Learning to Bootstrap for Entity Set Expansion

Lingyong Yan[1,3], **Xianpei Han**[1,2,*], **Le Sun**[1,2], **Ben He**[3,1,*]
[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, sunle}@iscas.ac.cn, benhe@ucas.ac.cn

Abstract

Bootstrapping for Entity Set Expansion (ESE) aims at iteratively acquiring new instances of a specific target category. Traditional bootstrapping methods often suffer from two problems: 1) *delayed feedback*, i.e., the pattern evaluation relies on both its direct extraction quality and the extraction quality in later iterations. 2) *sparse supervision*, i.e., only few seed entities are used as the supervision. To address the above two problems, we propose a novel bootstrapping method combining the Monte Carlo Tree Search (MCTS) algorithm with a deep similarity network, which can efficiently estimate delayed feedback for pattern evaluation and adaptively score entities given sparse supervision signals. Experimental results confirm the effectiveness of the proposed method.

1 Introduction

Bootstrapping is widely used for Entity Set Expansion (ESE), which acquires new instances of a specific category by iteratively evaluating and selecting patterns, while extracting and scoring entities. For example, given seeds {*London, Paris, Beijing*} for capital entity expansion, a bootstrapping system for ESE iteratively selects effective patterns, e.g., "*the US Embassy in* *", and extracts new capital entities, e.g., *Moscow*, using the selected patterns.

The main challenges of effective bootstrapping for ESE owe to the *delayed feedback* and the *sparse supervision*. Firstly, bootstrapping is an iterative process, where noises brought by currently selected patterns can affect successive iterations (Movshovitz-Attias and Cohen, 2012; Qadir et al., 2015). Indeed, the pattern evaluation relies on not only its direct extraction quality but also the extraction quality in later iterations, which are correspondingly denoted as *instant feedback* and *delayed feedback* in this paper. For instance, as shown in Figure 1, although "* *is a big city*" and "*the US Embassy in* *" have equal direct extraction quality, the former is considered less useful since its later extracted entities are mostly unrelated. As a result, selecting patterns with high instant feedback but low delayed feedback can cause semantic drift problem (Curran et al., 2007), where the later extracted entities belong to other categories. Secondly, the above difficulty is further compounded by sparse supervision, i.e., using only seed entities as supervision, since it provides little evidence for deciding whether an extracted entity belongs to the same category of seed entities or not.

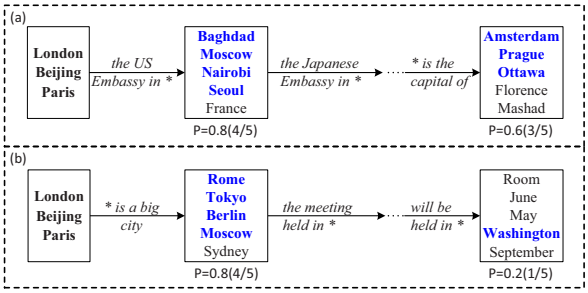


Figure 1: Example of the delayed feedback problem, which expands the capital seeds {*London, Paris, Beijing*}. We demonstrate the top entities extracted by a pattern (correct ones are shown in blue), where P is the precision of extracted entities.

To address the above two challenges, we propose a novel bootstrapping method combining the Monte Carlo Tree Search (MCTS) algorithm with a deep similarity network, aiming to effectively evaluate the delayed feedback of patterns and adaptively score entities given sparse supervision signals. Specifically, our method tackles the delayed feedback problem by enhancing the traditional bootstrapping method using the MCTS algorithm, which effectively estimates each pattern's delayed feedback via efficient multi-step lookahead search. In this way, our method can select the pattern based on its delayed feedback

*Corresponding authors.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 292–301, Hong Kong, China, November 3–7, 2019.

rather than instant feedback, which is beneficial in that the former feedback is regarded more reliable and accurate for bootstrapping. To resolve the sparse supervision problem, we propose a deep similarity network—pattern mover similarity network (PMSN), which uniformly embeds entities and patterns using the distribution vectors on context pattern embeddings, and measures their semantic similarity to seeds as their ranking scores based on those embeddings; furthermore, we combine the PMSN with the MCTS, and fine-tune the distribution vectors using the estimated delayed feedback. In this way, our method can adaptively embed and score entities.

Major contributions of this paper are tri-fold.

- Enhanced bootstrapping via the MCTS algorithm to estimate delayed feedback in bootstrapping. To our best knowledge, this is the first work to combine bootstrapping with MCTS for entity set expansion.
- A novel deep similarity network to evaluate different categories of entities in the bootstrapping for Entity Set Expansion.
- Adaptive entity scoring by combining the deep similarity network with MCTS.

