Extracting Knowledge from Web Text with Monte Carlo Tree Search

Guiliang Liu, Xu Li, Jiakang Wang, Mingming Sun, Ping Li
Cognitive Computing Lab
Baidu Research
No.10 Xibeiwang East Road, Beijing, China
10900 NE 8th St. Bellevue, WA 98004, USA
gla68@sfu.ca, {lixu13,v_wangjiakang,sunmingming01,liping11}@baidu.com

ABSTRACT

To extract knowledge from general web text, it requires to build a domain-independent extractor that scales to the entire web corpus. This task is known as Open Information Extraction (OIE). This paper proposes to apply Monte-Carlo Tree Search (MCTS) to accomplish OIE. To achieve this goal, we define a Markov Decision Process for OIE and build a simulator to learn the reward signals, which provides a complete reinforcement learning framework for MCTS. Using this framework, MCTS explores candidate words (and symbols) under the guidance of a pre-trained Sequence-to-Sequence (Seq2Seq) predictor and generates abundant exploration samples during *training*. We apply the exploration samples to update the reward simulator and the predictor, based on which we implement another MCTS to search the optimal predictions during *inference*. Empirical evaluation demonstrates that the MCTS inference substantially improves the accuracy of prediction (more than 10%) and achieves a leading performance over other state-of-the-art comparison models.

ACM Reference Format:
Guiliang Liu, Xu Li, Jiakang Wang, Mingming Sun, Ping Li. 2020. Extracting Knowledge from Web Text with Monte Carlo Tree Search. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3366423.3380010

1 INTRODUCTION

The rapidly-increasing Internet users generate a growing amount of text data every day. Those text data (e.g., news, tweets or blogs) contain abundant information and meaningful knowledge on various domains. To utilize the large amount of information from the web text, we would like to build an unlexicalized, domain-independent extractor that scales to the diversity and size of the Web corpus [2, 25, 26]. Among all the topics studied in recent years, Open Information Extraction (OIE) is essentially the most relevant task that satisfies the above specifications. OIE requires generating structured representations of information from unstructured knowledge implicit in natural language sentences. Figure 1 presents an example of OIE. An OIE extractor should directly distill natural language texts into facts (entity-relation tuples) without human involvement. The extracted facts can be applied as the source data of many data mining tasks and fuel countless downstream applications [11, 13, 16], including question & answering, text summarization, word analogy, etc.

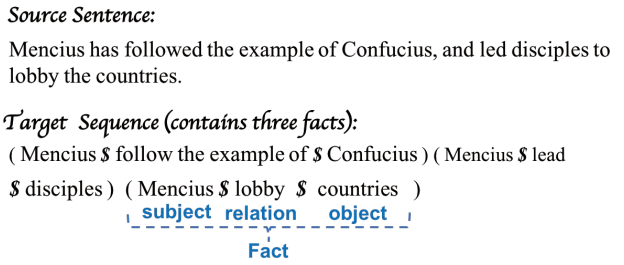

Figure 1: An example of OIE. Given a source sentence, we generate a sequence of facts (F_1, F_2, F_3). A fact is consisted of subject, relation and object. We use parentheses to separate facts and $ to denote different components of a fact. Both the symbols (e.g., $) and the words are required to be predicted.

Traditional OIE works commonly apply the pattern matching techniques [2, 7, 18] to locate the components of a fact, but pre-defined matching rules are difficult to generalize across different domains. To tackle this problem, a recent work [26] learned an end-to-end extractor which directly generates a sequence of facts from a source sentence. Based on the Seq2Seq model, their extractor applies the recurrent encoder-decoder model that iteratively predicts the words and the symbols (e.g., '$' and '(' in Figure 1) of prediction sequences. To enhance the performance, [25] deployed reinforcement learning (RL) which defines a reward function and updates the extractor with only sentence-reward pairs from the training dataset. The limitation of [25, 26] is twofold: 1) the lack of exploration constrains the model's knowledge about different combinations of candidate words; 2) at each generation step, the traditional Seq2Seq predictor applies either greedy or beam search to determine the predicted word, which is likely to trap the solution into a local optimum.

This paper proposes to use Monte Carlo Tree Search (MCTS) [5] to overcome the aforementioned limitation. By applying the rewards and the prior probabilities from a sequence-to-sequence (Seq2Seq) predictor, MCTS builds an action-state game tree, explores different candidate words (and symbols) at every tree node, and guides the tree search with a Predictor Upper Confidence Bound (PUCB). As an effective heuristic, PUCB not only evaluates the currently predicted words but also looks ahead to the end of prediction, which effectively prevents local optimum and substantially boosts searching quality.

However, to extend the MCTS solution to OIE tasks, we would have to deal with multiple challenges: 1) MCTS solutions are built on the Markov Decision Process (MDP) whereas none of the previous

WWW '20, April 20–24, 2020, Taipei

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380010

works have defined an MDP for knowledge extraction problems. 2) MCTS uses rewards to guide the tree search, but the process of sequence prediction does not explicitly return such a reward signal. 3) An OIE extractor should efficiently handle a large amount of web text, but it is difficult for traditional parallel MCTS to finish actions exploration and predictions generation within a reasonable time.

