Encoder-Decoder Shift-Reduce Syntactic Parsing

Jiangming Liu and Yue Zhang

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Established in collaboration with MIT

Presented by:

Xavier Carreras

Transition-based syntactic parsing

Transition-based syntactic parsing system maintains a stack (**S**), which contains partially-constructed outputs and a queue (**Q**), which contains ordered incoming words.

At each step, a transition action is taken to consume the input and construct the output. When the syntactic tree is completed, a sequence of actions (**A**) is achieved.

Input sentence -> sequence of actions -> syntactic tree

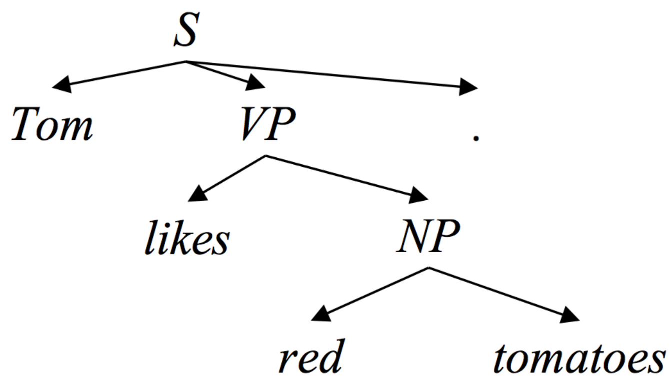
Transition-based syntactic parsing

- Constituent parsing
  - Constituent trees are represented as the sequences of transition actions. e.g. the **top-down system** (Dyer et al., 2016).

- Dependency parsing
  - Dependency trees are represented as the sequences of transition actions. e.g. the **arc-standard system** (Nivre et al., 2007).

Transition-based syntactic parsing

- Constituent parsing



(a) Constituent tree

Initial State $(\phi, Q, 0)$

Final State $(s_0, \phi, 0)$

Induction Rules:

SHIFT $$\frac{(S, q_0 | Q, n)}{(S | q_0, Q, n)}$$

NT(X) $$\frac{(S, Q, n)}{(S | e(x), Q, n+1)}$$

REDUCE $$\frac{(S | e(x) | s_j | \ldots | s_0, Q, n)}{(S | e(x, s_j, \ldots, s_0), Q, n-1)}$$

– Inputs:

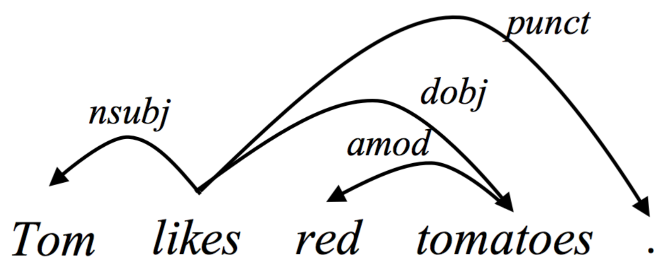
Tom likes red tomatoes .

– Actions:

NT(S) Shift NT(VP) Shift NT(NP) Shift Shift Reduce Reduce Shift Reduce

Transition-based syntactic parsing

- Dependency parsing



(b) Dependency tree

Initial State	(ϕ, Q, ϕ)
Final State	(s_0, ϕ, L)

Induction Rules:

SHIFT	$\dfrac{(S, q_0 \vert Q, L)}{(S \vert q_0, Q, L)}$
LEFT-ARC(L)	$\dfrac{(S \vert s_1 \vert s_0, Q, L)}{(S \vert s_0, Q, L \cup s_1 \leftarrow s_0)}$
RIGHT-ARC(L)	$\dfrac{(S \vert s_1 \vert s_0, Q, L)}{(S \vert s_1, Q, L \cup s_1 \rightarrow s_0)}$

– Inputs:

Tom likes red tomatoes .

– Actions:

Shift Shift Left-Arc(nsubj) Shift Shift Left-Arc(amod) Right-Arc(dobj) Shift Right-Arc(punct)

