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,
Beijing, China

Problem

Open Information Extraction (**OIE**):

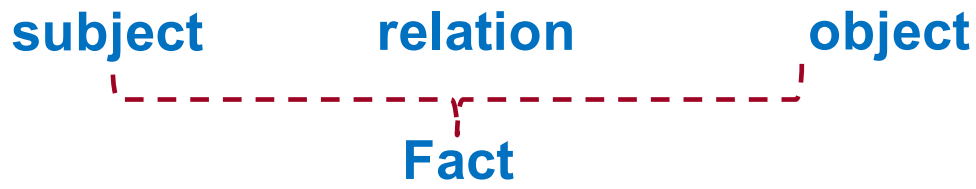
- Three main property: 1) **unlexicalized**, 2) **domain-independent** 3) scales to the **diversity and size of** the web corpus [1], for example:

Source Sentence:

Mencius has followed the example of Confucius, and led disciples to lobby the nations.

Target Sequence (contains three facts):

(Mencius $ follow the example of $ Confucius) (Mencius $ lead $ disciples) (Mencius $ lobby $ nations)



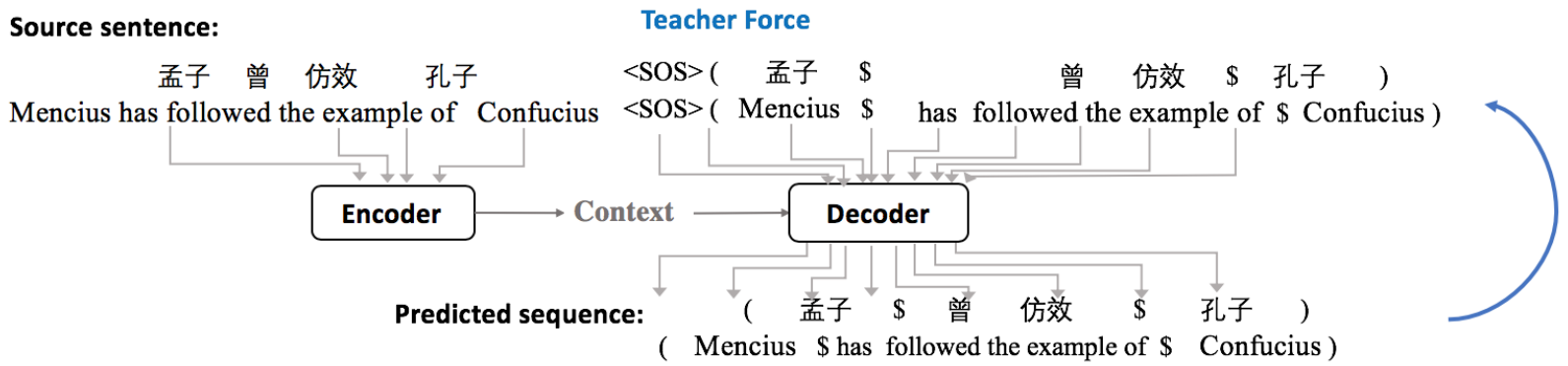
- In this work, we solve OIE as a task of **sequence prediction:** source sentence -> target sequence.

[1] Michele Banko, et,al. Open Information Extraction from the Web. In Proceedings of IJCAI. 2007