In this work, to resolve these challenges, we develop a novel RL framework that incorporates the MCTS solution into the task of OIE. The RL framework defines a similarity reward that encourages the model to generate the facts resembling the ground-truth ones. We also propose a novel reward simulator that dynamically evaluates the predicted facts and continuously provides reward signals to MCTS when the ground-truth labels are not available. During training, MCTS explores the candidate words and symbols under the guidance of a pre-trained Seq2Seq predictor and generates numerous exploration samples, with which we update both the predictor and the simulator. Based on the updated predictor and simulator, we implement another MCTS during inference. To reduce the running time, we propose a batched parallel MCTS, which makes our model run nearly 20 times faster than traditional parallel MCTS. Empirical evaluation shows the MCTS inference achieves a significantly higher accuracy (more than 10%) than state-of-the-art models.

The **main contributions** of this paper are summarized as follows:

- To our best knowledge, this is the first work that applies MCTS to solving the knowledge extraction problem;
- We propose a reward simulator to generate rewards for MCTS;
- We develop an RL framework for the task of OIE which enables MCTS to search for the optimal prediction sequence during inference and largely improves the prediction accuracy.

2 RELATED WORKS

Open Information Extraction (OIE) [2] defines a domain-independent task of extracting facts (entity and relation tuples) from the natural language sentences (e.g., news or blogs), in contrast with the Close Information Extraction (CIE) which identifies instances from a fixed and finite set of corpus [17, 32]. Traditional OIE solutions often construct their handcrafted heuristics to locate the facts (e.g., OpenIE-v4 system defined two independent extractors: 1) SRLIE to find verb-based relations [4] and 2) ReNoun to extract nominal attributes [31]). The existing extraction models mainly apply pattern matching techniques [2, 4, 31]. The handcrafted heuristics and the pattern matching methods, however, have limited generalization ability across different domains.

To tackle the problem, some recent OIE works have built deep recurrent models to capture features from source sentences and iteratively generate a fact (or its labels) word-by-word. For instance, Sun et al. [26] built an end-to-end OIE extractor based on the Seq2Seq predictor [27], by directly transferring the input source sentence into a prediction sequence containing a list of facts. Sun et al. [25] further improved the performance of Seq2Seq predictors with reinforce algorithm [19], which computes a reward signal and updates the model with $\langle sequence, rewards \rangle$ pairs. However, their training data are limited to the existing samples in the dataset. Without exploring other possible predictions, the reinforce method achieves only limited improvement over the Seq2Seq extractor. More recently, [15] proposed a Confidence Exploration for OIE without learning a reward function, ie., their method explores only during training.

Monte-Carlo Tree Search (MCTS) [5] is an iterative heuristic search algorithm that explores different actions and searches for the optimal sequence of decisions. The exploring and searching ability has benefited many tasks under game environment [22, 23] (e.g., Go), which requires a massive number of simulations to determine the optimal move. To improve the searching efficiency and efficacy, [21] proposed the Predictor Upper Confidence Bound (PUCB), which guides MCTS with the prior probabilities from a pre-trained predictor. Another technique of reducing the running time is parallelization: Chaslot et al. [3] introduced a parallel MCTS implemented with a multi-thread program, but this parallel framework is only applicable to the traditional MCTS without interacting with a predictor.

3 TASK FORMULATION AND APPROACH

This section introduces our approach to formulating OIE as a sequence prediction task and developing a reinforcement learning (RL) framework for MCTS.

3.1 OIE as a Sequence Prediction Task

We accomplish the OIE task in the manner of sequence prediction [1, 14, 26]. The first step is dividing the natural language text into source sentences. Given a source sentence X, our model iteratively generates a prediction sequence $\hat{Y}$ of T items: $\hat{Y}_{1...T} = \{\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_T\}$. Each item $\hat{y}_t$ can be a word or a symbol (e.g., '$' and '(' in Figure 1) that marks the predicted facts. A predicted fact $\hat{F}$ is a sub-sequence of prediction $\hat{Y}$: $F = \{\hat{y}_t, \cdots, \hat{y}_{t+\gamma}\}$ and $t + \gamma < T$. F contains both the words recording entity-relations and the symbols that mark the structure of this fact. The order of facts within a prediction sequence has not been considered during evaluation.

3.2 Construct Markov Model for OIE

We define a Markov Decision Process (MDP) for the task of sequence prediction. At inference step t, the **state $\mathbf{s_t}$** of MDP contains both the source sentence X and the up-to-now predictions $\hat{Y}_{1..t-1}$. The **action $\mathbf{a_t}$** is the next item (word or symbol) to be predicted ($\hat{y}_t$), which is selected according to the estimated probability distribution $\Pr(\hat{y}_t|X, \hat{Y}_{1..t-1})$ for all candidate words and symbols. After determining a_t, our agent reaches the **next states $\mathbf{s_{t+1}}$**= $\langle X, \hat{Y}_{1..t} \rangle$. During this transition, the Markov property is strictly preserved as $\langle X, \hat{Y}_{1..t} \rangle$ records all previous knowledge (often with a recurrent network). However, one difficulty of building a complete MDP is that *the predictions do not explicitly provide a reward signal*. We discuss this phenomenon and provide our solution in the following.

