Learning to Bootstrap Entity Set Expansion via Deep Similarity Network and Monte Carlo Tree Search

Anonymous ACL submission

Abstract

Bootstrapping for Entity Set Expansion (ESE) aims at iteratively acquiring new instances of a specific target category. However, traditional bootstrapping ESE often suffers from two problems: 1) *delayed feedback*, where the pattern evaluation relies on both its direct extraction quality and extraction quality in later iterations; 2) *sparse supervision*, where the supervision signals are often a few seed entities, which makes entity scoring difficult. To address the above two problems, we propose a deep similarity network-based bootstrapping method combined with the Monte Carlo Tree Search (MCTS) algorithm, which can efficiently estimate delayed feedback for pattern evaluation and adaptively score entities given sparse supervision signals. Experimental results verified the effectiveness of our method.

1 Introduction

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

The main challenges of effective bootstrapping for ESE owe to the *delayed feedback* and the *sparse supervision*. Firstly, bootstrapping is an iterative process, where noisy inclusion by currently selected patterns can affect successive iterations (Movshovitz-Attias and Cohen, 2012; Qadir et al., 2015); conversely, the pattern evaluation relies on not only its direct extraction quality but also the extraction quality in later iterations, which are called instant feedback and delayed feedback correspondingly in this paper. For instance, as shown in Figure 1, although "* *is a big city*" and "*the US Embassy in* *" have the same direct extraction quality, the former is still worse than the latter since its later extracted entities are mostly unrelated. Selecting patterns with high instant feedback but low delayed feedback can cause semantic drift problem (Curran et al., 2007), where extracted entities in later iterations 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 to decide whether an extracted entity belongs to the same category or not.

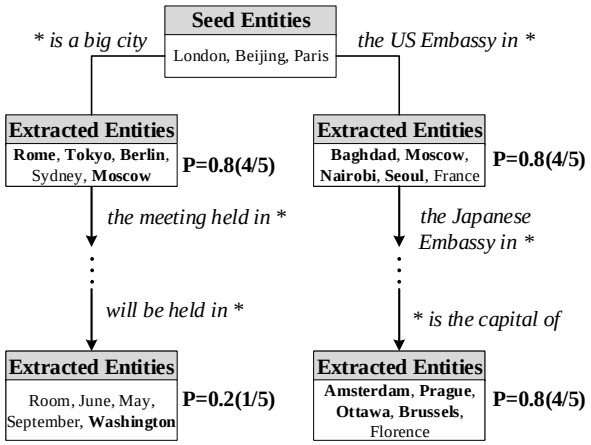


Figure 1: Example of two patterns "* *is a big city*" and "*the US Embassy in* *" used for expanding capital seeds, where two patterns have the same instant feedback but different delayed feedback. We demonstrate top entities extracted by a pattern (true ones are shown in bold) due to the page limitation; P is the precision of extracted entities.

Currently, most previous studies evaluate patterns mainly based on its direct extraction features, e.g., the matching statistics with known entities (Riloff and Jones, 1999). To avoid semantic drifting, most of them exploit extra constraints, such as parallel multiple categories (The-

len and Riloff, 2002; Yangarber, 2003), negative samples (Yangarber et al., 2002; McIntosh, 2010; Shi et al., 2014), and mutual exclusion bootstrapping (Curran et al., 2007; McIntosh and Curran, 2008). These constraints can also be regard as undirected delayed feedback, since patterns violating constraints often have low delayed feedback. To address the *sparse supervision* challenge, most previous studies score entities by leveraging statistical features inner bootstrapping process (Riloff and Jones, 1999; Stevenson and Greenwood, 2005; Pantel and Pennacchiotti, 2006; McIntosh and Curran, 2008; Pantel et al., 2009), which often fails since sparse statistical features provide little semantic information to evaluate entities; some recent studies use fixed word embeddings as external resources and evaluate entities by their similarity to seeds. (Batista et al., 2015; Gupta and Manning, 2015); however, the fixed embeddings cannot preciously represent underground semantics of seed entities in given corpus since they are often pre-trained on other corpus; Recently, Berger et al. (2018) learn custom embeddings through the bootstrapping process, but its learning cost is still too much.