2 Background

Traditional bootstrapping systems for ESE are usually provided with sparse supervision, i.e., only a few seed entities, and iteratively extract new entities from corpus by performing the following steps, as demonstrated in Figure 2(a).

Pattern generation. Given seed entities and the extracted entities (known entities), a bootstrapping system for ESE firstly generates patterns from the corpus. In this paper, we use lexicon-syntactic surface words around known entities as patterns.

Pattern evaluation. This step evaluates generated patterns using sparse supervision and other sources of evidence, e.g., pattern embedding similarity. Many previous studies (Riloff and Jones, 1999; Curran et al., 2007; Gupta and Manning, 2014) use the RlogF function or its variants to evaluate patterns, which usually estimate the instant feedback of each pattern.

Entity expansion. This step selects top patterns to match new candidate entities from the corpus.

Entity scoring. This step scores candidate entities using sparse supervision, bootstrapping or other external sources of evidence. The top scored entities are then added to the extracted entity set.

As aforementioned, traditional bootstrapping systems for ESE do not consider the delayed feedback when evaluating patterns, leaving considerable potential in further improvement. To estimate the delayed feedback of a pattern, a simple solution is to perform lookahead search for a fixed number of steps and estimate the quality of its successive extracted entities. However, lookahead search may suffer from efficiency issue brought by the enormous search space since there are many candidate patterns at each step. Besides, due to the sparse supervision problem, entity scoring is often unreliable, which in turn influences the delayed feedback estimation.

To address the above two problems, this paper enhances the bootstrapping system using the Monte Carlo Tree Search (MCTS) algorithm for lookahead search, combined with a pattern mover similarity network (PMSN) for better entity scoring. Specifically, we use MCTS for the efficient lookahead search by pruning bad patterns. We additionally estimate the delayed feedback using the PMSN to score entities, given the sparse supervision signals, and fine-tune the PMSN using the delayed feedback estimated by MCTS. In this way, both the MCTS algorithm and the PMSN are devised to enhance each other, resulting in efficient delayed feedback estimation for pattern evaluation and reliable entity scoring.

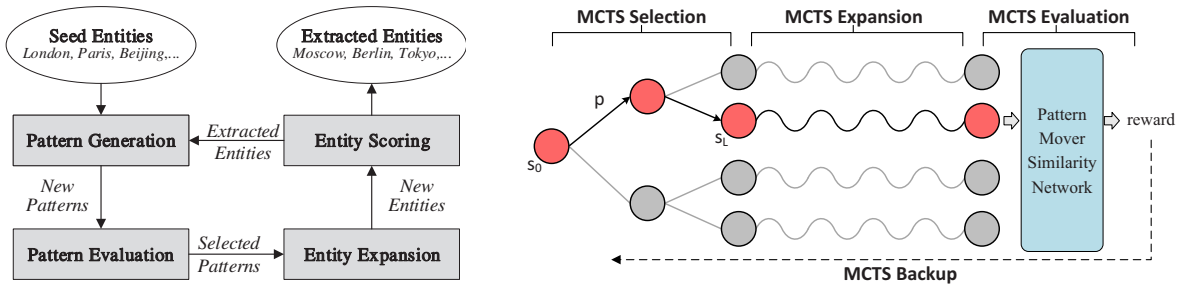
3 Enhancing Bootstrapping via Monte Carlo Tree Search

In this section, we describe how to enhance the traditional bootstrapping for ESE using the MCTS algorithm for efficient delayed feedback estimation.

3.1 Efficient Delayed Feedback Estimation via MCTS

To estimate the delayed feedback, this paper uses the Monte Carlo Tree Search (MCTS) algorithm for efficient lookahead search.

At each bootstrapping iteration, the MCTS algorithm performs multiple lookahead search procedures (MCTS simulations). Starting from the same root node, each MCTS simulation performs forward search by iteratively constructing sub-nodes, and moving to one of these sub-nodes. Therefore, the whole MCTS algorithm looks like a tree structure (see Figure 2(b)), where the node s represents for known entities, i.e., both seed entities and previously extracted entities, and the edge,



(a) Traditional bootstrapping system for Entity set expansion. The main difference between our method and traditional methods is that we enhance the **pattern evaluation** by the MCTS algorithm, as demonstrated in Figure (b).

(b) The Monte Carlo Tree Search for the pattern evaluation in a bootstrapping system for entity set expansion. The red circles refer to the search node (i.e., seed entities plus extracted entities). A unidirectional edge represents the selection of a pattern to match new entities, which results in a new node.

Figure 2: Traditional Bootstrapping for ESE (a) and the enhanced pattern evaluation using MCTS (b).

linking from a node s to one of its sub-nodes s', represents for selecting a pattern p to expand s. At the very beginning, the root tree node s_0 is constructed using seed entities and the extracted entities from previous iterations, and is fixed among different simulations. Besides, for each (s, p) pair, we store a cumulative reward $Q(s, p)$ and a visit count $N(s, p)$ during tree search for the subsequent reward function defined in Section 3.3.