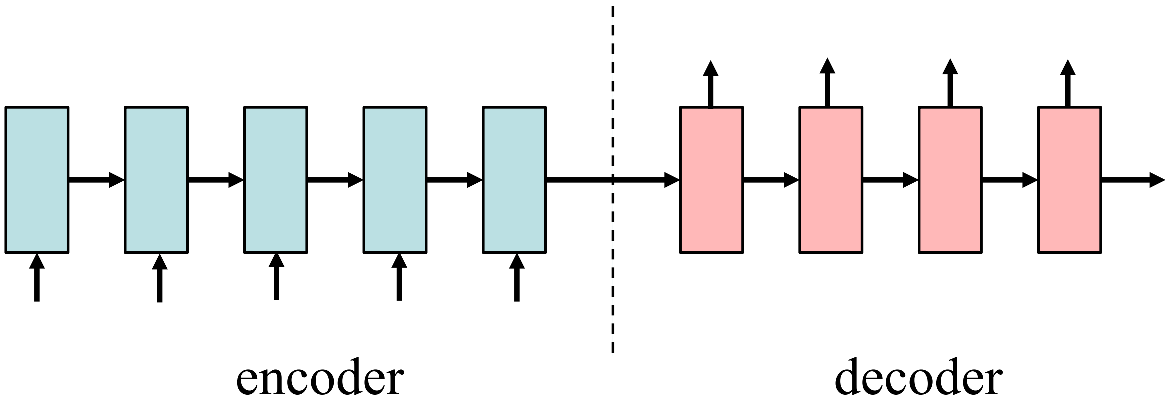
Transition-based syntactic parsing

- Generalization
  - A general sequence-to-sequence task
  - Given a sentence $x_1, x_2, \ldots, x_n$, the goal is to generate a corresponding sequence of actions $a_1, a_2, \ldots, a_m$.
  - Possible for other transition-based systems

Motivation

- Encoder-decoder neural networks
  - Encoder --- recurrent neural networks to represent sentences
  - decoder --- recurrent neural networks to output what we want in sequence.

  - Simple structures

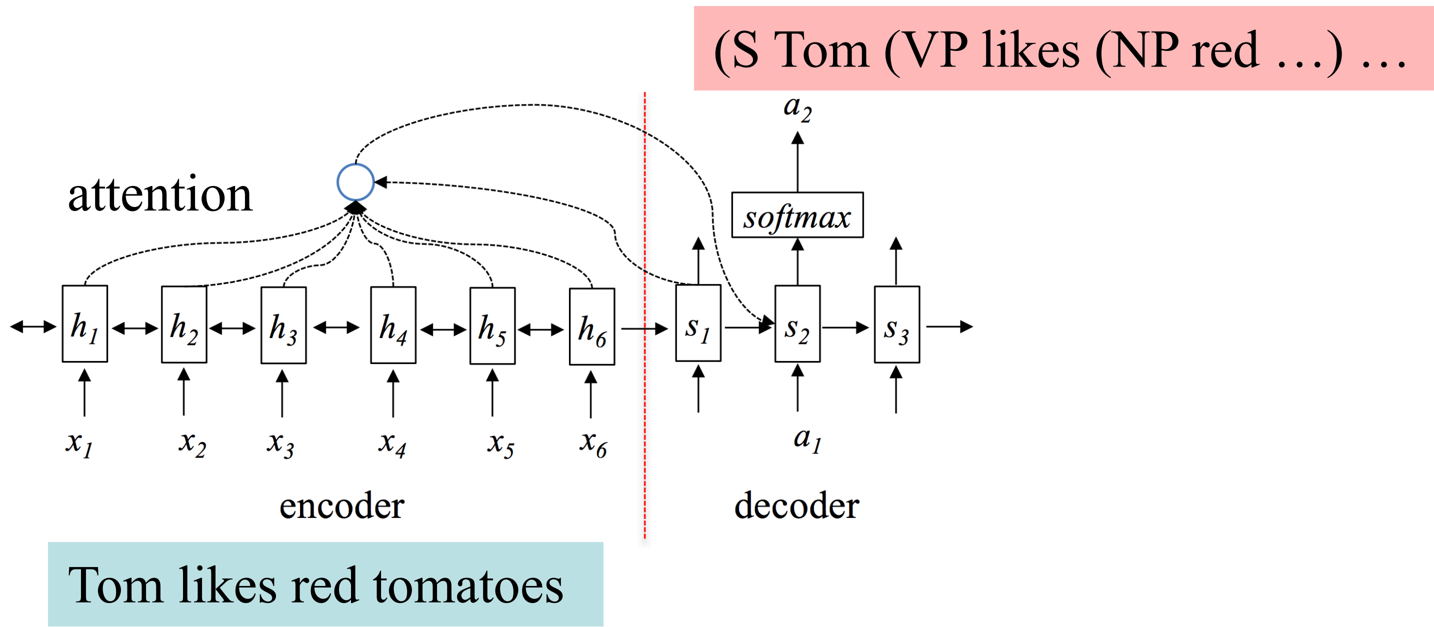


Motivation

- Neural machine translation with encoder-decoder neural networks (Bahdanau et al., 2015)
  - The encoder is used to represent source-side sentences (e.g. in French), the decoder outputs target-side sentences (e.g. in English)
  - The successful seminal work by applying the attention mechanism over the encoder