In this paper, we propose a deep similarity network-based bootstrapping method, which can adaptively evaluate patterns and score entities by combining Monte Carlo Tree Search (MCTS) algorithm. Specifically, our method solves 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 rather than instant feedback, which is 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 pattern by an adaptive distribution on context pattern embeddings, and measures their semantic similarity to seed using a pattern mover similarity measurement. We combine the PMSN with the MCTS, and fine-tune the distribution using estimated delayed feedback. In this way, our method can adaptively embed and score entities in corpus.

The contributions of our work are:

1) We enhance the traditional bootstrapping via the MCTS algorithm to estimate delayed feedback in bootstrapping. To our best knowledge, this is the first time to combine bootstrapping with the MCTS algorithm.
2) We propose a novel deep similarity network, which can adaptively evaluate different categories of entities in Entity Set Expansion.

This paper is organized as follows: In Section 2, we describe how to enhance bootstrapping via the MCTS algorithm. In Section 3, we describe our pattern mover similarity network in detail. Section 4 describes the experiments. Section 5 briefly reviews related work and Section 6 concludes this paper.

2 Enhancing Bootstrapping via Monte Carlo Tree Search

In this section, we describe how to enhance traditional bootstrapping ESE using the MCTS algorithm. By using the MCTS algorithm, our method can estimate the delayed feedback for each pattern through multi-step lookahead search and select the patterns with the highest delayed feedback.

2.1 Traditional Bootstrapping

A traditional bootstrapping ESE system is usually provided sparse supervision, i.e., a few seed entities, and iteratively extracts new entities from corpus by iteratively performing the following steps (see Figure 2a):

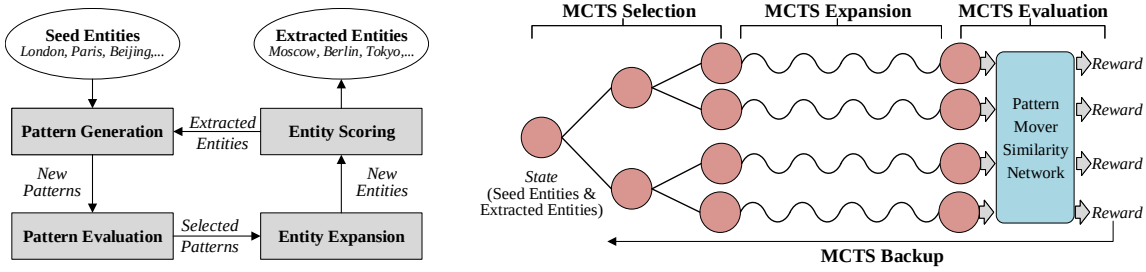
Pattern generation. Given seed entities and extracted entities to current (known entities), a bootstrapping ESE system first generates patterns from the corpus. In this paper, we use lexicon-syntactic surface words around known entities as the patterns.

Pattern evaluation. This step evaluates generated patterns using sparse supervision and other evidence. Many previous studies (Riloff and Jones, 1999; Thelen and Riloff, 2002; Curran et al., 2007; Gupta and Manning, 2014) use the RlogF function or its variants to evaluate patterns.

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 evidences or other external resources. And the top entities will be added to the extracted entity set.

From the above steps, we can see that it is difficult for traditional methods to estimate the delayed feedback in the pattern evaluation step, since it cannot estimate the delayed feedback of a pat-



(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 Monte Carlo Tree Search algorithm (see b in left)

(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 states (seed entities plus extracted entities) in the MCTS process. Edges linked from a left circle to a right circle represent to select a pattern to match new entities from the left state, which results in a new state.

Figure 2: Traditional Bootstrapping for Entity Set Expansion (a) and enhancing pattern evaluation using the MCTS algorithm(b).

tern unless already selecting it. Therefore, to efficiently estimate the delayed feedback, we enhance the pattern evaluation step using the classical lookahead search algorithm—Monte Carlo Tree Search, which estimates the delayed feedback of each pattern by multi-step lookahead search. In this way, our method can directly select patterns with the highest delayed feedback.

2.2 Enhancing Pattern Evaluation via MCTS

In this subsection, we describe how to estimate delayed feedback of each pattern using the MCTS algorithm.