3.3 Complete the Environment with Rewards

Unlike game environments that directly return reward signals, our OIE task does not explicitly provide such a signal, so we define an OIE reward and introduce our approach to computing it.

Our OIE reward r_t is the similarity score Sim between predicted facts and ground-truth facts. At inference step t, r_t is compute by: 1) given a prediction sequence $\hat{Y}_{1..t}$ (contains N_P predicted facts $\{\hat{F}_1, ..., \hat{F}_{N_P}\}$) and a target sequence Y^* (contains N_G ground-truth facts $\{F^*_1, ..., F^*_{N_G}\}$), we match each predicted fact with a ground-truth fact by finding an optimal assignment that maximizes their matching similarity. The matched fact pairs are

$\{\langle \hat{F}_i, F^*_j \rangle_l\}_{l=1}^{\min(N_p, N_G)}$. 2) The similarity score between the prediction sequence and the target sequence is defined as the similarity of matched facts: $Sim(\hat{Y}_{1..t}, Y^*) = \sum_{l=1}^{\min(N_p, N_G)} \eth(\langle \hat{F}_i, F^*_j \rangle_l)$, where $\eth$ denotes the Gestalt Pattern Matching [20]. It is a string-matching algorithms for determining the similarity of two strings. *The essence of similarity score agrees with the goal of OIE, which is extracting the most complete and accurate facts that resemble the ground-truth ones.* To maximize $Sim(\hat{Y}_{1..t}, Y^*)$, the agent adjusts its policy and learns how to generate the prediction sequences containing the ground-truth facts. However, a limitation of our similarity function is its requirement of target sequences *(available only during training)*. During inference, the target sequences (Y^*) are unknown, which hinders the reward computation. To resolve this problem, we introduce a novel reward simulator $\mathbb{S}$ (Section 4.1) that learns with the similarity function and generates rewards with only $\hat{Y}_{1..t}$ and $X_{1...M}$.

3.4 Our Approach: An RL Framework for MCTS

We introduce our RL framework that enables MCTS to find the optimal prediction sequence for the task of OIE. Figure 2 illustrates our RL framework where we incorporate MCTS into both the training and the inference procedures to generate the prediction sequence $\hat{Y}$.

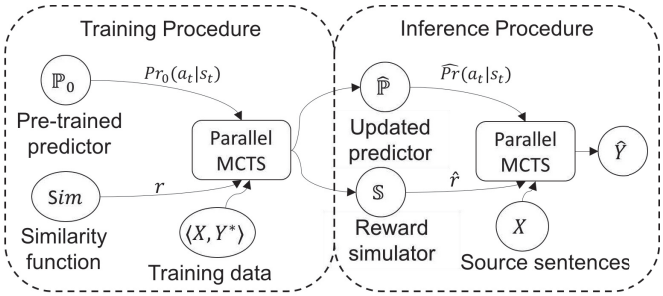

Figure 2: The RL framework for the task of OIE.

During *training*, we input the training data $\langle X, Y^* \rangle$ and a pre-trained Seq2Seq predictor $\mathbb{P}_0$ into the parallel MCTS. MCTS generates numerous exploration sequences and their rewards, with which we update the predictor and learn a reward simulator $\mathbb{S}$. During *inference*, given only the source sentences (without target sequences), MCTS uses the rewards from the updated simulator and probabilities from the updated predictor to guide the tree search and generate the prediction sequences $\hat{Y}$. This framework enables our model to take advantage of the exploration ability of MCTS during training and efficiently search for the optimal predictions during inference. The learning details of both procedures are introduced in Section 5.

4 MODEL

This section introduces three main elements of the RL framework, including a reward simulator, a Seq2Seq predictor, and MCTS.

4.1 Reward Simulator

We introduce a novel reward simulator that evaluates current predictions and generates rewards when target sequences are not available. The reward simulator is a key component of a complete RL environment for OIE, especially under the industrial circumstance where our model is required to generate predictions without knowing the target sequences Y^*. To achieve this goal, given a source sentence X and a prediction sequence $\hat{Y}_{1...t}$ containing N_p facts $\{\hat{F}_i\}_{i=1}^{N_p}$, the reward simulator $\mathbb{S}$ should dynamically determine whether the extracted facts record the complete and correct information in X. To implement such a simulator, we design a two-tower neural structure which separately fits a fact and a source sentence (Figure 3). Both towers apply bi-directional LSTM to extract sequential features. The features are merged to the output layer which generates a score $\mathbb{S}(X, \hat{F}_i)$ for the fact. The summation of the scores for every facts is the simulated reward of a prediction sequence: $r_t = \sum_{i=1}^{N_p} \mathbb{S}(X, \hat{F}_i)$.