Motivation

- Constituent parsing with encoder-decoder neural networks (Vinyals et al., 2015)



  – Achieves relatively low accuracies on standard benchmarks by **translating a sentence into its bracketed representation**

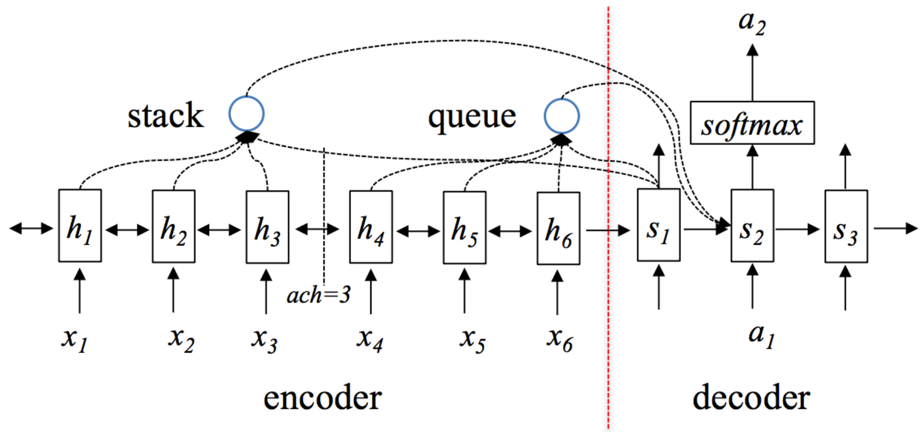
Models

- Simple encoder-decoder structure
  - **translating a sentences into its sequence of transition actions**

- Difference from Vinyals et al., (2015)
  - Instead of bracketed representation, the model outputs sequences of transition actions.
  - Instead of vanilla attention over the whole sentence, stack-queue sensitive attention mechanism is applied.
  - Same encoder, different decoder

Models

- Stack-queue decoder



We use separate attention models over encoder hidden states to represent the stack and the queue, respectively.

Note: *x* is the representation of a word, consisting of the tuned word embedding, the tuned POS embedding and the fixed pretrained word embedding.

Models

- Difference from Dyer et al., (2015 & 2016)

  - Instead of stack-LSTM, bidirectional LSTM is used for encoder.

  - Instead of changing the representation of the sentence, our system implicitly models stack information by using stack-queue sensitive attention mechanism.

Transition-based syntactic parsing

- Training

Our models are trained to minimize a cross-entropy loss objective with l2-regularization term, defined by

$$L(\theta) = -\sum_i \sum_j log\ p_{a_{ij}} + \frac{\lambda}{2}||\theta||^2,$$

where θ is the set of parameters, $p_{a_{ij}}$ is the probability of the jth action in the ith training example given by the model and λ is a regularization hyper-parameter. $\lambda = 10^{-6}$. We use stochastic gradient descent with Adam (Kingma and Ba, 2015) to adjust the learning rate.

Experiments

- Data

  – WSJ sections of PTB for constituent parsing and dependency parsing, where sections 02-21 are taken for training, section 22 for development and section 23 for test data.

  – For dependency parsing, the constituent trees are convert to Stanford dependencies (v3.3.0).

  – Pretrained word embeddings are trained on the AFP portion of English Gigaword.

Experiments

- Dependency parsing
  - Development results of dependency parsing

Model	UAS (%)
Dyer et al. (2015)	92.3
Vanilla decoder	88.5
SQ decoder + average pooling	91.9
SQ decoder + attention	92.4

Vanilla decoder: vanilla attention

SQ decoder: stack-queue sensitive encoder

+ average pooling: use pooling to represent stack and queue, respectively.

+ attention: use stack-queue sensitive attention (our model).

Experiments

- Dependency parsing
  - Final results

Model	UAS (%)	LAS (%)
Graph-based		
Kiperwasser and Goldberg (2016)	93.0	90.9
Dozat and Manning (2017)	95.7	94.1
Transition-based		
Chen and Manning (2014)	91.8	89.6
Dyer et al. (2015)	93.1	90.9
Kiperwasser and Goldberg (2016)†	93.9	91.9
Andor et al. (2016)	92.9	91.0
Andor et al. (2016)*	94.6	92.8
SQ decoder + attention	93.1	90.1

Table 3: Results for dependency parsing, where * use global training, † use dynamic oracle.