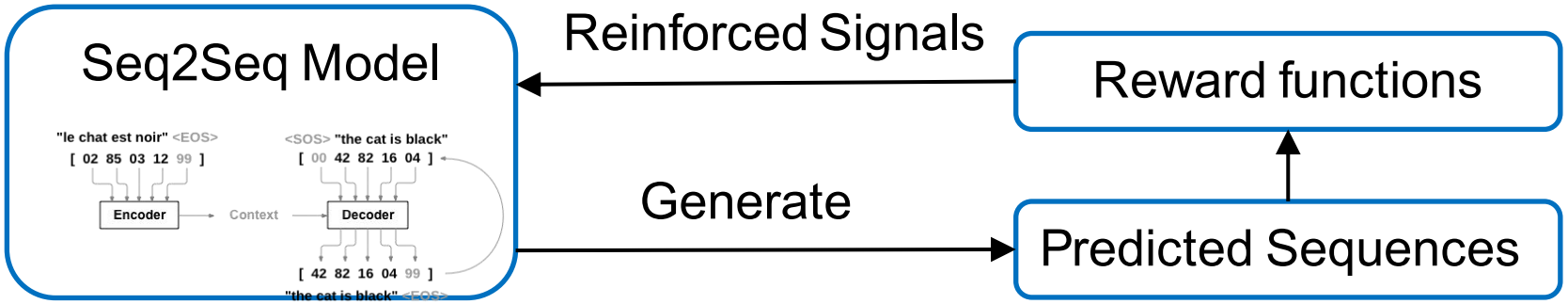
Motivation

Previous works on OIE:

- Sequence-to-Sequence (**Seq2Seq**) model: Supervised Learning.



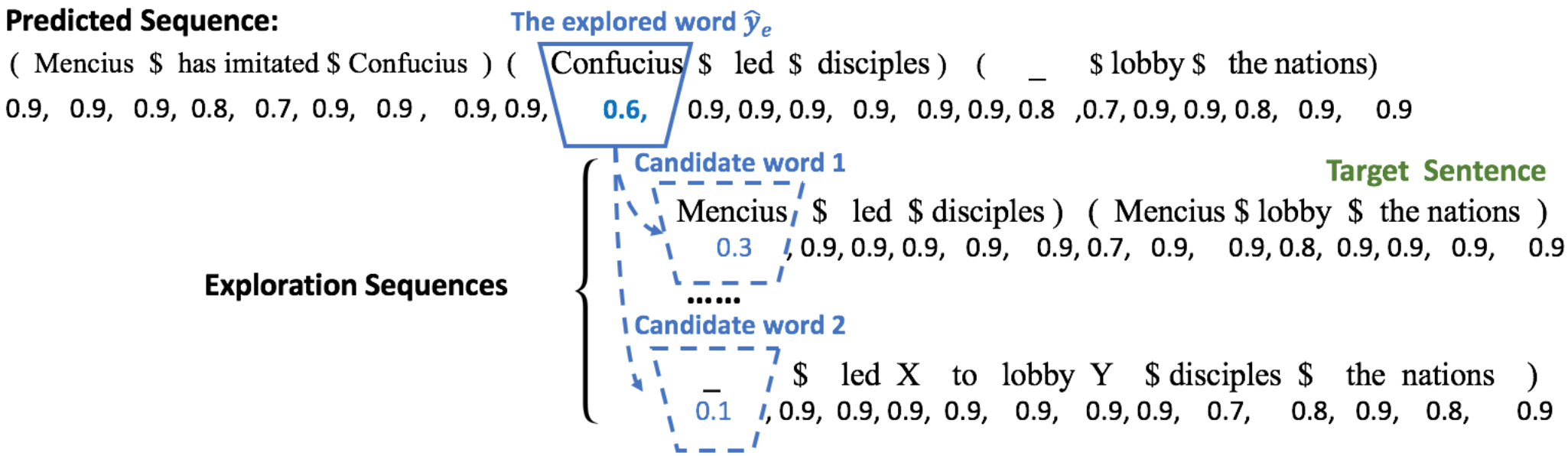
- **Reinforced** Algorithm: Semi-supervised Learning.



Motivation

Common Issues of previous works:

- **1) No look ahead** and **2) limited exploration** during prediction.
- Why so important? *A previously predicted word (sub-optimal) will significantly influence the following predictions*, for example:



Model

Monte Carlo Tree Search (MCTS):

- Run Monte Carlo simulations on a Upper Confidence Tree (UCT). At each node (state s_t:source sentence + predictions until t-1), we select an edge (action a_t: word at t) by:

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

  a) Q function $Q(s_t, a_t)$ learns expected rewards (provides **look ahead**).

  b) c_{puct} controls the scale of exploration. It encourages selecting the unvisited nodes (provides **exploration**).