Specifically, each MCTS simulation contains four stages, as demonstrated in Figure 2(b):

Selection. Starting from s_0, each simulation first traverses the MCTS search tree until reaching the leaf node s_L (which is never reached before) or reaching a fixed depth. Each traversing step i corresponding to selecting a pattern p^i by:

$$p^i = \arg\max_p Q(s, p) + \mu(s, p) \tag{1}$$

where $\mu(s, p) \propto \frac{p_\sigma(s,p)}{1+N(s,p)}$, $p_\sigma(s, p)$ is the prior probability of p returned by the policy network p_σ, which is described in detail in Section 3.2.

Among an MCTS simulation, the lookahead search space is reduced mainly in this stage. Specifically, according to e.q. (1), the lookahead search is more likely to select the potential patterns with high cumulative rewards or high prior probabilities, rather than all patterns.

Expansion. When the traversal reaches a leaf node s_L, we expand this node by selecting a new pattern to match more entities. Due to the lack of the cumulative rewards on the new patterns, we select the new pattern with the highest prior probability returned by p_σ.

Evaluation. Once finishing the above expansion stage or the simulation procedure reaches a fixed depth, the reward R for the leaf node is returned by first quickly selecting multiple patterns to expand the leaf node and then evaluating the quality of all newly extracted entities in this simulation. We herein use the RlogF function to quickly select patterns rather than the policy network (Running RlogF function is much faster than the policy network). And the reward function is described in Section 3.3 in detail.

Backup. At this stage, the returned reward is used to update the cumulative rewards and visit counts of previous (s, p) pairs by:

$$\begin{aligned} N(s, p) &= \sum_j^n \mathbf{1}(s, p, j) \\ Q(s, p) &= \frac{1}{N(s, p)} \sum_{j=1}^n \mathbf{1}(s, p, j) \cdot R \end{aligned} \tag{2}$$

where $\mathbf{1}(s, p, j)$ indicates whether an edge (s, p) was traversed during the j^{th} simulation.

After finishing all MCTS simulations, we use the cumulative reward of each (s_0, p) pair as the delayed feedback for pattern p. Because the cumulative rewards are updated many times using the quality evaluation of their future extracted entities, the cumulative reward of each (s_0, p) pair can be regarded as the precise approximation of the delayed feedback if we have a proper reward function. As a result, our method selects the top patterns with the highest cumulative rewards, which are more likely to extract correct entities.

3.2 Prior Policy using Pattern Mover Similarity Network

The prior policy network is mainly used to prune bad patterns and therefore reduce the search space in the MCTS. Intuitively, if a pattern is not similar

to other patterns around some entities, it is likely a general pattern or noisy pattern to these entities, and therefore should be pruned.

To this end, this paper proposes a novel deep similarity network, namely Pattern Mover Similarity Network (PMSN), which uniformly embeds patterns, entities and entity sets, and estimates their embedding similarities. We describe this model in detail in Section 4.

Particularly, for a pattern p linked from a tree node s, the PMSN is used as the prior policy network to compute the prior probability by:

$$p_\sigma(s, p) = \frac{\text{SIM}(p, E)}{\sum_{p'} \text{SIM}(p', E)} \quad (3)$$

where E is the set of known entities included in node s, and $\text{SIM}(p, E)$ is the similarity of the pattern p to the entity set E using the PMSN.

To further reduce the search space (note that the number of patterns at each step can be tens of thousands), we use the RlogF function to retain only the top k for lookahead search. In the experiments, we set k to a reasonably large value, i.e., 200, to balance between efficiency and effectiveness.

3.3 Reward Function in MCTS

The reward function is critical for efficiently estimating the real delayed feedback of each pattern. Intuitively, a pattern should have higher delayed feedback if it extracts more similar entities and less unrelated entities. Base on this intuition, we devise the reward function as follows:

$$R = \frac{\sum_{e \in E'} \text{SIM}(e, E_0)}{|E'|} \sigma(\frac{|E'|}{a}) \quad (4)$$

where E_0 is the set of known entities in root node, E' is the set of new entities, $\text{SIM}(e, E)$ is the similarity score of newly extracted entity e to known entities, $\sigma(\cdot)$ is the sigmoid function, and a is a "temperature" hyperparameter. The above reward function can be regarded as a trade-off between the extraction quality (the first term) and the extraction amount (the second term). To compute the similarity score, we also exploit the PMSN.

