Name Entity Recognition with Policy-Value Networks

Yadi Lao[†] Jun Xu[‡*] Sheng Gao[†] Jun Guo[†] Ji-Rong Wen[‡]

[†] Beijing University of Posts and Telecommunications
[‡] School of Information, Beijing Key Laboratory of Big Data Management and Analysis Methods, Renmin University of China
{laoyadi,gaosheng,junguo}@bupt.edu.cn,{junxu,jrwen}@ruc.edu.cn

ABSTRACT

In this paper we propose a novel reinforcement learning based model for named entity recognition (NER), referred to as MM-NER. Inspired by the methodology of the AlphaGo Zero, MM-NER formalizes the problem of named entity recognition with a Monte-Carlo tree search (MCTS) enhanced Markov decision process (MDP) model, in which the time steps correspond to the positions of words in a sentence from left to right, and each action corresponds to assign an NER tag to a word. Two Gated Recurrent Units (GRU) are used to summarize the past tag assignments and words in the sentence. Based on the outputs of GRUs, the policy for guiding the tag assignment and the value for predicting the whole tagging accuracy of the whole sentence are produced. The policy and value are then strengthened with MCTS, which takes the produced raw policy and value as inputs, simulates and evaluates the possible tag assignments at the subsequent positions, and outputs a better search policy for assigning tags. A reinforcement learning algorithm is proposed to train the model parameters. Empirically, we show that MM-NER can accurately predict the tags thanks to the exploratory decision making mechanism introduced by MCTS. It outperformed the conventional sequence tagging baselines and performed equally well with the state-of-the-art baseline BLSTM-CRF.

ACM Reference Format:
Yadi Lao, Jun Xu, Sheng Gao, Jun Guo, Ji-Rong Wen. 2019. Name Entity Recognition with Policy-Value Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331349

1 INTRODUCTION

Named entity recognition (NER) has gained considerable research attention for a few decades. The main goal is to extract entities such as PERSON, LOCATION etc. in a given sentence. Existing models can be categorized into the statistical models and the deep neural networks based models. Traditional research focuses on the linear statistical models, including the maximum entropy (ME) classifier [10] and maximum entropy Markov models (MEMMs) [8]. These models predict a distribution of the tags at each time step and then use beam-like decoding to find optimal tag sequences. Lafferty et al. proposed conditional random fields (CRF) to leverage global sentence level feature and solve the label bias problem in MEMM [5]. All the linear statistical models rely heavily on hand-crafted features, e.g., the true entities usually start with capital letters. In recent years deep neural networks based models have been proposed for NER. Most of them directly combine the deep neural networks with CRF. For example, Huang et al. [4] used a bidirectional LSTM to automatically extract word-level representations and then combined with CRF for jointly label decoding. Ma and Hovy [6] introduced a neural network architecture that both word level and character level features are used, in which bidirectional LSTM, CNN, and CRF are combined. Reinforcement learning has also been proposed for the task. For example, Maes et al. [7] formalized the sequence tagging task as a Markov decision process (MDP) and used SARSA to construct optimal sequence directly in a greedy manner. Chang et al. [1] proposed a novel bandit-like strategy to search a better policy than its reference teacher in structured prediction task. Inspired by the reinforcement learning model of AlphaGO Zero [11] programs designed for the Game of Go, in this paper we solve the NER task with a Monte Carlo tree search (MCTS) enhanced Markov decision process (MDP). The new model, referred to as MM-NER (MCTS enhanced MDP for Name Entity Recognition), makes use of an MDP to model the sequential tag assignment process. At each position (corresponding to word position), based on the past words and tags, two Gated Recurrent Unit (GRUs) are used to summarize the past words and tags respectively. Based on the outputs of these two GRUs, a policy function (a distribution over the valid tags) for guiding the tag assignment and a value function for estimating the accuracy of tagging are produced. To avoid the problem of assigning tags without utilizing the whole sentence level tags, in stead of choosing a tag directly with the raw policy predicted by the policy function, MM-NER explores more possibilities in the whole space. The exploration is conducted with the MCTS guided by the produced policy function and value function, resulting a strengthened search policy for the tag assignment. Moving to the next iteration, the algorithm moves to the next position and continue the above process until at the end of the sentence.