Formally, given seed entity set E_s and the extracted entity set E to current, our method will build a search tree by first constructing a root state as $s_0 = E_s \cup E$. Starting from the root state, our method performs several times simulations to estimate the delayed feedback of patterns; each simulation contains several steps, and each step refers to selecting one pattern to match new entities; once we obtain new entities after one step, we add them to the extracted entity set to get a new entity set E', and build a new state as $s' = E_s \cup E'$. Besides, we build an edge (s, p) linking from state s to state s', which means selecting pattern p from state s results in state s'. To estimate final delayed feedback, each edge stores an action value $Q(s, p)$ and a visit count $N(s, p)$ during tree search. Specifically, each MCTS simulation in our method will perform the following four steps (see Figure 2b):

Selection. Starting from the root state s_0, our method performs one simulation by traversing the MCTS search tree until the leaf state (which is never reached before) or reaching fixed depth. Specifically, at each step $i(i > 0)$ of this simulation, we select a new action p^i from state s by:

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

$$\mu(s, p) \propto \frac{p_\sigma(s, p)}{1 + N(s, p)} \quad (2)$$

where $p_\sigma(s, p)$ is the prior probability of p returned by the policy network p_σ, which will be described in detail below.

Expansion. When the traversal reaches a leaf state at step L, current state s_L is expanded, where a legal action will be selected and expand new entities to build a new state. We select the action based on each pattern's prior probability returned by the policy network p_σ.

Evaluation. Once finishing expanding the leaf state to a new state or the search process reaches a certain depth, our method will quickly perform several steps using the RlogF function (which replace the policy network for the quick pattern selection); then, we evaluate the quality of final extracted new entities and return it as the reward R of this simulation.

Backup. At the end of each simulation, we use the reward to update action values and visit counts of all <state, pattern> pairs as:

$$N(s, p) = \sum_j^n 1(s, p, j) \quad (3)$$

$$Q(s, p) = \frac{1}{N(s, p)} \sum_{j=1}^{n} 1(s, p, j) \cdot R \quad (4)$$

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

After finishing all MCTS simulations, the action values of all <state, pattern> pairs (s_0, p), can indicate the estimated delayed feedback of each pattern at current iteration. Therefore, if the reward function can preciously evaluate the extraction quality, the action values can be preciously estimated by the MCTS algorithm. And our method selects the top patterns with the highest action values and is more likely to extract reliable new entities.

Prior Policy using Pattern Mover Similarity Network

A promising prior policy network is critical in MCTS simulations, since there are many candidate patterns at each step, which makes final search space extremely large. To prune bad patterns, we use the pattern mover similarity network as the prior policy network.

To give a prior evaluation of patterns, one way is to exploit statistical features-based functions, e.g., the RlogF function, like many traditional methods (Riloff and Jones, 1999; Pantel and Pennacchiotti, 2006; McIntosh and Curran, 2008; Gupta and Manning, 2014). However, these functions are mainly designed for directly bootstrapping rather than the MCTS simulations; in addition, since there is sparse supervision provided, statistical features are often unreliable and depend heavily on the seed selection. Another way is to exploit embedding-based models like some recent studies (Batista et al., 2015; Berger et al., 2018). Embedding-based models can leverage external resources, i.e., pre-trained wording embeddings, to overcome the sparse supervision problems.

In this paper, we also exploit the embedding-based model as our policy network. Particularly, we propose a unified deep similarity network called pattern mover similarity network, which are correspondingly used as the prior policy network when evaluating patterns and used in the reward function when evaluating entities. We will describe this model in detail in Section 3.

Specifically, to assign a prior probability to pattern p from seach state s, we first use the pattern mover similarity network to calculate its similarity $sim(p, E)$ to all entities E in state s. then, we calculate its prior probability $p_\sigma(s, p)$ as follows:

$$p_\sigma(s, p) = \frac{sim(p, E)}{\sum_{p'} sim(p', E)} \quad (5)$$

In addition, to reduce the pattern selection complexity (the number of patterns can be tens of thousands), we firstly use the RlogF function to filter out some less related patterns and only select the top k patterns to calculate their selection probabilities. In our experiments, we set k as 200.

Reward Function in MCTS

The reward function is critical for efficiently estimating the real delayed feedback of each pattern. In this paper, we design the reward function as follows:

$$R = \frac{\sum_{e \in E'} sim(e, E)}{|E'|} sigmoid(\frac{|E'|}{a}) \quad (6)$$

where E is the set of known entities (seed entities plus extracted entities in previous iterations) in root state, E' is the set of all new extracted entities at the end of each simulation, $sim(e, E)$ is the similarity score of new extracted entity e to known entities, a is a constant. We exploit pattern mover similarity network to calculate the similairty score.

