Encoder-Decoder Shift-Reduce Syntactic Parsing

Jiangming Liu and Yue Zhang

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).
Constituent parsing

- Inputs:
  Tom likes red tomatoes .
- Actions:

  \[ \text{NT}(S) \quad \text{Shift} \quad \text{NT}(VP) \quad \text{Shift} \quad \text{NT}(NP) \quad \text{Shift} \quad \text{Shift} \quad \text{Reduce} \quad \text{Reduce} \quad \text{Shift} \quad \text{Shift} \quad \text{Reduce} \]
Transition-based syntactic parsing

- **Dependency parsing**

- **Inputs:**
  
  \( \text{Tom likes red tomatoes .} \)

- **Actions:**

  \( \text{Shift Shift Left-Arc(nsubj) Shift Shift Left-Arc(amod) Right-Arc(dobj) Shift Right-Arc(punct)} \)
Transition-based syntactic parsing

- Generalization
  - A general sequence-to-sequence task
  - Given a sentence $x_1, x_2, ..., x_n$, the goal is to generate a corresponding sequence of actions $a_1, a_2, ..., a_m$.
  - 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 \( \theta \) is the set of parameters, \( p_{a_{ij}} \) is the probability of the \( j \)th action in the \( i \)th training example given by the model and \( \lambda \) 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

<table>
<thead>
<tr>
<th>Model</th>
<th>UAS (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dyer et al. (2015)</td>
<td>92.3</td>
</tr>
<tr>
<td>Vanilla decoder</td>
<td>88.5</td>
</tr>
<tr>
<td>SQ decoder + average pooling</td>
<td>91.9</td>
</tr>
<tr>
<td>SQ decoder + attention</td>
<td>92.4</td>
</tr>
</tbody>
</table>

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

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

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

   ![Graph showing precision against dependency length](chart1.png)

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

   ![Graph showing accuracy against part-of-the-speech tags](chart2.png)

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

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

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

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


Reference


The codes are public on https://github.com/LeonCrashCode/Encoder-Decoder-Parser.