Reinforcement learning is used to train the model parameters. In the training phase, at each learning iteration and for each training sentence (and the corresponding labels), the algorithm first conducts an MCTS inside the training loop, guided by the current policy function and value function. Then the model parameters are

* Corresponding author

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331349

adjusted to minimize the loss function. The loss function consists of two terms: 1) the squared error between the predicted value and the final ground truth accuracy of the whole sentence tagging; and 2) the cross entropy of the predicted policy and the search probabilities for tags selection. Stochastic gradient descent is utilized for conducting the optimization.

To evaluate the effectiveness of MM-NER, we conducted experiments on the basis of CoNLL 2003 NER dataset. The experimental results showed that MM-NER can outperform the baselines of BLSTM, BLSTM-CNN, and CRF, BLSTM-CRF with random initialization. The results also showed that MM-NER performed almost equally well with the state-of-the-art baseline of BLSTM-CRF with Glove initialization. We analyzed the results and showed that MM-NER achieved the performances through conducting lookahead MCTS to explore in the whole tagging space.

2 MDP FORMULATION OF NER

2.1 NER as an MDP

Suppose that $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$ is a sequence of words to be labeled for NER, and $\mathbf{Y} = \{y_1, \cdots, y_M\}$ is the corresponding ground truth NER tag sequence. All components $\mathbf{x}_i$ of $\mathbf{X}$ are the L-dimensional preliminary representations of the words, i.e., the word embedding. All components y_i of $\mathbf{Y}$ are assumed to be selected from possible NER tags set $\mathcal{Y}$. The goal is to learn a model that can automatically assign a tag to each word in the input sentence $\mathbf{X}$.

MM-NER formulates the assignment of tags to sentences as a process of sequential decision making with an MDP in which each time step corresponds to a position in the sentence. The states, actions, transition function, value function, and policy function of the MDP are set as:

States S**:** We design the state at time step t as a triple $s_t = [\mathbf{X}_t^l = \{\mathbf{x}_{t-w}, \cdots, \mathbf{x}_t\}, \mathbf{X}_t^r = \{\mathbf{x}_t, \cdots, \mathbf{x}_{t+w}\}, \mathbf{Y}_t = \{\mathbf{y}_1, \cdots, \mathbf{y}_{t-1}\}]$ where $\mathbf{X}_t^l$ and $\mathbf{X}_t^r$ are the sequences of the left context window and right context window of the input sentence with length M. w is the window size and $\mathbf{Y}_t$ is the prefix of the label sequence of length $t-1$. At the beginning ($t = 1$), the state is initialized as $s_1 = [\{\mathbf{x}_1\}, \{\mathbf{x}_1, \cdots, \mathbf{x}_{1+w}\}, \emptyset]$, where $\emptyset$ is the empty sequence.

Actions $\mathcal{A}$**:** At each time step t, the $\mathcal{A}(s_t) \subseteq \mathcal{Y}$ is the set of actions the agent can choose. That is, the action $a_t \in \mathcal{A}(s_t)$ actually is a tag $y_t \in \mathcal{Y}$ for word $\mathbf{x}_t$.

Transition function T**:** $T : S \times \mathcal{A} \rightarrow S$ is defined as

$$s_{t+1} = T(s_t, a_t) = T([\mathbf{X}_t^l, \mathbf{X}_t^r, \mathbf{Y}_t], a_t) = [\mathbf{X}_{t+1}^l, \mathbf{X}_{t+1}^r, \mathbf{Y}_t \oplus \{a_t\}]$$

where $\oplus$ appends a_t to $\mathcal{Y}_t$. At each time step t, based on state s_t the system chooses an action (tag) a_t for the word position t. Then, the system moves to time step $t+1$ and the system transits to a new state s_{t+1}: first, the left and right context window $\mathbf{X}_t^l, \mathbf{X}_t^r$ are updated by moving its window to obtain $\mathbf{X}_{t+1}^l, \mathbf{X}_{t+1}^r$; second, the system appends the selected tag to the end of $\mathbf{Y}_t$, generating a new tag sequence.