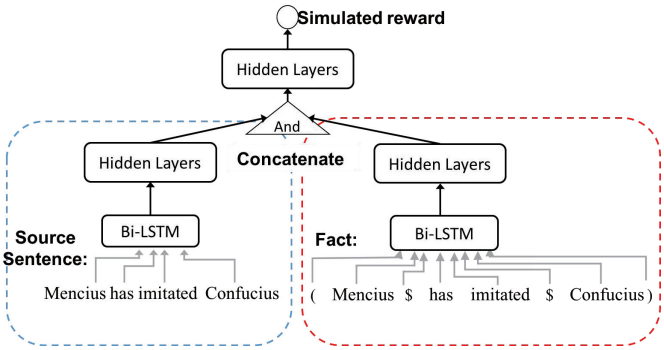

Figure 3: Model structure of our reward simulator.

To learn an accurate simulation reward, we utilize the training data (ground-truth facts are available) and apply the real similarity score $\eth(\hat{F}_i, F^*_j)$ to supervise the training of simulator by the L2-norm loss function (Formula (3)). This reward simulator offers an alternative approach to learning a reward signal when it is hard to explicitly compute a reward for the sequence prediction problem.

4.2 Seq2Seq Predictor

The Seq2Seq model [26, 27] uses an encoder to fit the input sentences and a decoder to generate the prediction sequences. Both the encoder and the decoder apply a bidirectional Gated Recurrent Units (GRU) layer to capture sequential relations between words. Following [26], we build an end-to-end Seq2Seq predictor which directly generates a prediction sequence $\hat{Y}$ (which contains extracted facts $\{\hat{F}_1, \hat{F}_2, \cdots, \hat{F}_{N_P}\}$) when given a source sentence X. At inference step t, the decoder produces a probability distribution of the candidate words and symbols: $\Pr(y_t|y_1, \cdots, y_{t-1}; c_t) = g(h_{t-1}, \phi_t, c_t)$, where h_{t-1} and ϕ_t are the hidden states from the GRU encoder and the GRU decoder respectively, g is the word generation model, and c_t is the dynamic context vector. The decoder implements the generation model g with the *copy mechanism* [10] which generates a prediction $\hat{y}_t$ by either copying words from the input sentence X or selecting symbols from a set of pre-defined markers (e.g., '$'). This design satisfies the specification of OIE, which requires: 1) extracting meaningful information from source sentences (by copying words); and 2) reconstituting the words with a pre-defined format (defined with the symbols). The dynamic context vector c_t implements the *coverage mechanism* [30]. It applies an additional coverage vector for every word in source sentences to remember their individual attention history, which prevents information loss or redundancy in prediction sequences.

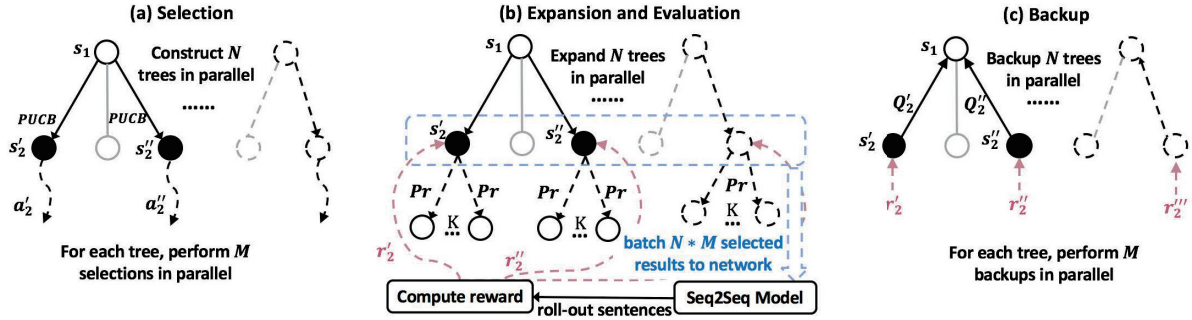

Figure 4: The parallel implementation of MCTS for OIE. After selection, we batch N(batch size)$* M$ (play number) states from the leaf nodes before inputting them to the predictor. The predictor generates the exploration sequences with policy roll-out. We evaluate the sentences with the similarity function ($Sim(\hat{Y}_{1..t}, Y^*)$) and the reward simulator ($\mathbb{S}$) during training and inference respectively. To complete $N * M$ plays, our method requires only one interaction with the neural model, and thus effectively accelerates MCTS.

4.3 MCTS

In this section, we introduce MCTS based on the Seq2Seq predictor [23] and our approach to accelerating MCTS with parallelization.