3 Pattern Mover Similarity Network

In this section, we describe the pattern mover similarity network (PMSN), which 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 Pattern Embeddings (APE) which can adaptively represent patterns, entities, and entity sets in a unified way; 2) the pattern mover similarity (PMS) measurement which calculates the similarity of two APEs.

The PMSN model is mainly used in three aspects: 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 new extracted entities inner MCTS simulation, whose evaluation scores will be used to calculate rewards. 3) the PMSN is also used as the entity scoring function at the Entity Scoring stage in the bootstrapping process as mentioned in 2.1.

3.1 Adaptive Pattern Embedding

In this subsection, we first describe how to embed patterns; then, we introduce how to obtain initial APE; finally, we introduce the probability adaption mechanism for better entity representation.

Pattern Embedding. As a basic step of our PMSN model, we use distributional representation techniques recently, e.g., word2vec (Mikolov et al., 2013), to embed context patterns. Specifically, we use the mean word embedding of a pattern surface text as the pattern embedding. The

word embeddings are from Glove (Pennington et al., 2014). And we filter out the pattern containing more than one OOV terms, whose embeddings are not contained in Glove.

Initial APE. In this paper, The APE embeds entities by a distribution on its context pattern embeddings, where the pattern embeddings can represent different semantics the entity may belong to or be related to; the distributional probabilities on those pattern embeddings refers to the different representativeness of difference patterns. Intuitively, if a context pattern of an entity have higher distributional probability, the pattern is more likely to represent the main semantic concept of this entity and the entities matched by this pattern will be more likely to belong to the same category with this entity.

Therefore, if we can find n most representative context patterns and calculate their distributional probabilities based on their representativeness, we can preciously represent the main semantics of entities. In this paper, we consider a pattern is representative to an entity if: the pattern co-occurs frequently with the entity; the pattern matched as few other entities as possible. Based on this intuition, we calculate the representative score for each context pattern p of an entity e as follows:

$$w(p,e) = \frac{N(e,p) \times \log N(e,p)}{C(p)} \tag{7}$$

where $C(p)$ is the number of different entities matched by p, $N(e,p)$ is the frequency of p_i co-occurring with entity e. Based on the calculated representative scores, the distributional probabilities of the APE can be calculated by normalizing all representative scores to sum 1.

In addition, above approach can be also used to calculate the APE for patterns and entity sets. 1) For patterns, we use the patterns themselves as their context patterns, and the corresponding distributional probability is 1.0. 2) For entity sets, e.g., a set of seed entities, we take the whole entity set as the input. The context patterns are those patterns which appears in the context of at least one entity in the entity set; and their distributional probabilities can be calculated as follows:

$$w(p,E) = \frac{N(E,p) \times \log N(E,p)}{C(p)} \tag{8}$$

where E is the entity set, $N(E,p)$ is the frequency of p co-occurring with entities in E.

Finally, we can denote the APE as $< X, w >$, where X is the context pattern embedding matrix, w is the vector of distribution probabilities. Due to the computing limitation, we only select top n representative patterns. Therefore, X is actually a $n \times d$ matrix, where d is the dimension of each pattern embedding, w is a n-dimensional vector.

Probability Adaption Using the MCTS Algorithm. Although the above approach can provide unified representations for both patterns and entities, it can still fail to represent the underground semantic concept of seed entities. For example, most context patterns of seed entity set {*London, Paris, Beijing*} are more related to the city concept, e.g., "* *is a big city*", but the above approach can still assign higher probabilities to those unrelated patterns matching many other entities.

To address the above problem, we design a probability adaption mechanism, which uses the MCTS algorithm to fine-tune the distributional probabilities. Specifically, we fine-tune the probabilities of patterns using returned action values after finishing MCTS simulations. As mentioned in Section 2.2, the action values of patterns are used as delayed feedback. And those patterns with higher delayed feedback are more likely to be representative ones, since they extract more positive entities and less negative entities. Therefore, we fine-tune the distributional probabilities as following:

$$w_t(p,e) \propto w_{t-1}(p,e) \cdot Q(s_0,p) \tag{9}$$

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

3.2 Pattern Mover Similarity

To compute the similarity of two APEs, we design a similarity measurement which considers both the similarity of pattern embeddings and the similarity of their probabilities.