Value function V**:** The state value function $V : S \rightarrow \mathbb{R}$ is a scalar evaluation, predicting the accuracy of the tag assignments for the whole sentence (an episode), on the basis of the input state. The value function is learned so as to fit the real tag assignment accuracy of the training sentences.

In this paper, we use two GRUs and one-layer MLP to respectively map the left and right context window $\mathbf{X}_t^l, \mathbf{X}_t^r$ in the state s_t to two real vectors, and then define the value function as nonlinear transformation of the weighted sum of the MLP's outputs $\mathbf{g}(s)$ and current candidate action in one-hot representation $\mathbf{a}_t$:

$$V(s) = \sigma\left(\langle \mathbf{W}_v \mathbf{g}(s), \mathbf{a}_t \rangle\right) \tag{1}$$

where $\mathbf{W}_v \in R^{|\mathcal{A}(s)|*|g(s)|}$ is the weight vector to be learned during training, $|g(s)|$, $|\mathcal{A}(s)|$ are the dimension of MLP's output and the number of possible action, $\langle \cdot, \cdot \rangle$ is dot product operation, $\sigma(x) = \frac{1}{1+e^{-x}}$ is the nonlinear sigmoid function.

$$\mathbf{l}(s) = \left[\mathrm{GRU}^{\mathrm{l}}(\mathbf{X}_t^l)^T, \mathrm{GRU}^{\mathrm{r}}(\mathbf{X}_t^r)^T\right]^T, \mathbf{g}(s) = \mathbf{W}_g \mathbf{l}(s) + \mathbf{b}_g \tag{2}$$

The two GRU networks are defined as follows: given $s_t = [\mathbf{X}_t^l = \{\mathbf{x}_{t-w}, \cdots, \mathbf{x}_t\}, \mathbf{X}_t^r = \{\mathbf{x}_t, \cdots, \mathbf{x}_{t+w}\}, \mathbf{Y}_t = \{\mathbf{y}_1, \cdots, \mathbf{y}_{t-1}\}]$, where $\mathbf{x}_k (k = 1, \cdots, t)$ is the word at k-th position, represented with its word embedding. GRU^l outputs a representation h_k for position k:

$$\begin{aligned}
\mathbf{z}_k &= \sigma(\mathbf{W}_z \mathbf{x}_k + \mathbf{U}_z \mathbf{h}_{k-1}), \mathbf{r}_k = \sigma(\mathbf{W}_r \mathbf{x}_k + \mathbf{U}_r \mathbf{h}_{k-1}), \\
\tilde{\mathbf{h}}_k &= \tanh(\mathbf{W}_h \mathbf{x}_k + \mathbf{U}_z(\mathbf{f}_k \circ \mathbf{c}_{k-1})), \\
\mathbf{h}_k &= (1 - \mathbf{z}_k) \circ \tanh(\mathbf{c}_k) + \mathbf{z}_t \circ \tilde{\mathbf{h}}_k
\end{aligned}$$

where $\mathbf{h}$ and $\mathbf{c}$ are initialized with zero vector; operator "$\circ$" denotes the element-wise product and "σ" is applied to each of the entries. The last hidden state vector is used as the output of GRU, that is $\mathrm{GRU}^{\mathrm{l}}(\mathbf{X}_t^l) = \mathbf{h}_{1+w}{}^T$.

The function $\mathrm{GRU}^{\mathrm{r}}(\mathbf{X}_t^r)$, which is used to map the right context $\mathbf{X}_t^r$ into a real vector, is defined similarly to that of for $\mathrm{GRU}^{\mathrm{l}}(\mathbf{X}_t^l)$.