4.3.1 Predictor MCTS. MCTS is mainly about building and updating a game tree where nodes denote the states s_t=$\langle X, \tilde{Y}_{1..t-1}\rangle$ and edges represent the actions to be explored a_t=$\langle\tilde{y}\rangle$. Each edge also records a visit count $N(s_t, a_t)$, an action-value $Q(s_t, a_t)$, and the prior probability $\Pr_0(a_t|s_t)$ from a pre-trained Seq2Seq predictor $\mathbb{P}$. Based on the game tree, we run the tree search by iteratively implementing a *play* including four phases: 1) *Selection*: select actions a_t according to a Predictor Upper Confidence Bound (PUCB) and traverse the tree from root to a leaf node. PUCB is defined as:

$$a_t = \arg\max_a \left[Q(s_t, a) + c_{puct} Pr_0(s_t, a) \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s_t, a)} \right], \quad (1)$$

where $\Pr_0(s_t, a)$ holds the prior probabilities from a pre-trained predictor $\mathbb{P}_0$ and c_{puct} controls the scale of exploration. This search control strategy initially prefers actions with high prior probability and low visit count, but asymptotically prefers actions with high action value [23]. 2) *Evaluation*: evaluate the sequences of actions selected along the tree traversal and compute the reward r_t. 3) *Expansion*: expand the leave node with K child nodes that contain the actions with top-k prior probabilities. 4) *backup*: increment the visit count $N(s_t, a_t)$ and update the action-values $Q(s_t, a_t)$ (the expected cumulative future rewards [28] that look ahead to the end of predictions) on all the traversed edges.

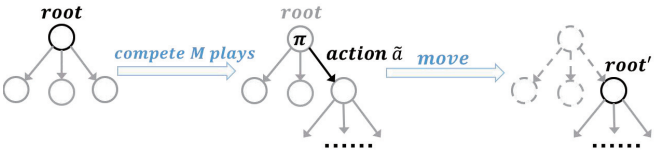

Figure 5: An example of *move* from the root node.

After implementing M (play number) *play*s, we make a *move* from the root node by sampling an action according to the distribution of visit number: $\hat{a} \backsim \pi(a|s_{root})$, because visit numbers are proportional to the action optimality (computed by PUCB during tree search). After determining the predicted action $\hat{a}$, we move the root to the child node through the edge recording the action (see Figure 5). After moving, the next play will start from the new root. We iteratively implement the move T times until we generate a final prediction sequence $\hat{Y}_{1...T} = \{\hat{a}_1, \hat{a}_2, \cdots, \hat{a}_T\}$.

4.3.2 Parallelize MCTS. To improve the running efficiency, we propose a batching update to the parallel implementation of MCTS . Previous works on parallel MCTS algorithms [3] perform the parallel tree searches in separate threads. During running, each thread interacts with the Seq2Seq predictor and the reward simulator independently, which undermines the advantage of highly parallel matrix computation implemented in GPU [9]. As evidence, our experiment (Table 3) shows the running time of the evaluation phase (where MCTS interacts with the neural models) dominates others for the traditional parallel MCTS. To overcome this limitation, we merge the selected states from individual threads and batch the inputs before computing prior probabilities and rewards. After evaluation, those values are distributed to the parallel threads. This updating significantly reduces the number of interactions, and thus improves the algorithm efficiency. We show an example in Figure 4.

5 LEARNING

This section introduces our learning method that incorporates MCTS into both the training and the inference procedures.

5.1 Training Procedure

The goal of training is 1) updating the pre-trained Seq2Seq predictor $\mathbb{P}^0$ and 2) learning a reward simulator $\hat{\mathbb{S}}$. We apply the Reinforce algorithm [19] to update the Seq2Seq predictor, but unlike previous works [26] applying only sentences and rewards pairs from training dataset, our MCTS generates and evaluates numerous exploration sequences $\tilde{Y}^e$ during the tree search. The exploration sequences $\langle X, \tilde{Y}^e\rangle$ along with the reward signals $r^e = Sim(\tilde{Y}^e, Y^*)$ record the searching information of MCTS and significantly expand the training samples. The parameters of the Seq2Seq predictor $\mathbb{P}$ are updated by:

$$\theta^{\mathbb{P}} \leftarrow \theta^{\mathbb{P}} + \frac{d}{d\theta^{\mathbb{P}}} \left[\sum_{n=1}^{N} r_n^e \log \Pr(\tilde{Y_n}^e | X_n) \right] \quad (2)$$

For the reward simulator $\mathbb{S}$, it is trained to measure if the exploration sequences $\tilde{Y}^e$=$\{\tilde{F}_1^e, ..., \tilde{F}_{N_p}^e\}$ manage to capture the facts hidden in source sentence X, so we update the simulator parameters with the

same exploration samples by:

$$\theta^{\mathbb{S}} \leftarrow \theta^{\mathbb{S}} - \frac{d}{d\theta^{\mathbb{S}}} \sum_{n=1}^{N} \sum_{i=1}^{N_p} \left[\eth(F^*_{i,n}, \tilde{F}^e_{i,n}) - \mathbb{S}(X_n, \tilde{F}^e_{i,n}) \right]^2 \quad (3)$$

where $F^*_{i,n}$ and $\tilde{F}^e_{i,n}$ are facts in target sequences and exploration sequences, N is the batch size, N_p is the number of predicted fact and Gestalt Pattern Matching $\eth$ measures string similarity.