4 Pattern Mover Similarity Network

The pattern mover similarity network (PMSN) is a unified model for adaptively scoring the similarity of entities or patterns to seed entities. Specifically, the pattern mover similarity network contains two components: 1) the adaptive distributional pattern embeddings (ADPE) that adaptively represent patterns, entities, and entity sets in a unified way; 2) the pattern mover similarity (PMS) measurement that estimates the similarity of two ADPEs.

The PMSN model is mainly used in three aspects as follows. 1) The PMSN is used as the prior policy network in the MCTS algorithm to evaluate the similarity of patterns. 2) The PMSN is used to evaluate the newly extracted entities within the MCTS simulation, whose evaluation scores are subsequently used to update rewards. 3) The PMSN is also used as the entity scoring function at the Entity Scoring stage in the bootstrapping process as mentioned in Section 2.

4.1 Adaptive Distributional Pattern Embeddings

In this section, we first describe how to embed patterns; then, we introduce how to obtain the basic distributional pattern embeddings for uniformly representing entities and patterns without adaptation; finally, we introduce the adaptation mechanism combined with the MCTS algorithm.

Pattern Embedding. As a basic step of our PMSN model, we first embed a context pattern of an entity as a single embedding vector. Specifically, we use the average word embeddings, from the pre-trained GloVe (Pennington et al., 2014) embeddings, of a pattern surface text as the pattern embeddings. We filter out patterns containing at least two OOV terms. According to our pilot experiments, replacing the average GloVe word embeddings with alternatives such as Convolutional Neural Network and Recurrent Neural Network does not influence performance.

Basic distributional pattern embeddings without adaptation. Based on the single embedding of each context pattern, this paper represents an entity using a distributional vector on its context pattern embeddings, called distributional pattern embeddings (DPE). The intuition behind is that each context pattern represents one aspect of meanings of an entity according to the distributional hypothesis in linguistics (Harris, 1954), while the importance of different patterns to the semantic of an entity varies from each other. Therefore, we use a distributional vector, which stores the importance score of different patterns, together with the context pattern embeddings to represent the semantic of an entity.

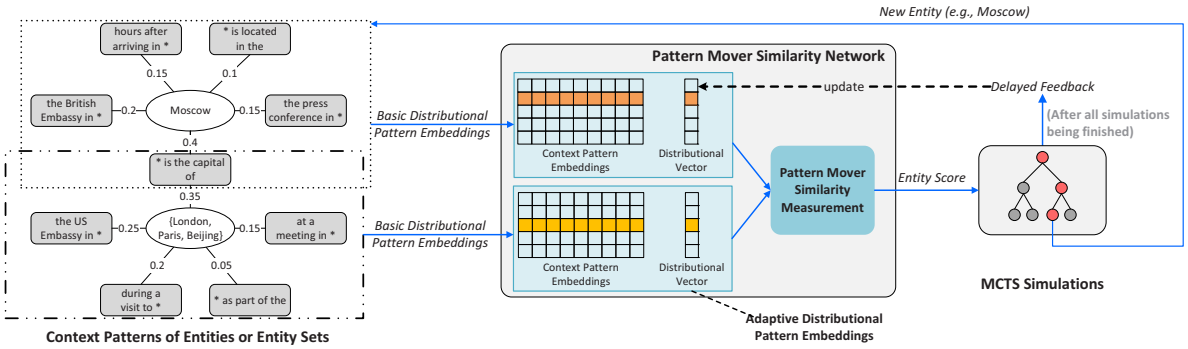


Figure 3: Overall architecture of the Pattern Mover Similarity Network (PMSN), as shown in the rounded square in the middle of the figure.

To estimate each pattern's importance score, we suggest that a context pattern is important to the semantic of an entity if: the pattern matches the entity frequently; the pattern matches as few other entities as possible. Therefore, we design a importance score function for each context pattern p of an entity e as follows:

$$w(p,e) = \frac{N(e,p) \times \log N(e,p)}{C(p)} \tag{5}$$

where $C(p)$ is the number of different entities matched by p, $N(e,p)$ is the frequency of p_i matching entity e. Ultimately, the above importance scores are mapped to the distributional probabilities of each entity's context pattern such that all probabilities sum up to 1.

In addition, we estimate the basic DPE for a single pattern and a whole entity set. Specifically, a single pattern can be regarded as a special "entity", whose context pattern is only the pattern itself. Similarly, as shown in Figure 3, an entity set can be regarded as another special "entity", whose context patterns are the union of all context patterns of entities in this set. Besides, the importance score function for context patterns of an entity set is slightly different from the one in e.q. (5):

$$w(p,E) = \frac{N(E,p) \times \log N(E,p)}{C(p)} \tag{6}$$

where E is the entity set, $N(E,p)$ is the frequency of p matching all entities in E.