Policy function p: The policy $\mathbf{p}(s)$ defines a function that takes the state as input and output a distribution over all of the possible actions $a \in \mathcal{A}(s)$. Specifically, each probability in the distribution is a normalized softmax function whose input is the bilinear product of the state representation in Equation (2):

$$p(a|s) = \mathrm{softmax}(\mathbf{U}_p \mathbf{g}(s)),$$

where $\mathbf{U}_p \in R^{|\mathcal{A}(s)|*|g(s)|}$ is the parameter. The policy function is:

$$\mathbf{p}(s) = \langle p(a_1|s), \cdots, p(a_{|\mathcal{A}(s)|}|s) \rangle. \tag{3}$$

2.2 Strengthening raw policy with MCTS

Selecting the NER tags directly with the predicted raw policy $\mathbf{p}$ in Equation (3) may lead to suboptimal results because the policy is calculated based on the past tags. The raw policy has no idea about the tags that will be assigned for the future words. To alleviate the problem, following the practices in AlphaGo Zero [11], we propose to conduct lookahead search with MCTS. That is, at each position t, an MCTS search is executed, guided by the policy function $\mathbf{p}$ and the value function V, and output a strengthened new search policy $\boldsymbol{\pi}$. Usually, the search policy $\boldsymbol{\pi}$ has high probability to select a tag with higher accuracy than the raw policy $\mathbf{p}$ defined in Equation (3).

Algorithm 1 shows the details of the MCTS in which each tree node corresponds to an MDP state. It takes a root node s_R, value function V and policy function $\mathbf{p}$ as inputs. The algorithm iterates K times and outputs a strengthened search policy $\boldsymbol{\pi}$ for selecting a tag for the root node s_R. Suppose that each edge $e(s, a)$ (the edge from state s to the state $T(s, a)$) of the MCTS tree stores an action value $Q(s, a)$, visit count $N(s, a)$, and prior probability $P(s, a)$. At each of the iteration, the MCTS executes the following steps:

Selection: Each iterations starts from the root s_R and iteratively selects the tags that maximize an upper confidence bound:

$$a_t = \arg\max_a (Q(s_t, a) + \lambda U(s_t, a)), \quad (4)$$

where $\lambda > 0$ is the tradeoff coefficient, and the bonus $U(s_t, a) = p(a|s_t)\frac{\sqrt{\sum_{a' \in \mathcal{A}(s_t)} N(s_t, a')}}{1+N(s_t,a)}$. $U(s_t, a)$ is proportional to the prior probability but decays with repeated visits to encourage exploration.

Evaluation and expansion: When the traversal reaches a leaf node s_L, the node is evaluated with the value function $V(s_L)$ (Equation (1)). Note following the practices in AlphaGo Zero, we use the value function instead of rollouts for evaluating a node.

Then, the leaf node s_L may be expanded. Each edge from the leaf position s_L (corresponds to each action $a \in \mathcal{A}(s_L)$) is initialized as: $P(s_L, a) = p(a|s_L)$ (Equation (3)), $Q(s_L, a) = 0$, and $N(s_L, a) = 0$. In this paper all of the available actions of s_L are expanded.

Back-propagation and update: At the end of evaluation, the action values and visit counts of all traversed edges are updated. For each edge $e(s, a)$, the prior probability $P(s, a)$ is kept unchanged, and $Q(s, a)$ and $N(s, a)$ are updated:

$$Q(s, a) \leftarrow \frac{Q(s, a) \times N(s, a) + V(s_L)}{N(s, a) + 1}; N(s, a) \leftarrow N(s, a) + 1. \quad (5)$$

Calculate the strengthened search policy: Finally after iterating K times, the strengthened search policy $\boldsymbol{\pi}$ for the root node s_R can be calculated according to the visit counts $N(s_R, a)$ of the edges starting from s_R:

$$\pi(a|s_R) = \frac{N(s_R, a)}{\sum_{a' \in \mathcal{A}(s_R)} N(s_R, a')}, \quad (6)$$

for all $a \in \mathcal{A}(s_R)$.