5.2 Inference Procedure

During the inference procedure, given an input sentence, we implement another to search the prediction sequences, under the guidance of the updated predictor $\hat{\mathbb{P}}$ and reward simulator $\hat{\mathbb{S}}$ from training. Unlike previous learning-based sequence prediction models [19, 26] which determine predictions by directly applying a simple heuristic search algorithm (e.g., beam search) on the output probabilities, our algorithm utilizes reward signals, and the reinforcement learning method in MCTS guarantees that each predicted item (word or symbol) will look ahead to the end of prediction and has the largest expected cumulative rewards [28]. Our experiment results (table 1) also indicate that the beam search cannot achieve comparable performance with MCTS even if we use a large beam size.

6 EMPIRICAL EVALUATION

This section studies the performance of MCTS by evaluating the predicted facts and measuring the running efficiency.

6.1 Experiment Setting

6.1.1 Dataset: A recent survey for OIE [18] provided a detailed investigation over the available datasets. We find the most commonly applied datasets are WEB, WIKI, NYT, and PENN, which record sentences from web text, Wikipedia, New York Times Corpus, and Penn Treebank respectively. The datasets, however, record only less than 500 source sentences [6]. The lack of training data makes our models overfit the sentences, as our preliminary experiments show.

In this paper, we apply a recently proposed SAOKE dataset [1] which contains over 47,000 source-target sequences $\langle X, Y \rangle$ pair from Baike, one of the largest web-based encyclopedias. Unlike many system-labeling datasets (e.g., OIE 2016 [24]) that utilize outputs from pattern-matching systems to supervise training, SAOKE contains manually labeled ground-truth facts from crowd-sourcing. The facts are extractions from human professionals, and thus more accurately present the information of source sentences.

6.1.2 Running Settings: We split the SAOKE dataset, containing 47,000 source sentences and target sequences, into training (80%), validation(10%) and testing(10%) set. To simulate the practical OIE environment, we hide the target sequences from models during inference. The expansion number K is set to 3, with which our preliminary experiment shows a satisfactory performance. Our method is implemented with the PaddlePaddle[2] deep learning platform.

6.1.3 Comparison Methods: We compare two traditional OIE models based on pattern matching techniques. The first model CORE [29] is a system that selects entity-relation triples by matching a series of intermediate NLP components, including word segmentation, syntactic parsing, and rules extraction. The second model [12] builds an unsupervised OIE extractor based on Dependency Semantic Normal Forms (DSNF). Compared to CORE, this model imposes no restrictions on the relative positions among entities and relationships.

We also compare several more advanced end-to-end extractors: Logician [26] is a Seq2Seq predictor designed for the OIE task. It applies target sequences to supervise the model training with the teacher-forcing technique. To improve the model performance, Ranzato et al. [19] proposed a reinforce algorithm for the sequence prediction tasks. They refined the model with only samples and their expected rewards from training data. A continuing work [25] defines an Open-Domain Information Narration (OIN) task which transfers the extracted facts back to a source sentence. Their model assembles both the OIE agent and the OIN agent into a "dual" system, and utilizes the dual structure as a reinforcement learning paradigm.

Besides the existing baseline methods, we also investigate the impact of inference strategies. We perform a beam search on the updated Seq2Seq predictor $\hat{\mathbb{P}}$ from training procedure (denoted by MCTS@Train) and infer the final predictions (denoted by Beam@Infer). This comparison method (MCTS@Train + Beam@Infer) enables us to study the effect of replacing MCTS inference with a beam search. We compare it with *our approach*: MCTS@Train + MCTS@Infer.

6.2 Performance Evaluation

This experiment studies the accuracy of extracted facts. We evaluate the facts by their similarity with the ground-truth facts. The extracted facts are labeled as "true" if their relations, subjects, and objects are the same as that from the ground-truth facts. Given these labels, we compute the quantification metrics including precision (P), recall (R) and F1-score. The performances of our approach and comparison methods are reported in Table 1. For the models applying beam search, we experiment with both a common beam size ($B = 3$) and a large beam size ($B = 50$) to study the impact of search scale.

Table 1: Evaluation results for the predicted facts.

	Training Data			Testing Data		
	P	R	F1	P	R	F1
DSNF	0.126	0.100	0.170	0.220	0.112	0.148
CORE	0.348	0.183	0.240	0.400	0.1760	0.232
Logician($B = 3$)	0.560	0.478	0.515	0.469	0.400	0.432
Reinforce($B = 3$)	0.580	0.460	0.513	0.487	0.410	0.445
Dual(B=3)	0.594	0.499	0.543	0.494	0.426	0.457
Logician($B = 50$)	0.555	0.491	0.521	0.466	0.407	0.435
Reinforce($B = 50$)	0.583	0.460	0.514	0.485	0.416	0.448
Dual($B = 50$)	0.594	0.501	0.544	0.498	0.422	0.457
MCTS@Train + Beam@Infer($B = 3$)	0.573	0.475	0.519	0.518	0.425	0.467
MCTS@Train + Beam@Infer($B = 50$)	0.586	0.473	0.523	0.519	0.422	0.465
MCTS@Train + MCTS@Infer (**Ours**)	**0.690**	**0.582**	**0.632**	**0.611**	**0.506**	**0.554**

From the results, we can see that the pattern matching methods (DSNF and CORE) achieve only limited performance, especially for the recall metric. It is caused by a large number of unmatched facts during extraction, which proves that the pre-defined rules or schemas fail to generalize to all the natural language corpus. Compared to them, the end-to-end model Logician manages to predict the facts

[1] http://ai.baidu.com/broad/subordinate?dataset=saoke
[2] https://www.paddlepaddle.org.cn

with higher precision and recall. We also find that despite both Reinforce and Dual can improve the performance of our Seq2Seq model, but insufficient exploration inhibits the growth of accuracy.