Given two APEs, i.e., two pattern embedding matrices with distribution vectors, we calculate their similarity using the following formulas inspired by Kusner et al. (2015):

$$\max_{T \geq 0} \sum_{i,j=1}^{n} T_{ij} sim(i,j) \tag{10}$$

subject to:

$$\sum_{n}^{j=1} T_{ij} = w_i \quad \forall i \in 1,...,n \tag{11}$$

$$\sum_{n}^{i=1} T_{ij} = w_j \quad \forall j \in 1,...,n \tag{12}$$

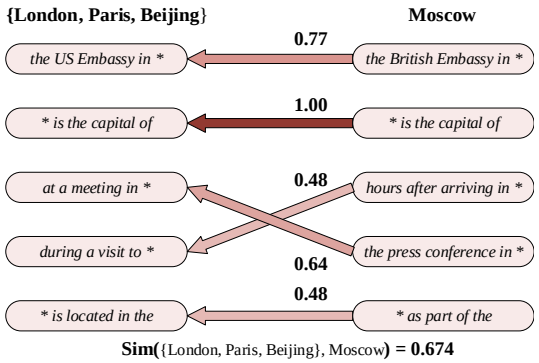


Figure 3: The components of the PMS metric between the APEs of the seed entity set {*London, Paris, Beijing*} and entity *Moscow* (with equal distributional probability for simplicity). The arrows represent flow between two patterns and their similarity.

where $sim(i, j)$ is the cosine similarity between the i^{th} pattern embedding of one entity and the i^{th} pattern embedding of the other entity. And we denote the above measurement as the Pattern Mover Similarity (PMS) measurement.

By using the above formula, PMS will assign high similarity scores to those APE pairs if: there are many common or similar patterns between two APE; they have similar high distributional probabilities on those representative patterns. Figure 3 shows an example about how PMS works.

Using the above measurement, our method can uniformly calculate the similarity of any two APEs of entities, patterns or entity sets.

4 Experiments

In this section, we demonstrate the experimental settings and the results of our bootstrapping method on entity set expansion.

4.1 Experimental Settings

Corpus. We conduct experiments on three public datasets: Google Web 1T corpus (Brants and Franz, 2006), APR and Wiki (Shen et al., 2017). 1) Google Web 1T corpus contains a large scale of n-grams compiled from a 1 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 Figure 1) list in Shi et al. (2014) and compare our method with traditional bootstrapping ESE methods 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 list in Shen et al. (2017) and compare the final entity scoring performance with it on both datasets.

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 in our experiments.

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

1) POS: bootstrapping method which only uses positive seeds and no no 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 positive and negative seeds both.
4) SetExpan(Shen et al., 2017): corpus-based entity set expansion method, which adaptively selects context features and unsupervisedly ensemble 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.

Metrics. To compare our methods with traditional bootstrapping methods on Google web 1T datatset, we use P@n (precision at top n), and the mean average precision (MAP) as the same as Shi et al. (2014). As for APR and Wiki datasets, we use MAP@n (n=10,20,50) to evaluate entity scoring performance of our method. In our experiments, we select seed entities manually; the correctness of all extracted entities is manually judged, with some external support resources, e.g., the entity list collected from Wikipedia.

4.2 Experimental Results

Comparison with three baseline methods on Google Web 1T corpus. Table 2 shows the performance of different bootstrapping methods on Google Web 1T dataset. We can see that our full model outperforms our three baseline methods: specifically, comparing with the baseline POS, our method can achieve 41% improvement on P@100, 35% improvement on P@200 and 45% improve-

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	0.9	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	0.6	0.56	0.48	0.42	0.61

Table 2: Overall results for entity set expansion on Google Web 1T dataset, 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 uses the human feedback for seed entity selection.

Method	APR		
	MAP@10	MAP@20	MAP@50
SetExpan	0.897	**0.862**	0.789
Ours	**0.913**	0.853	**0.797**

Method	Wiki		
	MAP@10	MAP@20	MAP@50
SetExpan	0.957	0.901	0.745
Ours	**0.963**	**0.925**	**0.786**

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

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 dataset.

ment on MAP; comparing with baseline MEB, our method can achieve 24% improvement on P@100 and 18% improvement on P@200; comparing to baseline COB, our method achieves 3% improvement on both P@100 and P@200, and 2% improvement on MAP. That means our method can extract more true entities and assign a higher ranking score to these entities.

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 novel entity set expansion method–SetExpan, which is a non-bootstrapping method(see Table 3). In Table 3, we can see that our method outperforms SetExpan on both APR and Wiki datasets: our method achieves 0.8% improvement on MAP@50 on APR dataset; and 4.1% improvement on MAP@50 on Wiki dataset. This further verifies that our method can improve the performance of bootstrapping for Entity Set Expansion.