Experiments

- Dependency parsing
  - Analysis

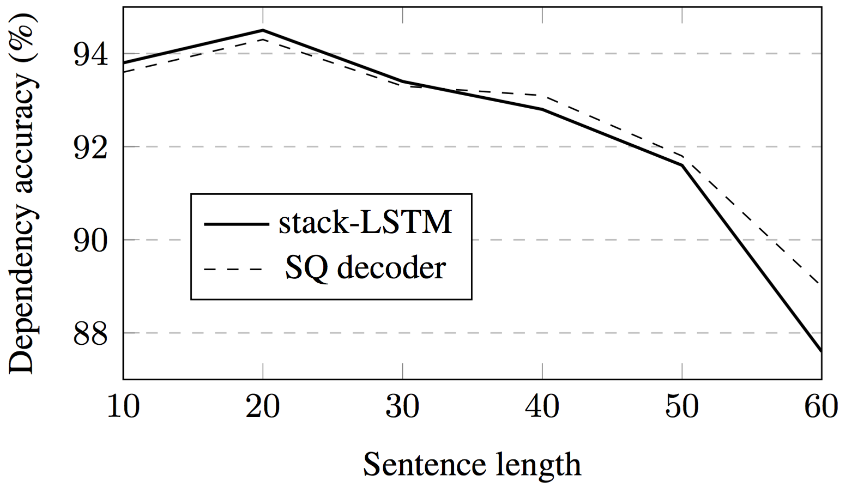


Figure 5: Accuracy against sentence length in bins of size 10, where 20 contains sentences with length [10, 20).

The composition function is applied in the stack-LSTM parser to explicitly represent the partially-constructed trees, ensuring high precision of short sentences. On the other hand, errors are also fully represented and accumulated in long sentences.

Experiments

- Dependency parsing
  - Analysis

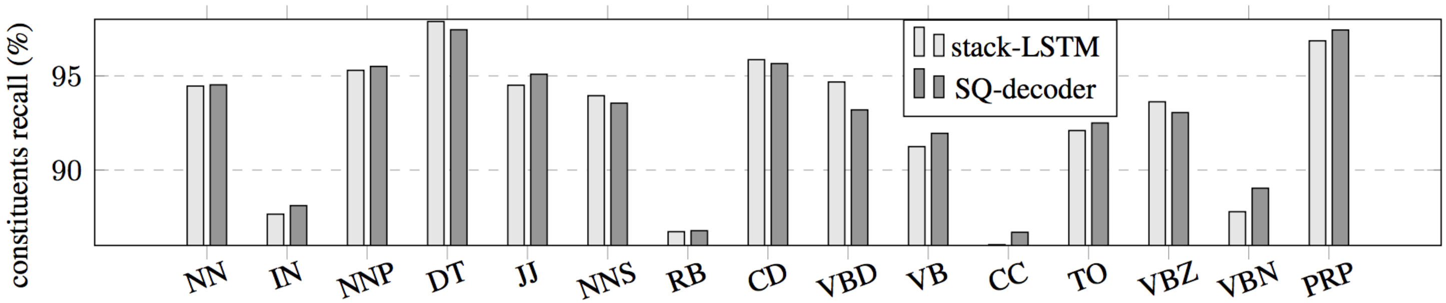


Figure 6: Accuracy against part-of-the-speech tags.

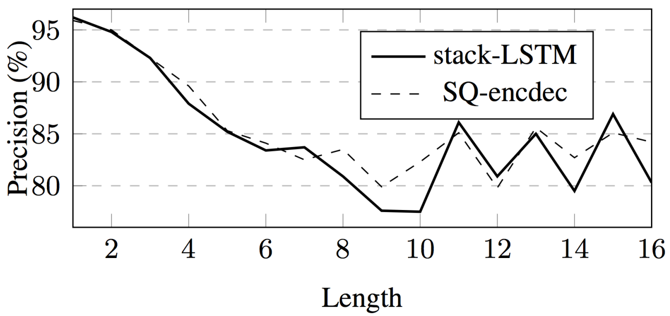


Figure 7: Arc precision against dependency length. The length is defined as the absolute difference between the indices of the head and modifier.

While the error distributions of the two parsers on the fine-grained metrics are slightly different, the main trends of the two models are consistent, which shows that our model can learn similar information compared to the parser of Dyer et al. (2015), without explicitly modeling stack information.