Finally, we denote the DPE as $< X, w >$, where X is the context pattern embedding matrix, and w is the vector of distribution probabilities. For efficiency, we only select the top n important patterns. Therefore, X is actually an $n \times d$ matrix, where d is the dimension of each pattern embedding, and w is an n-dimensional vector.

Adaptive distributional pattern embeddings combined with MCTS. Although the basic DPE can provide unified representations for both patterns and entities, it could still fail to represent the underlying semantics of seed entities as unrelated patterns may match many other entities. To address this problem, we combine the MCTS algorithm with the PMSN, to fine-tune the distributional vector of the basic DPE, resulting in an Adaptive Distributional Pattern Embedding (ADPE). As shown in Figure 3, at each iteration, multiple MCTS simulations are performed to accumulate the long-term feedback of selecting a pattern, where the PMSN is used for entity scoring; after that, the delayed feedback can be efficiently calculated for each pattern. Apart from being used to evaluate the reward of selecting a pattern, the delayed feedback can also be used to estimate the importance score of a pattern, since patterns with the high delayed feedback can extract more correct entities and less incorrect entities in the future. Therefore, at each iteration, we fine-tune the distributional probabilities after the MCTS simulations as following:

$$w_t(p,e) \propto w_{t-1}(p,e) \cdot Q(s_0,p) \tag{7}$$

where $w_{t-1}(p,E)$ is the probability of p at iteration $t-1$, $Q(s_0,p)$ is returned cumulative reward of p_i in iteration t.

4.2 Pattern Mover Similarity

The similarity measurement for two ADPEs is important, since it is used for delayed feedback estimation and entity scoring. Intuitively, given two entities embedded by two ADPEs, they can be regarded similarly to each other, if they have similar context patterns and similar distributions on these patterns. Therefore, the similarity measurement should take both context pattern embeddings

and corresponding distributional vectors into consideration. Inspired by Kusner et al. (2015) on the sentence similarity measurement, we devise a similarity measurement for two ADPEs as follows:

$$\begin{aligned} \max_{T \geq 0} \quad & \sum_{i,j=1}^{n} T_{ij} \cdot \text{SIM}(i,j) \\ \text{s.t.} \quad & \sum_{j=1}^{n} T_{ij} = w_i, \quad \forall i \in 1,...,n \\ & \sum_{i=1}^{n} T_{ij} = w_j, \quad \forall j \in 1,...,n \end{aligned} \tag{8}$$

where $\text{SIM}(i,j)$ is the cosine similarity between the i-th pattern embedding of one entity and the j-th pattern embedding of the other entity. We denote the above measurement as the Pattern Mover Similarity (PMS) measurement.

5 Experiments

5.1 Experimental Settings

Category	Description	Category	Description
CAP	Capital name	**FAC**	Man-made structures
ELE	Chemical element	**ORG**	Organizations
FEM	Female first name	**GPE**	Geo-political entities
MALE	Male first name	**LOC**	Non-GPE locations
LAST	Last name	**DAT**	A date or period
TTL	Honorific title	**LANG**	Any named language
NORP	Nationality, Religion, Political		

Table 1: Target categories used on Google Web 1T.

Datasets. We conduct experiments on three public datasets: Google Web 1T (Brants and Franz, 2006), APR, and Wiki (Shen et al., 2017). 1) Google Web 1T contains a large scale of n-grams compiled from a one trillion words corpus. Following Shi et al. (2014), we use 5-grams as the entity context and filter out those 5-grams containing all stopwords or common words. We use 13 categories of entities (see Table 1) list in Shi et al. (2014) and compare our method with traditional bootstrapping methods for ESE on this corpus. 2) APR (2015 news from AP and Reuters) and Wiki (a subset of English Wikipedia) are two datasets published by Shen et al. (2017). Each of them contains about 1 million sentences. We use totally 12 categories of entities as listed in Shen et al. (2017) and compare the final entity scoring performance on both datasets.

Baselines. To evaluate the efficiency of the MCTS and the PMSN, we use several baselines:

1) POS: bootstrapping method which only uses positive seeds without any other constraint;

2) MEB(Curran et al., 2007): mutual exclusion bootstrapping method, which uses the category exclusion as the constraints of bootstrapping;

3) COB(Shi et al., 2014): a probabilistic bootstrapping method which uses both positive and negative seeds.

4) SetExpan(Shen et al., 2017): corpus-based entity set expansion method, which adaptively selects context features and unsupervisedly ensembles them to score entities.

Specifically, we compare baselines (1)-(3) and our method on Google Web 1T; we compare baseline (4) and our method on APR and Wiki [1].