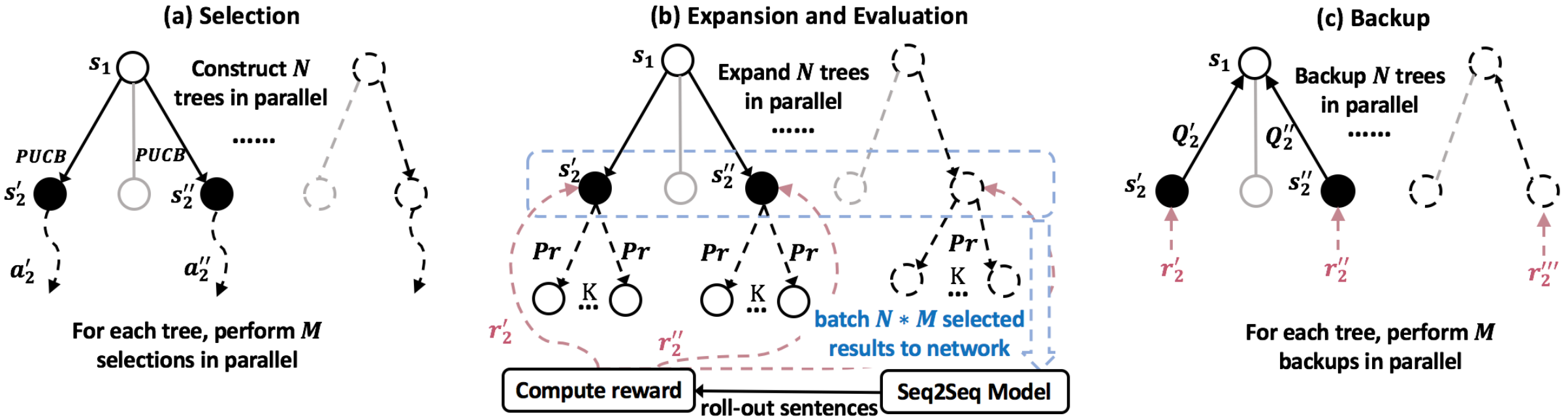
- Run MCTS by:

  a) Implementing a play (simulation) recording four phases: selection, expansion, evaluation and backup (parallelization for acceleration).

  b) Determine the predicted words from the distribution of visited number.

Model

Monte Carlo Tree Search (MCTS):



- Run MCTS by:
  a) Implementing a play (simulation) recording four phases: selection, expansion, evaluation and backup (parallelization for acceleration).
  b) Determine the predicted words from the distribution of visited number.

Model

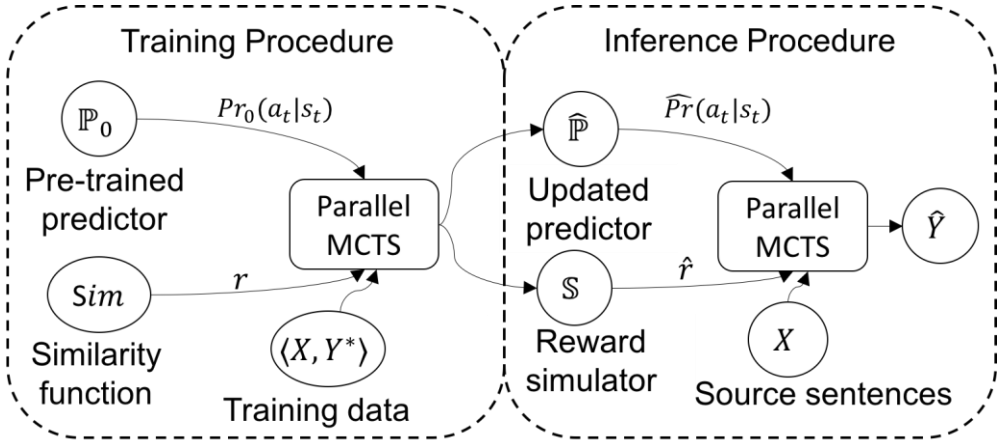
Reward Simulator:

