besides, we can also see that "3 layers" outperforms "1 layer" on the CoNLL, which indicates that capturing more high-order relations between entities and patterns can significantly improve the performance comparing to only using the first-order relations. On OntoNotes, "3 layers" slightly outperforms "1 layer", which may be mainly due to the unbalanced category distribution in OntoNotes makes the learning a bit underfitting [6].

Comparison of different node representation initialization methods. To study the importance of pre-trained embeddings to the BootstrapNet's final performance, we conduct another experiment with randomly initialized embeddings. The randomly initialized embeddings are only learned during the pre-training stage in alg. 1 and fixed during the following learning phases; to fully learn the randomly initialized embeddings, we double the pre-training epochs while keeping all other settings unchanged. The final results are shown in Table 2. We can see that the initialization methods influence the overall performance a lot on both CoNLL and OntoNotes. Specifically, the precision of randomly initialized embeddings drops to the half of GloVe initialization on both datasets; on OntoNotes, the expansion numbers also decrease when using the randomly initialized embeddings. Because there is only sparse supervision provided, the pre-trained GloVe vectors can provide some extra information for better model learning. The lack of extra supervision is likely to lead to aggregation of noisy information, which may cause the entities belonging to similar but different categories to have similar representations. As a result, some categories containing fewer entities are more likely to drift to other categories, which decreases the expansion throughput and precision.

Related Work

Bootstrapping is the algorithm that starts from a small set of seed instances and expands to more instances. Due to its minimal requirement for supervision signals, it has been widely applied in information extraction (Riloff and Shepherd 1997; Qadir et al. 2015; Gupta, Roth, and Schütze 2018), as well as word disambiguation (Yoshida et al. 2010), entity translation (Lee and Hwang 2013), model learning (Whitney and Sarkar 2012), etc.

Early bootstrapping methods (Riloff and Shepherd 1997; Collins and Singer 1999) on information extraction evaluate and select patterns and new instances mainly based on instance-pattern first-order relations (i.e., instance-pattern matching statistics) without consider the entire bootstrapping process, which can easily lead to semantic drifting problem (Curran, Murphy, and Scholz 2007). To reduce the semantic drifts, most previous studies focus on adding extra constraints or features, e.g., mutual exclusive bootstrapping (Curran, Murphy, and Scholz 2007; McIntosh and Curran 2008; Gupta, Roth, and Schütze 2018), negative seeds (Yangarber, Lin, and Grishman 2002; Shi et al. 2014), coupling constraints (Carlson et al. 2010), lexical and statistical features (Gupta and Manning 2014), cosine similarity over one-hot vectors (Liao and Grishman 2010), word embeddings (Batista, Martins, and Silva 2015; Gupta and Manning 2015), etc. Some other studies focus on detecting drift points (i.e., patterns and instances) and stopping expanding detected drift points (Li et al. 2014; 2018). Besides, similar to our method, some studies using the bipartite graph and reduce the semantic drift using graph-based methods (Komachi et al. 2008; Tao et al. 2015). However, these methods still lack the high-order information or require extra manual participants. Recently, (Berger et al. 2018) try to address the sparse supervision problem using active learning strategy to acquire supervision signals from human; (Zupon et al. 2019) leverage entity-pattern co-occurrence information to learn the custom embedding beyond sparse supervision; Yan et al. (2019) estimate delayed feedback (which is the high-order relation) for bootstrapping using the Monte Carlo Tree Search algorithm. However, all these methods model the bootstrapping process in a pipelined and separate paradigm.

Graph Neural Network (Gori, Monfardini, and Scarselli 2005) is another related work that can aggregate high-order information on the graph structure. The graph neural network in our model is closely related to graph convolutional networks(GCN) (Kipf and Welling 2017) and its variant—graph attention network(GAT) (Veličković et al. 2018). In NLP, Marcheggiani and Titov (2017) and Bastings et al. (2017) use the GCN to encode dependency parsing tree as syntax information for semantic role labeling and machine translation respectively; Beck, Haffari, and Cohn (2018) introduce the gate mechanism to GCN for learning a graph-to-sequence model; recently, Guo, Zhang, and Lu (2019) leverage the GAT to capture syntax information for the relation extraction.

Conclusions

In this paper, we propose BootstrapNet, an end-to-end neural network which can model the bootstrapping process in an encoder-decoder architecture. Specifically, we use a graph attention network as the encoder to capture both the first- and the high-order relations between entities and patterns, and a recurrent neural network-based decoder is used to sequentially expand new entities and update category representations. A multi-view learning algorithm is further proposed to effectively learn our BootstrapNet using sparse supervision signals. Experiments show that our BootstrapNet outperforms state-of-the-art methods.

For future work, since the input of our model is a constructed bipartite graph, our method can be easily adapted to other information extraction tasks. For example, if we use the *<head entity, tail entity>* pair as the instance and the context around it as the pattern, we can construct an instance-pattern bipartite graph and extract new relations using our BootstrapNet.

[6]There are two categories of entities (i.e., "LAW" and "LANGUAGE") whose numbers are less than 200.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grants no. 61433015, 61572477 and 61772505, and Beijing Academy of Artificial Intelligence (BAAI).

References

Bastings, J.; Titov, I.; Aziz, W.; Marcheggiani, D.; and Sima'an, K. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. In *Proceedings of EMNLP*, 1957–1967.

Batista, D. S.; Martins, B.; and Silva, M. J. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of EMNLP*, 499–504.