Another crucial obseravation is that the updated predictor manages to further improve the model performance for testing data (see MCTS@Train+Beam@Infer) after training with the exploration samples generated by MCTS. It's because the MCTS effectively explores the candidate words and symbols that are likely to appear under different situations, and thus generates the exploration sentences of high quality. As evidence, Figure 6 shows the rewards of facts generated by MCTS are much larger than that from the Seq2Seq predictor (logician). By combining the exploration samples with the large scale tree search, our approach (MCTS@Train+MCTS@Infer) achieves the best performance among all comparison methods.

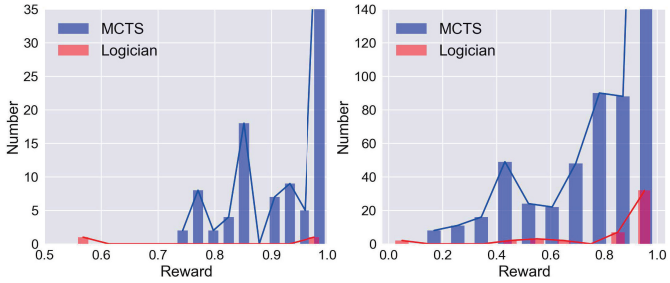

Figure 6: Distribution of fact rewards from prediction sentences. The numbers of source sentences are 1 (left) and 20 (right). Compared to Logician, MCTS explores more facts having rewards that approximate 1 (the reward for ground-truth).

However, *the success of MCTS does not simply imply a larger scale of search will always help*. For example, our results show that the performance of the beam search cannot match that of MCTS during inference even if we set a very large beam size ($B = 50$). Unlike other search algorithms, MCTS benefits a lot from its underlying heuristic (PUCB) which not only balances the exploration and exploitation (see Eq. (1)) but also enables our model to look ahead to the end of predictions (with action-value Q).

6.3 Error Analysis

We summarize the prediction errors of our approach and analyze the model limitation, which pinpoints the direction of improvement for future work. Following [8], we randomly select 100 prediction sequences generated by MCTS during inference and summarize the error of predicted facts in Table 2.

Table 2: Prediction error by percentage.

Error Type	Training	Testing
Correct relation,incorrect subjects	28.1%	13.5%
Correct relation,incorrect objects	15.6%	24.3%
Incomplete/over-extracted relation	34.4%	32.4%
Incorrect relation category	12.5%	8.1%
Wrong relation	9.4%	16.2%
Other, include incorrect format	0%	5.4%

We find a large section of incorrect facts (over 40% for the training set and 30% for the testing set) appears when our model manages to extract the correct relations but predicts the wrong subjects or objects. A typical example is the misuse of placeholder "_". When the information in a source sentence is incomplete, the target fact will use a "_" to mark the missing subject or object. It is hard for our model to realize the missing information and generate a "_" in the correct position. Another major type of error (over 30% for both training and testing set) is incomplete or over-extracted relations. It is difficult for our model to match the complete human knowledge.

6.4 Efficiency Evaluation

The running time is an important criterion for MCTS, as it consumes significantly larger computing sources than the traditional pattern match methods. In this experiment, we compare the running time of our batched parallel MCTS against another two common implementations of MCTS, including a single thread MCTS and a traditional parallel MCTS [3]. To provide a detailed evaluation of efficiency, we divide an entire play of MCTS into four main phases including Selection, Expansion, Evaluation, and Backup (introduced in Section 4.2) and study their individual running time.

Table 3: Total running time (by minutes). Num indicates the number of source sentences and $\mp$ denotes the estimated time.

Model	Num	Selection	Evaluation	Expansion	Backup
Single MCTS	30	2.22	3020.74	0.07	1.60
	60	4.38	$5915.83^{\mp}$	0.11	3.12
	120	8.98	$12244.14^{\mp}$	0.22	7.69
Parallel MCTS	30	0.03	440.31	0.00	0.04
	60	0.06	$780.82^{\mp}$	0.01	0.07
	120	0.12	$1590.77^{\mp}$	0.01	0.14
Batched Parallel MCTS	30	0.06	22.61	0.00	0.04
	60	0.13	37.00	0.01	0.11
	120	0.28	78.98	0.02	0.37

Table 3 records the running time of extracting facts from different numbers of source sentences (30, 60 and 120). An apparent observation is the parallel MCTS (both traditional and ours) processes much faster than single thread MCTS. It is consistent with the results in [3]. We also find the running time of the evaluation dominates the other three phases. It is because interacting with deep model s (the Seq2Seq predictor and the reward simulator) requires complex GPU matrix computation and becomes rather time-consuming. This phenomenon becomes more serious as we are handling more source sentences together. To alleviates the problem, our approach merges the individual threads and batches the select samples before evaluations, which significantly reduces the increase of running time as the model is required to process a larger number of source sentences.