Metrics. We use P@n (precision at top n), and the mean average precision (MAP) on Google Web 1T as in Shi et al. (2014). As for the APR and the Wiki , we use MAP@n (n=10,20,50) to evaluate entity scoring performance of our method. In our experiments, we manually select frequent entities as the seeds from these datasets; the correctness of all extracted entities is manually judged with external supporting resources, e.g., the entity list collected from Wikipedia[2].

5.2 Experimental Results

Comparison with three baseline methods on Google Web 1T. Table 2 shows the performance of different bootstrapping methods on Google Web 1T. We can see that our full model outperforms three baseline methods: comparing with POS, our method achieves 41% improvement in P@100, 35% improvement in P@200 and 45% improvement in MAP; comparing with MEB, our method achieves 24% improvement in P@100 and 18% improvement in P@200; comparing with COB, our method achieves 3% improvement in both P@100 and P@200 metrics, and 2% improvement in MAP. The above findings indicate that our method can extract more correct entities with higher ranking scores than the baselines.

Comparison with SetExpan on APR and Wiki. To further verify that our method can learn a better representation and adaptively score entities in ESE, we compare our method with the state-

[1]Due to scalability issue, we omit the result of SetExpan on Google Web 1T. On the two smaller datasets, results of the under-performing baselines (1-3) are also omitted for space reason. The incompetitive performance of those 3 baselines could be caused by the difficulty in deriving robust statistical features out of the sparse context patterns on the small ARP and Wiki datasets.

[2]The code is released at https://www.github.com/lingyongyan/mcts-bootstrapping.

Method	P@10	P@20	P@50	P@100	P@200	MAP
POS	0.84	0.74	0.55	0.41	0.34	0.42
MEB	0.83	0.79	0.68	0.58	0.51	-
COB*	0.97	0.96	0.90	0.79	0.66	0.85
Ours^{full}	**0.97**	**0.96**	**0.92**	**0.82**	**0.69**	**0.87**
Ours^{-MCTS}	0.85	0.81	0.73	0.63	0.52	0.75
Ours^{-PMSN}	0.63	0.60	0.56	0.48	0.42	0.61

Table 2: Overall results for entity set expansion on Google Web 1T,where Ours^{full} is the full version of our method, Ours^{-MCTS} is our method with the MCTS disabled, and Ours^{-PMSN} is our method but replacing the PMSN with fixed word embeddings. * indicates COB using the human feedback for seed entity selection.

Method	APR			Wiki		
	MAP@10	MAP@20	MAP@50	MAP@10	MAP@20	MAP@50
SetExpan	0.90	0.86	0.79	0.96	0.90	0.75
Ours	**0.96**	**0.90**	**0.81**	**0.98**	**0.93**	**0.79**

Table 3: The adaptive entity scoring performance of different methods on the APR and Wiki.

Category	**P@20**	**P@50**	**P@100**	**P@200**
CAP	1.00	1.00	0.94	0.69
ELE	1.00	0.84	0.51	0.36
FEN	1.00	1.00	1.00	0.95
MALE	1.00	1.00	1.00	0.98
LAST	1.00	1.00	1.00	1.00
TTL	0.85	0.64	0.49	0.34
NORP	0.95	0.96	0.89	0.60
FAC	0.95	0.86	0.59	0.42
ORG	1.00	1.00	1.00	0.94
GPE	1.00	1.00	1.00	0.94
LOC	0.80	0.76	0.70	0.67
DAT	1.00	1.00	0.82	0.56
LANG	1.00	0.98	0.81	0.52

Table 4: The performance of our *full* method on different categories on Google Web 1T.

of-the-art entity set expansion method–SetExpan, which is a non-bootstrapping method on the APR and Wik (see Table 3). From Table 3, we can see that our method outperforms SetExpan on both datasets: on the APR, our method achieves 6% improvement in MAP@10 and 2% improvement in MAP@50 ; on the Wiki, our method can achieve 2% improvement in MAP@10 and 4% improvement in MAP@50. The above results further confirm that our method can improve the performance of bootstrapping for Entity Set Expansion.

5.3 Detailed Analysis

Comparison with the Ours^{-MCTS} method and the Ours^{-PMSN} method. From Table 2, we can also see that if we replace the Monte Carlo Tree Search by selecting top-n patterns, the performance decreases by 19% in P@100 and 17% in P@200; if we replace the PMSN with word embedding, the performance decreases by 34% in P@100 and 27% in P@200. The results show that both the PMSN and the MCTS algorithm are critical for our model's performance. Remarkably, the PMSN and the MCTS algorithms can enhance each other in that the PMSN can learn a better representation by combing with the MCTS algorithm, and the MCTS can in turn effectively estimate delayed feedback using the PMSN.