2.3 Learning and inference algorithms

2.3.1 Reinforcement learning of the parameters. The model has parameters $\boldsymbol{\Theta}$ (including $\mathbf{W}_v, \mathbf{W}_g, b_g, \mathbf{U}_p$ and parameters in GRUs) to learn. In the training phase, suppose we are given N labeled sentence $D = \{(\mathbf{X}^{(n)}, \mathbf{Y}^{(n)})\}_{n=1}^N$. Algorithm 2 shows the training procedure. First, the parameters $\boldsymbol{\Theta}$ is initialized to random weights in $[-1, 1]$. At each subsequent iteration, for each $(\mathbf{X}, \mathbf{Y})$, a tag sequence is predicted for $\mathbf{X}$ with current parameter setting: at each position t, an MCTS search is executed, using previous iteration of value function and policy function, and a tag a_t is selected according to the search policy $\boldsymbol{\pi}_t$. The tagging terminates at the end of the sentence and achieved a predicted tag sequence $(a_1, \cdots, a_M)$. Given the ground truth tag sequence $\mathbf{Y}$, the overall tagging metric of the sentence $\mathbf{X}$ is calculated, denoted as r. The data generated at each time step $E = \{(s_t, \boldsymbol{\pi}_t)\}_{t=1}^M$ and the final evaluation r are utilized as the signals in training for adjusting the value function. The model parameters are adjusted to minimize the error between the predicted value $V(s_t)$ and tagging metric r, and to maximize the similarity of the policy $\mathbf{p}(s_t)$ to the search probabilities $\boldsymbol{\pi}_t$. Specifically, the parameters $\boldsymbol{\Theta}$ are adjusted by gradient descent on a loss function ℓ that sums over the mean-squared error and cross-entropy losses, respectively:

$$\ell(E, r) = \sum_{t=1}^{|E|} \left((V(s_t) - r)^2 + \sum_{a \in \mathcal{A}(s_t)} \pi_t(a|s_t) \log \frac{1}{p(a|s_t)} \right). \quad (7)$$

Algorithm 1 TreeSearch

Input: root s_R, value V, policy $\mathbf{p}$, search times K

for $k = 0$ **to** $K - 1$ **do**
  $s_L \leftarrow s_R$
  {Selection}
  while s_L is not a leaf node **do**
    $a \leftarrow \arg\max_{a \in \mathcal{A}(s_L)} Q(s_L, a) + \lambda \cdot U(s_L, a)$ {Eq. (4)}
    $s_L \leftarrow$ child node pointed by edge (s_L, a)
  end while
  {Evaluation and expansion}
  $v \leftarrow V(s_L)$ {simulate v with value function V}
  for all $a \in \mathcal{A}(s_L)$ **do**
    Expand an edge e to $s = [s_L.\mathbf{X}_{t+1}^l, s_L.\mathbf{X}_{t+1}^r, s_L.\mathbf{Y}_t \oplus \{a\}]$
    $e.P \leftarrow p(a|s_L); e.Q \leftarrow 0; e.N \leftarrow 0$ {init edge properties}
  end for
  {Back-propagation}
  while $s_L \neq s_R$ **do**
    $s \leftarrow$ parent of s_L; $e \leftarrow$ edge from s to s_L
    $e.Q \leftarrow \frac{e.Q \times e.N + v}{e.N+1}$ {Eq. (5)}; $e.N \leftarrow e.N + 1$; $s_L \leftarrow s$
  end while
end for
for all $a \in \mathcal{A}(s_R)$ **do**
  $\pi(a|s_R) \leftarrow \frac{e(s_R, a).N}{\sum_{a' \in \mathcal{A}(s_R)} e(s_R, a').N}$
end for
return $\boldsymbol{\pi}$

Algorithm 2 Train MM-NER model

Input: Labeled data $D = \{(\mathbf{X}^{(n)}, \mathbf{Y}^{(n)})\}_{n=1}^N$, learning rate η, K