4.3 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% on P@100 and 17% on P@200; if we replace the PMSN with word embedding, the performance decreases by 34% on P@100 and 27% on P@200. The results show that both the PMSN and the MCTS algorithm are critical for our model's performance. And they can be enhanced by each other: the PMSN can learn a better representation by combing with the MCTS algorithm; the MCTS can effectively estimate delayed feedback using the PMSN.

The performance of our full method on different categories on Google Web 1T dataset. Table 4 shows the performance of our method on different categories on Google Web 1T dataset. We can see that our method can achieve high performance in most categories except for ELE, TTL and FAC entities: the lower performance on ELE entities may mainly in order that total elements' number is less than 150, and most of them occur seldomly; the lower performance on TTL and FAC entities may mainly in order that context patterns of TTL entities are similar to context patterns of person names, and context patterns of FAC are similar to context pattern of location entities.

Top entities selected by different methods. To intuitively demonstrate the effectiveness of delayed feedback in our method, we illustrate the top

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.

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 also see that without the MCTS algorithm or PMSN , most top patterns are less related and easily semantic drift to other categories.

5 Related Work

Weakly supervised methods for information extraction (IE) are often provided scant supervision, such as knowledge base facts as distant supervision (Mintz et al., 2009; Hoffmann et al., 2011; Zeng et al., 2015), and light amount of supervision samples in bootstrapping(Riloff and Jones, 1999; Carlson et al., 2010; Gupta and Manning, 2014), or label propagation (Chen et al., 2006). As a classical technique, bootstrapping usually exploits pattern (Curran et al., 2007; Shi et al., 2014), document (Stevenson and Greenwood, 2005; Liao and Grishman, 2010) or syntactic and semantic contextual features (He and Grishman, 2015; Batista et al., 2015) to extract and classify new instances.

Limited to the sparse supervision, many previous bootstrapping methods often combine supervision signals with statistical features in previous iterations to evaluate and select patterns. Among them, the RlogF-like function (Agichtein and Gravano, 2000; Pantel and Pennacchiotti, 2006; Gupta and Manning, 2014) is the most popular method, but often suffers from semantic drift problem. Some other studies introduce extra constraints when selecting patterns, e.g., parallel mutual exclusion (Curran et al., 2007; McIntosh and Curran, 2008), coupling constraints (Carlson et al., 2010), and specific extraction pattern forms (Kozareva and Hovy, 2010). 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. However, all these methods exploit only instant feedback rather than delayed feedback.

To evaluate entity scores, a simple approach is directly labeling the extracted entities by selected patterns as positive ones (Curran et al., 2007; Carlson et al., 2010); a more effective approach is to leverage the pattern matching statistics (Riloff, 1996; Agichtein and Gravano, 2000; Thelen and Riloff, 2002; Pantel and Pennacchiotti, 2006; Kozareva and Hovy, 2010), or one-hot context pattern embeddings (Yangarber et al., 2000; Paşca, 2007; Pantel et al., 2009) inner the bootstrapping process. However, these methods depend on the iteration process and often fail for sparse matching patterns. Recently, Gupta and Manning (2015) use word embedding similarity of entities to help entity classification; Shen et al. (2017) use a non-bootstrapping method to adaptively select context features and ensemble them to evaluate entities; Berger et al. (2018) exploit human labels an external supervision to guide bootstrapping and learn a custom embedding to help human annotating.

6 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 show that our method can select better patterns based on efficiently estimated delayed feedback and learn a better entity scoring function using the PMSN combined with the MCTS algorithm. For future work, because bootstrapping is a common method for many information extraction tasks, we will design new models and apply them on related tasks, such as relation extraction, knowledge fusion, etc.

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.

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.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2006. Relation Extraction Using Label Propagation Based Semi-supervised Learning. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 129–136, Stroudsburg, PA, USA. Association for Computational Linguistics.

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.

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.

Zornitsa Kozareva and Eduard Hovy. 2010. Learning Arguments and Supertypes of Semantic Relations Using Recursive Patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1482–1491, 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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

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. 1996. An empirical study of automated dictionary construction for information extraction in three domains. volume 85, pages 101–134. Elsevier.

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.

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.

Roman Yangarber, Winston Lin, and Ralph Grishman. 2002. Unsupervised Learning of Generalized Names. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, pages 1–7, 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.