Performance of our full method on different categories of Google Web 1T. From Table 4, We can see that our method achieves high performance in most categories except for ELE, TTL and FAC entities. The lesser performance of our method on ELE entities is likely caused by the data sparseness — there are fewer than 150 elements of the ELE entities in total. The lower performance on TTL and FAC entities is likely due to the fact that the context patterns of TTL and FAC entities are similar to those of person and location names respectively, which makes them easily be regarded as special person names and location names respectively.

Influence of the number of context patterns. Figure 4 shows the performance of our full method under different context pattern numbers. It can be seen that the number of context patterns used to embed entity and entity set heavily influences the performance. Comparing with the settings using fewer context patterns (e.g., 10 and 20, 50), using more context patterns, i.e., 100, in the PMSN has superior performance since it can provide more context information to compare two entities. Besides, further adding context patterns in the PMSN causes the performance decrease for 3%, which is

Iters	Patterns selected by Ours^{full}	Patterns selected by Ours^{-MCTS}	Patterns selected by Ours^{-PMSN}
1	Embassy of Sweden in *	held a meeting in *	* and New York in
2	Embassy of Belgium in *	* meeting was held on	in New York or *
3	's capital city of *	* Meeting to be held	between New York and *
4	* is the capital city	* held its first meeting	* hotel reservations with discount
5	* was the capital of	* meeting to be held	* is a great city

Table 5: The top patterns selected by different methods when expanding capital entities in the first 5 iterations.

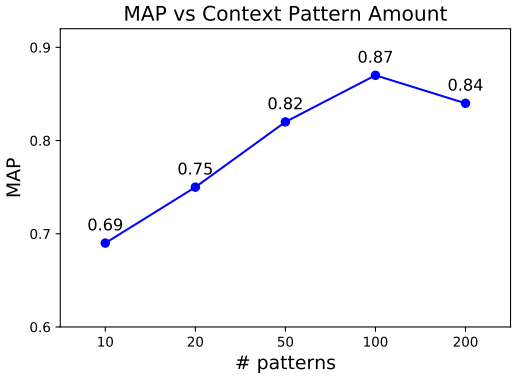


Figure 4: Performance of our *full* method using different context patterns in the PMSN.

likely due to the case that noises can be included when considering too many patterns.

Top patterns selected by different methods. To demonstrate the effectiveness of delayed feedback in our method, we illustrate the top 1 pattern in the first five iterations of three methods in Table 5. From Table 5, we can see that the top patterns by our *full* method are more related to seed entities than other two baselines. Besides, we can see that without the MCTS algorithm or the PMSN, most top patterns are less related and easily semantically drifted to other categories.

6 Related Work

Entity set expansion (ESE) is a weakly supervised task, which is often given seed entities as supervision and tries to expand new entities related to them. According to the used corpus, there are two types of ESE: limited corpus (Shi et al., 2014; Shen et al., 2017) and large open corpus, e.g. making use of a search engine for the web search (Wang and Cohen, 2007).

Weakly supervised methods for information extraction (IE) are often provided insufficient supervision signals, such as knowledge base facts as distant supervision (Mintz et al., 2009; Hoffmann et al., 2011; Zeng et al., 2015; Han and Sun, 2016), and light amount of supervision samples in bootstrapping(Riloff and Jones, 1999). As a classical technique, bootstrapping usually exploits pattern (Curran et al., 2007), document (Liao and Grishman, 2010) or syntactic and semantic contextual features (He and Grishman, 2015) to extract and classify new instances.

Limited to the sparse supervision, previous work estimate patterns mainly based on its direct extraction features, e.g., the matching statistics with known entities (Riloff and Jones, 1999; Agichtein and Gravano, 2000), which often suffers from the semantic drift problem. To avoid semantic drift, most existing approaches exploit extra constraints, such as parallel multiple categories (Thelen and Riloff, 2002; Yangarber, 2003; McIntosh, 2010), coupling constraints (Carlson et al., 2010), and mutual exclusion bootstrapping (Curran et al., 2007; McIntosh and Curran, 2008). Besides, graph-based methods (Li et al., 2011; Tao et al., 2015) and the probability-based method (Shi et al., 2014) are also used to improve the bootstrapping performance.

To address the *sparse supervision* problem, many previous studies score entities by leveraging lexical and statistical features (Yangarber et al., 2000; Stevenson and Greenwood, 2005; Pantel and Pennacchiotti, 2006; Paşca, 2007; Pantel et al., 2009), which, despite the promising effectiveness, could often fail since the sparse statistical features provide little semantic information to evaluate entities. Recently word embedding based methods (Batista et al., 2015; Gupta and Manning, 2015) use fixed word embedding learned on external resources and evaluate entities by their similarity to seeds. Recently, Berger et al. (2018) propose to learn custom embeddings at each bootstrapping iteration, to trade efficiency for effectiveness.