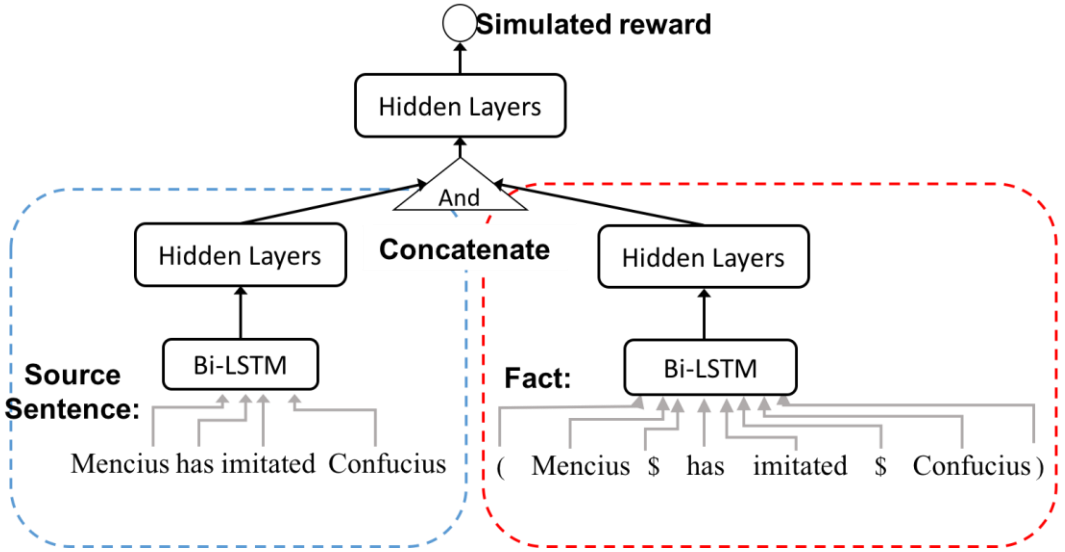
- Learn (Mimic) the reward signals.
  a) Training rewards: similarity between predicted facts and target facts.

$$Sim(\hat{Y}_{1..t}, Y^*) = \sum_{l=1}^{\min(N_p, N_G)} \eth((\hat{F}_i, F_j^*)_l)$$

  b) During testing, target facts are unavailable and requires a reward simulator:

- Defines a complete RL environment





Experiment

Running Setting:

- Dataset: SAOKE[1] contains over 47,000 source-target sequences. (manually labeled by crowdsourcing)
- Comparison Methods:
  a) DSNF and CORE (pattern matching)
  b) Logician (Seq2Seq model for OIE.)
  c) Reinforced algorithm.
  d) Dual structured reinforced learning.
  e) Search method: (greedy and beam)

[1] http://ai.baidu.com/broad/subordinate?dataset=saoke

Results:

- Label a extracted fact as 1 if it is correct (sim>0.85, ≈target fact) and 0 otherwise.
- Compute Precision, Recall and F1

	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**

Table 1: Evaluation results for the predicted facts.

Conclusion

Take home messages from presenter:

- If no explore, add exploration.
- If don't know how to explore, try Monte-Carlo Method (e.g. UCT model).
- If MC gives you huge time complexity, remember it is often parallelizable.
- If it is still unaffordable (inputs are images), check our future work.

Future Work:

- Representation Learning, for example:
  a) Data disentangling, map input (images under the curse of dimensionality) to independent factors of variation by Variational methods (VAE).

THANK YOU!