7 CONCLUSION

This paper introduces an RL framework that enables MCTS to search the optimal prediction for the task of OIE. Under this framework, MCTS efficaciously explores the action spaces with the guidance of a pre-trained Seq2Seq model during training. We apply the explorations samples to update the predictor and the simulator. During inference, we launch another MCTS to search for the optimal predictions with the updated predictor and simulator. Empirical evaluation demonstrates that our approach achieves a significant higher prediction accuracy and running efficiency over other comparison methods.

REFERENCES

[1] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An Actor-Critic Algorithm for Sequence Prediction. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Toulon, France.

[2] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*. Hyderabad, India, 2670–2676.

[3] Guillaume Chaslot, Mark H. M. Winands, and H. Jaap van den Herik. [n.d.]. Parallel Monte-Carlo Tree Search. In *Proceedings of 6th International Conference on Computers and Games (CG)*.

[4] Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An Analysis of Open Information Extraction Based on Semantic Role Labeling. In *Proceedings of the 6th International Conference on Knowledge Capture (K-CAP)*. Banff, Alberta, Canada, 113–120.

[5] Rémi Coulom. 2006. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Proceedings of the 5th International Conference on Computers and Games (CG)*. Turin, Italy, 72–83.

[6] Filipe de Sá Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and Efficiency of Open Relation Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Seattle, WA, 447–457.

[7] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open Information Extraction: The Second Generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*. Barcelona, Spain, 3–10.

[8] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open Question Answering over Curated and Extracted Knowledge Bases. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. New York, NY, 1156–1165.

[9] Kayvon Fatahalian, Jeremy Sugerman, and Pat Hanrahan. 2004. Understanding the efficiency of GPU algorithms for matrix-matrix multiplication. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware 2004*. Grenoble, France, 133–137.

[10] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany.

[11] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge Graph Embedding Based Question Answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM)*. Melbourne, Australia, 105–113.

[12] Shengbin Jia, Shijia E, Maozhen Li, and Yang Xiang. 2018. Chinese Open Relation Extraction and Knowledge Base Establishment. *ACM Trans. Asian & Low-Resource Lang. Inf. Process.* 17, 3 (2018), 15:1–15:22.

[13] Dingcheng Li, Siamak Zamani, Jingyuan Zhang, and Ping Li. 2019. Integration of Knowledge Graph Embedding Into Topic Modeling with Hierarchical Dirichlet Process. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NNAACL-HLT)*. Minneapolis, MN, 940–950.

[14] Xu Li, Mingming Sun, and Ping Li. 2019. Multi-Agent Discussion Mechanism for Natural Language Generation. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*. Honolulu, Hawaii, 6096–6103.

[15] Guiliang Liu, Xu Li, Miningming Sun, and Ping Li. 2020. An Advantage Actor-Critic Algorithm with Confidence Exploration for Open Information Extraction. In *Proceedings of the 2020 SIAM International Conference on Data Mining (SDM)*. Cincinnati, Ohio.

[16] Mausam. 2016. Open Information Extraction Systems and Downstream Applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*. New York, NY, 4074–4077.

[17] Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany.

[18] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A Survey on Open Information Extraction. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*. Santa Fe, New Mexico, 3866–3878.

[19] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico.

[20] John W Ratcliff and David E Metzener. 1988. Pattern Matching the Gestalt Approach. *Dr Dobbs Journal* 13, 7 (1988), 46.

[21] Christopher D. Rosin. 2010. Multi-armed bandits with episode context. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*. Fort Lauderdale, FL.

[22] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[23] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (2017), 354.

[24] Gabriel Stanovsky and Ido Dagan. 2016. Creating a Large Benchmark for Open Information Extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, TX, 2300–2305.

[25] Mingming Sun, Xu Li, and Ping Li. 2018. Logician and Orator: Learning from the Duality between Language and Knowledge in Open Domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Brussels, Belgium, 2119–2130.

[26] Mingming Sun, Xu Li, Xin Wang, Miao Fan, Yue Feng, and Ping Li. 2018. Logician: A Unified End-to-End Neural Approach for Open-Domain Information Extraction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*. Marina Del Rey, CA, 556–564.

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Canada, 3104–3112.

[28] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[29] Yuen-Hsien Tseng, Lung-Hao Lee, Shu-Yen Lin, Bo-Shun Liao, Mei-Jun Liu, Hsin-Hsi Chen, Oren Etzioni, and Anthony Fader. 2014. Chinese Open Relation Extraction for Knowledge Acquisition. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Gothenburg, Sweden, 12–16.

[30] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany.

[31] Mohamed Yahya, Steven Whang, Rahul Gupta, and Alon Y. Halevy. 2014. ReNoun: Fact Extraction for Nominal Attributes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, 325–335.

[32] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel Methods for Relation Extraction. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, (EMNLP)*. Philadelphia, Pennsylvania.