7 Conclusions

In this paper, we propose a deep similarity network-based model combined with the MCTS algorithm to bootstrap Entity Set Expansion. Specifically, we leverage the Monte Carlo Tree Search (MCTS) algorithm to efficiently estimate the delayed feedback of each pattern in the bootstrapping; we propose a Pattern Mover Similarity Network (PMSN) to uniformly embed entities

and patterns using a distribution on context pattern embeddings; we combine the MCTS and the PMSN to adaptively learn a better embedding for evaluating both patterns and entities. Experimental results confirm the superior performance of our PMSN combined with the MCTS algorithm.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. 61433015, 61572477 and 61772505; the Young Elite Scientists Sponsorship Program no. YESS20160177; and University of Chinese Academy of Sciences.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting Relations from Large Plain-text Collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 85–94, NY, USA.

David S. Batista, Bruno Martins, and Mário J. Silva. 2015. Semi-Supervised Bootstrapping of Relationship Extractors with Distributional Semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 499–504, Lisbon, Portugal. Association for Computational Linguistics.

Matthew Berger, Ajay Nagesh, Joshua Levine, Mihai Surdeanu, and Helen Zhang. 2018. Visual Supervision in Bootstrapped Information Extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2043–2053, Brussels, Belgium. Association for Computational Linguistics.

Thorsten Brants and Alex Franz. 2006. The google web 1t 5-gram corpus version 1.1. *LDC2006T13*.

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 the Third ACM International Conference on Web Search and Data Mining*, pages 101–110, NY, USA.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, volume 6, pages 172–180. Citeseer.

Sonal Gupta and Christopher Manning. 2014. Improved Pattern Learning for Bootstrapped Entity Extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 98–108, Ann Arbor, Michigan. Association for Computational Linguistics.

Sonal Gupta and Christopher D. Manning. 2015. Distributed Representations of Words to Guide Bootstrapped Entity Classifiers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1215–1220, Denver, Colorado. Association for Computational Linguistics.

Xianpei Han and Le Sun. 2016. Global Distant Supervision for Relation Extraction. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Yifan He and Ralph Grishman. 2015. ICE: Rapid Information Extraction Customization for NLP Novices. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 31–35, Denver, Colorado. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based Weak Supervision for Information Extraction of Overlapping Relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 957–966, Lille, France.

Haibo Li, Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2011. Using Graph Based Method to Improve Bootstrapping Relation Extraction. In *Computational Linguistics and Intelligent Text Processing*, volume 6609, pages 127–138. Springer Berlin Heidelberg, Berlin, Heidelberg.

Shasha Liao and Ralph Grishman. 2010. Filtered Ranking for Bootstrapping in Event Extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 680–688, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tara McIntosh. 2010. Unsupervised Discovery of Negative Categories in Lexicon Bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 356–365, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tara McIntosh and James R. Curran. 2008. Weighted Mutual Exclusion Bootstrapping for Domain Independent Lexicon and Template Acquisition. In *Proceedings of the Australasian Language Technology*

Association Workshop 2008, pages 97–105, Hobart, Australia.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant Supervision for Relation Extraction Without Labeled Data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dana Movshovitz-Attias and William W. Cohen. 2012. Bootstrapping Biomedical Ontologies for Scientific Text Using NELL. pages 11–19.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale Distributional Similarity and Entity Set Expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, pages 938–947, Stroudsburg, PA, USA. Association for Computational Linguistics.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 113–120, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marius Paşca. 2007. Weakly-supervised Discovery of Named Entities Using Web Search Queries. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, pages 683–690, New York, NY, USA. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Ashequl Qadir, Pablo N Mendes, Daniel Gruhl, and Neal Lewis. 2015. Semantic Lexicon Induction from Twitter with Pattern Relatedness and Flexible Term Length. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2432–2439.

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

Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. SetExpan: Corpus-Based Set Expansion via Context Feature Selection and Rank Ensemble. In *ECML PKDD*, volume 10534, pages 288–304, Cham. Springer International Publishing.

Bei Shi, Zhenzhong Zhang, Le Sun, and Xianpei Han. 2014. A probabilistic co-bootstrapping method for entity set expansion. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 2280–2290.

Mark Stevenson and Mark A. Greenwood. 2005. A Semantic Approach to IE Pattern Induction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 379–386, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fangbo Tao, Bo Zhao, Ariel Fuxman, Yang Li, and Jiawei Han. 2015. Leveraging Pattern Semantics for Extracting Entities in Enterprises. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1078–1088, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Michael Thelen and Ellen Riloff. 2002. A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pattern Contexts. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, pages 214–221, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference On*, pages 342–350. IEEE.

Roman Yangarber. 2003. Counter-training in Discovery of Semantic Patterns. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 343–350, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, pages 940–946, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal. Association for Computational Linguistics.