Beck, D.; Haffari, G.; and Cohn, T. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks. In *Proceedings of ACL*, 273–283.

Berger, M.; Nagesh, A.; Levine, J.; Surdeanu, M.; and Zhang, H. 2018. Visual Supervision in Bootstrapped Information Extraction. In *Proceedings of EMNLP*, 2043–2053.

Carlson, A.; Betteridge, J.; Wang, R. C.; Hruschka, Jr., E. R.; and Mitchell, T. M. 2010. Coupled Semi-supervised Learning for Information Extraction. In *Proceedings of WSDM*, 101–110.

Chen, J.; Ji, D.; Tan, C. L.; and Niu, Z. 2006. Relation Extraction Using Label Propagation Based Semi-supervised Learning. In *Proceedings of COLING/ACL*, 129–136.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*, 1724–1734.

Collins, M., and Singer, Y. 1999. Unsupervised Models for Named Entity Classification. In *EMNLP*.

Curran, J. R.; Murphy, T.; and Scholz, B. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of PACLING*, volume 6, 172–180. Citeseer.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*, 4171–4186.

Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A new model for learning in graph domains. In *Proceedings of IJCNN*, 729–734.

Guo, Z.; Zhang, Y.; and Lu, W. 2019. Attention Guided Graph Convolutional Networks for Relation Extraction. In *Proceedings of ACL*, 241–251.

Gupta, S., and Manning, C. 2014. Improved Pattern Learning for Bootstrapped Entity Extraction. In *Proceedings of CoNLL*, 98–108.

Gupta, S., and Manning, C. D. 2015. Distributed Representations of Words to Guide Bootstrapped Entity Classifiers. In *Proceedings of NAACL-HLT*, 1215–1220.

Gupta, P.; Roth, B.; and Schütze, H. 2018. Joint Bootstrapping Machines for High Confidence Relation Extraction. In *Proceedings of NAACL-HLT*, 26–36.

Kipf, T. N., and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Komachi, M.; Kudo, T.; Shimbo, M.; and Matsumoto, Y. 2008. Graph-based analysis of semantic drift in Espresso-like bootstrapping algorithms. In *Proceedings of EMNLP*, 1011–1020.

Lee, T., and Hwang, S.-w. 2013. Bootstrapping Entity Translation on Weakly Comparable Corpora. In *Proceedings of ACL*, 631–640.

Li, Z.; Li, H.; Wang, H.; Yang, Y.; Zhang, X.; and Zhou, X. 2014. Overcoming Semantic Drift in Information Extraction. In *Proceedings of EDBT*, 169–180. OpenProceedings.org.

Li, Z.; He, Y.; Gu, B.; Liu, A.; Li, H.; Wang, H.; and Zhou, X. 2018. Diagnosing and Minimizing Semantic Drift in Iterative Bootstrapping Extraction. *TKDE* 30(5):852–865.

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

Marcheggiani, D., and Titov, I. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of EMNLP*, 1506–1515.

McIntosh, T., and Curran, J. R. 2008. Weighted Mutual Exclusion Bootstrapping for Domain Independent Lexicon and Template Acquisition. In *Proceedings of ALTA*, 97–105.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of EMNLP*, 1532–1543.

Pradhan, S.; Moschitti, A.; Xue, N.; Ng, H. T.; Björkelund, A.; Uryupina, O.; Zhang, Y.; and Zhong, Z. 2013. Towards Robust Linguistic Analysis using OntoNotes. In *Proceedings of CoNLL*, 143–152.

Qadir, A.; Mendes, P. N.; Gruhl, D.; and Lewis, N. 2015. Semantic Lexicon Induction from Twitter with Pattern Relatedness and Flexible Term Length. In *Proceedings of AAAI*, 2432–2439.

Qu, M.; Bengio, Y.; and Tang, J. 2019. GMNN: Graph Markov Neural Networks. In *ICML*, 5241–5250.

Riedel, S.; Yao, L.; McCallum, A.; and Marlin, B. M. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. In *Proceedings of NAACL-HLT*, 74–84.

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

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

Shi, B.; Zhang, Z.; Sun, L.; and Han, X. 2014. A probabilistic co-bootstrapping method for entity set expansion. In *Proceedings of COLING*, 2280–2290.

Tao, F.; Zhao, B.; Fuxman, A.; Li, Y.; and Han, J. 2015. Leveraging Pattern Semantics for Extracting Entities in Enterprises. In *Proceedings of WWW*, 1078–1088.

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

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Whitney, M., and Sarkar, A. 2012. Bootstrapping via Graph Propagation. In *Proceedings of ACL*, 620–628.

Yan, L.; Han, X.; Sun, L.; and He, B. 2019. Learning to bootstrap for entity set expansion. In *Proceedings of EMNLP*, 292–301.

Yangarber, R.; Lin, W.; and Grishman, R. 2002. Unsupervised Learning of Generalized Names. In *Proceedings of COLING*, 1–7.

Yoshida, M.; Ikeda, M.; Ono, S.; Sato, I.; and Nakagawa, H. 2010. Person Name Disambiguation by Bootstrapping. In *Proceedings of SIGIR*, 10–17. ACM.

Zupon, A.; Alexeeva, M.; Valenzuela-Escárcega, M.; Nagesh, A.; and Surdeanu, M. 2019. Lightly-supervised Representation Learning with Global Interpretability. In *Proceedings of the Third Workshop on Structured Prediction for NLP*, 18–28.