Experiments

- Constituent parsing
  - Final results

Model	F1 (%)
Vinyals et al. (2015)	88.3
Socher et al. (2013)	90.4
Zhu et al. (2013)	90.4
Shindo et al. (2012)	91.1
Dyer et al. (2016)	91.2
Liu and Zhang (2017b)	91.7
Liu and Zhang (2017a)	91.8
Choe and Charniak (2016) + rerank	92.4
Dyer et al. (2016) + rerank	93.3
Liu and Zhang (2017a) + rerank	93.6
SQ decoder + attention	90.5
SQ decoder + attention + rerank	92.7
SQ decoder + attention + semi-rerank	93.4

Table 4: Results for constituent parsing.

+ rerank / + semi-rerank: we use sampling techniques to get 100 candidate from our models, and use Choe and Charniak (2016) as our reranker.

Experiments

- Attention visualization

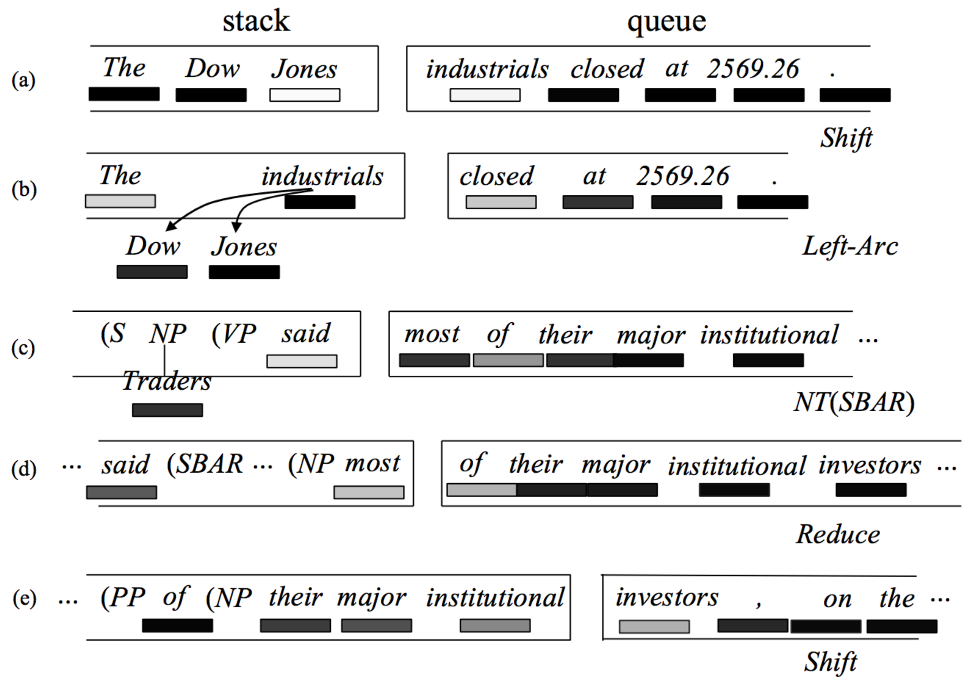


Figure 8: Output examples to visualize attention values. The grey scale indicates the value of the attention. (a) (b) are for dependency parsing, and (c) (d) (e) are for constituent parsing.

Contribution

- Study the effectiveness of the highly simple encoder-decoder structure for transition-based parsing.

- Without changing encoder-decoder structure, propose a stack-queue sensitive attention mechanism for transition-based parsing.

    - Great improvement compared to vanilla decoder (Vinyals et al. 2015)

    - Simpler and more general for different grammar formalisms without redesigning the stack representation, compared to stack-LSTM (Dyer et al., 2015 & 2016)

Contribution

– The proposed system achieves comparable results.

– Great potential by regarding the parsing task as translating a sentence into a shift-reduce action sequence, so that NMT techniques can be directly applied.

Reference

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *EMNLP*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. *ICLR* .

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *NAACL*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* .

Reference

Jiangming Liu and Yue Zhang. 2017a. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics. Accepted.*

Jiangming Liu and Yue Zhang. 2017b. Shift-Reduce Constituent Parsing with Neural Lookahead Features. *Transactions of the Association for Computational Linguistics* 5:45–58.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gu ̇lsen Eryigit, Sandra Kubler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(02):95–135.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *ACL*. Association for Computational Linguistics, pages 440–448.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *ACL*. pages 455–465.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *ICLR*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *ACL*. pages 434–443.

Q & A

The codes are public on

https://github.com/LeonCrashCode/Encoder-Decoder-Parser.