Initialize $\boldsymbol{\Theta} \leftarrow$ random values in $[-1, 1]$
repeat
  for all $(\mathbf{X}, \mathbf{Y}) \in D$ **do**
    $s_1 = [\{\mathbf{x}_1\}, \{\mathbf{x}_1, \cdots, \mathbf{x}_{1+w}\}, \{\emptyset\}]; M \leftarrow |X|; E \leftarrow \emptyset$
    for $t = 1$ **to** M **do**
      $\boldsymbol{\pi} \leftarrow$ TreeSearch$(s, V, \mathbf{p}, K)$ {Alg. (1)}
      $a = \arg\max_{a \in \mathcal{A}(s)} \pi(a|s)$ {select the best tag}
      $E \leftarrow E \oplus \{(s, \boldsymbol{\pi})\}$
      $s \leftarrow [s.\mathbf{X}_{t+1}^l, s.\mathbf{X}_{t+1}^r, s.\mathbf{Y}_t \oplus \{a\}]$
    end for
    $r \leftarrow$ Metric$(\mathbf{Y}, s.\mathbf{Y}_M)$ {overall tagging metric}
    $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \eta \frac{\partial \ell(E, r)}{\partial \boldsymbol{\Theta}}$ {ℓ is defined in Eq. (7)}
  end for
until converge
return $\boldsymbol{\Theta}$

The model parameters are trained by back propagation and stochastic gradient descent. Specifically, we use AdaGrad [3] on all parameters in the training process.

2.3.2 Inference. The inference of the NER tag sequence for a sentence is shown in Algorithm 3. Given a sentence $\mathbf{X}$, the system state is initialized as $s_1 = [\{\mathbf{x}_1\}, \{\mathbf{x}_1, \cdots, \mathbf{x}_{1+w}\}, \emptyset]$. Then, at each of the time steps $t = 1, \cdots, M$, the agent receives the state $s_t = [\mathbf{X}_t^l, \mathbf{X}_t^r, \mathbf{Y}_t]$ and search the policy $\boldsymbol{\pi}$ with MCTS, on the basis of the value function V and policy function $\mathbf{p}$. Then, it chooses an action

Algorithm 3 MM-NER Inference

Input: sentence $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$, value V, policy $\mathbf{p}$, and K,

$s \leftarrow [\{\mathbf{x}_1\}, \{\mathbf{x}_1, \cdots, \mathbf{x}_{1+w}\}, \{\emptyset\}]; M \leftarrow |\mathbf{X}|$
for $t = 1$ **to** M **do**
  $\boldsymbol{\pi} \leftarrow \text{TreeSearch}(s, V, \mathbf{p}, K)$
  $a \leftarrow \arg\max_{a \in \mathcal{A}(s)} \pi(a|s)$
  $s \leftarrow [s.\mathbf{X}^l_{t+1}, s.\mathbf{X}^r_{t+1}, s.\mathbf{Y} \oplus \{a\}]$
end for
return $s.\mathbf{Y}$

Table 1: Performance comparison for all methods .

	Precision	Recall	F1
BLSTM	80.14%	72.81%	76.29%
BLSTM-CNN	83.48%	83.28%	83.38%
CRF (random)	82.93%	79.94%	81.41%
BLSTM-CRF (random)	83.61%	84.78%	84.19%
MM-NER	84.19%	86.28%	85.22%
CRF (Glove)	85.32%	84.55%	84.93%
BLSTM-CRF (Glove)	88.57%	89.04%	88.30%

a for the word at position t. Moving to the next iteration $t+1$, the state becomes $s_{t+1} = [\mathbf{X}^l_{t+1}, \mathbf{X}^r_{t+1}, \mathbf{Y}_{t+1}]$. The process is repeated until the end of the sentence is reached. The code of MM-NER can be found at https://github.com/YadiLao/MM_Tag.

3 EXPERIMENTS

We tested the performances of MM-NER on CoNLL 2003 NER dataset[1], which contains 4 types of entities: persons (PER), organizations (ORG), locations (LOC), and miscellaneous names (MISC). In the experiments, we used the publicly available GloVe 100-dimensional embeddings as the representations of the words [9]. For MM-NER, the tag level macro $F1$ of the whole sentence is used as the tagging metric R during the training of the MM-NER model. The learning rate η, the tree search trade-off parameter λ, the number of hidden units h in GRUs, the window size w and the number of search times K. They were empirically set to $\eta = 0.01$, $\lambda = 1.0$, $h = 100$, $w = 3$, $K = 600$. Spelling features are concatenated in the state.

We reproduced models in [4] and [2] , including linear statistical model of CRF and neural models of BLSTM, BLSTM-CNN and its combination model BLSTM-CRF. Table 1 reports the performances of MM-NER and baseline methods in terms of NER precision, recall, F1. From the result we can see that, MM-NER outperformed the conventional baseline CRF with random initialization, deep learning baselines of BLSTM, BLSTM-CNN, and BLSTM-CRF with random initialization, indicating the effectiveness of the proposed MM-NER model. The reason why MM-NER underperform BLSTM-CRF with Glove initialization is that MCTS is more important than embedding for guiding the agent to search a better tagging result.

One of the key steps in MM-NER is using MCTS to improve the raw policy $\boldsymbol{p}$ of MDP. It is likely that the search policy $\boldsymbol{\pi}$ is better than the raw policy $\boldsymbol{p}$. We conducted experiments in inference stage to test the effects that raw policy $\boldsymbol{p}$, value network $\boldsymbol{v}$ and search policy $\boldsymbol{\pi}$ have on CoNLL 2003. For raw policy $\boldsymbol{p}$, value network $\boldsymbol{v}$, agent will greedily choose action with highest probability or value at each time step. As illustrated in table 2, we can see search policy $\boldsymbol{\pi}$ achieves highest F1. If we ignore the small F1 difference between $\boldsymbol{\pi}$ and $\boldsymbol{p}$, we can greatly reduce the time complexity to $O(M * |\mathcal{A}|)$, which is more efficient than Viterbi decoding whose time complexity is $O(M * |\mathcal{A}|^2)$. Since the goodness of the search policy $\boldsymbol{\pi}$ rely on the search times K, it is possible to make a trade-off between the tagging result and efficiency.

[1]https://www.clips.uantwerpen.be/conll2003/ner/

Table 2: F1 w.r.t different search times k. F1 scores for the raw policy p and value function v are also shown.

	100	300	600	1000	1500	**p**	**v**
F1	85.11%	85.16%	85.20%	85.22%	85.22%	85.17%	83.11%

4 CONCLUSION

In this paper we have proposed a novel approach to named entity recognition, referred to as MM-NER. MM-NER formalizes NER for a sentence as sequential decision making with MDP. The lookahead MCTS is used to strengthen the raw predicted policy so that the search policy has high probability to select the correct tags for each word. Reinforcement learning is utilized to train the model parameters. MM-NER enjoys several advantages: tagging with the shared policy and the value functions, end-to-end learning, and low time complexity in inference. Experimental results show that MM-NER outperformed or performed equally well with the baselines of CRF, BLSTM, BLSTM-CNN, and BLSTM-CRF. Future work includes improving the effectiveness and efficiency of MCTS.

5 ACKNOWLEDGEMENT

This work was funded by National Natural Science Foundation of China (61872338 and 61702047), Beijing Natural Science Foundation (4174098), Fundamental Research Funds for the Central Universities, and Research Funds of Renmin University of China (2018030246).

REFERENCES

[1] Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume III, and John Langford. 2015. Learning to search better than your teacher. In *ICML* (2015).
[2] Jason P. C. Chiu and Eric Nichols. 2016. Named Entity Recognition with Bidirectional LSTM-CNNs. *TACL*, 4, 357–370 (2016).
[3] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* 12, Jul (2011), 2121–2159.
[4] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).
[5] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML* (2001).
[6] Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354* (2016).
[7] Francis Maes, Ludovic Denoyer, and Patrick Gallinari. 2007. Sequence labeling with reinforcement learning and ranking algorithms. In *ECML*, 648–657 (2007).
[8] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. 2000. Maximum Entropy Markov Models for Information Extraction and Segmentation.. In *ICML*, 591–598, (2000).
[9] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543. (2014)
[10] Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*. (1996)
[11] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354.