Sequence-to-sequence Models for Cache Transition Systems

Xiaochang Peng\textsuperscript{1}, Linfeng Song\textsuperscript{1}, Daniel Gildea\textsuperscript{1} and Giorgio Satta\textsuperscript{2}
AMR

- “John wants to go”
After its competitor invented the front loading washing machine, the CEO of the American **IM company** believed that each of its employees had the ability for innovation, and formulated strategic countermeasures for innovation in the industry.
Transition-based AMR parsing

- There has been previous work (Sagae and Tsujii; Damonte et al.; Zhou et al.; Ribeyre et al.; Wang et al.) on transition-based graph parsing.

- Our work introduces a new data structure “cache” for generating graphs of certain treewidth.
Introduction to treewidth

A tree: treewidth 1

Complete graph of N nodes: treewidth N-1

Treewidth 2
Introduction to treewidth

small tree width
~ 2.8 on average

large tree width
Tree decomposition

graph

tree decomposition
Cache transition system

- Configuration $c = (\sigma, \eta, \beta, E)$
  - Stack $\sigma$: place for temporarily storing concepts
  - Cache $\eta$: working zone for making edges, fixed size corresponding to the treewidth.
  - Buffer $\beta$: unprocessed concepts
  - $E$: set of already-built edges
Cache transition system

- **Actions**
  - **SHIFT PUSH(i):** shift one concept from buffer to right-most position of cache, then select one concept (index i) from cache to stack.

```
stack         cache         buffer
(\$1)      \$ $ $       PER want-01 go-01

SHIFT  PUSH(1)

stack         cache         buffer
(\$,1)       \$ $ PER      want-01 go-01
```
Cache transition system

- **Actions**
  - POP: pop the top from stack and put back to cache, then drop the right-most item from cache.

```
stack     cache     buffer
($, 1)    $ $ PER   want-01 go-01
```

```
stack     cache     buffer
           $ $ $     want-01 go-01
```
Cache transition system

- **Actions**
  - Arc(i, l, d): make an arc (with direction d, label l) between the right-most node to node i. Arc(i,-,-) represents no edge between them.

<table>
<thead>
<tr>
<th>stack</th>
<th>cache</th>
<th>buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>($,1), ($,1)</td>
<td>$ PER want-01</td>
<td>go-01</td>
</tr>
</tbody>
</table>

Arc(1,-,-), Arc(2,L,ARG0)

<table>
<thead>
<tr>
<th>stack</th>
<th>cache</th>
<th>buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$ PER want-01</td>
<td>go-01</td>
</tr>
</tbody>
</table>
Example of cache transition

Action taken: Initialization

<table>
<thead>
<tr>
<th>stack</th>
<th>cache</th>
<th>buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>PER</td>
<td>want-01</td>
<td>go-01</td>
</tr>
</tbody>
</table>
Example of cache transition

Action taken: SHIFT, PUSH(1)

stack | cache | buffer

(1, $) | $ $ PER | want-01 go-01

Hypothesis: PER
Example of cache transition

Action taken: Arc(1, -, -), Arc(2, -, -)

<table>
<thead>
<tr>
<th>stack</th>
<th>cache</th>
<th>buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, $)</td>
<td>$</td>
<td>want-01</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>go-01</td>
</tr>
<tr>
<td></td>
<td>PER</td>
<td></td>
</tr>
</tbody>
</table>

Hypothesis: PER
Example of cache transition

<table>
<thead>
<tr>
<th>stack</th>
<th>cache</th>
<th>buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, $)</td>
<td>(1, $)</td>
<td>go-01</td>
</tr>
<tr>
<td>$ PER</td>
<td>want-01</td>
<td></td>
</tr>
</tbody>
</table>

Action taken: SHIFT, PUSH(1)

Hypothesis: PER want-01
Example of cache transition

Action taken: Arc(1, -, -), Arc(2, L, ARG0)

Hypothesis: PER want-01

stack | cache | buffer
--- | --- | ---
(1, $) (1, $) | $ PER want-01 | go-01
Example of cache transition

Action taken: SHIFT, PUSH(1)

Hypothesis:

stack
(1, $) (1, $) (1, $)

ARG0

Hypothesis: PER want-01 go-01

buffer
Example of cache transition

Action taken: Arc(1, L, ARG0), Arc(2, R, ARG1)

Hypothesis:
Example of cache transition

Action taken: POP POP POP POP

Hypothesis:

PER → ARG0
want-01 → ARG0
ARG1 → go-01
Sequence to sequence models for cache transition system

- Concepts are generated from input sentences by another classifier in the preprocessing step.

- Separate encoders are adopted for input sentences and sequences of concepts, respectively.

- One decoder for generating transition actions.
Seq2seq (soft-attention+features)
Seq2seq (hard-attention+features)
Experiments

- Dataset: LDC2015E86
  - 16,833(train)/1,368(dev)/1,371(test)
- Evaluation: Smatch (Cai et al., 2013)
We first evaluate our algorithm on Abstract Meaning Representation (AMR) (Banarescu et al. 2013). AMR is a semantic formalism where the meaning of a sentence is encoded as a rooted, directed graph. Figure 8 shows an example of an AMR graph in which the nodes represent the AMR concepts and the edges represent the relations between the concepts they connect. AMR concepts consist of predicate senses, named entity annotations, and in some cases, simply lemmas of English words. AMR relations consist of core semantic roles drawn from the Propbank (Palmer, Gildea, and Kingsbury 2005) as well as very fine-grained semantic relations defined specifically for AMR. We use the training set of LDC2015E86 for SemEval 2016 task 8 on meaning representation parsing (May 2016), which contains 16,833 sentences. This dataset covers various domains including newswire and web discussion forums.

For each graph, we derive a vertex order corresponding to the English word order by using the automatically generated alignments provided with the dataset, which align
## Development results

### Impact of various components

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soft</td>
<td>0.55</td>
<td>0.51</td>
<td>0.53</td>
</tr>
<tr>
<td>Soft+feats</td>
<td>0.69</td>
<td>0.63</td>
<td>0.66</td>
</tr>
<tr>
<td>Hard+feats</td>
<td>0.70</td>
<td>0.64</td>
<td>0.67</td>
</tr>
</tbody>
</table>

### Impact of cache size

<table>
<thead>
<tr>
<th>cache size</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.69</td>
<td>0.63</td>
<td>0.66</td>
</tr>
<tr>
<td>5</td>
<td>0.70</td>
<td>0.64</td>
<td>0.67</td>
</tr>
<tr>
<td>6</td>
<td>0.69</td>
<td>0.64</td>
<td>0.66</td>
</tr>
</tbody>
</table>
## Main results

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buys and Blunsom (2017)</td>
<td>--</td>
<td>--</td>
<td>0.60</td>
</tr>
<tr>
<td>Konstas et al. (2017)</td>
<td>0.60</td>
<td>0.65</td>
<td>0.62</td>
</tr>
<tr>
<td>Ballesteros and Al-Onaizan (2017)</td>
<td>--</td>
<td>--</td>
<td>0.64</td>
</tr>
<tr>
<td>Damonte et al. (2016)</td>
<td>--</td>
<td>--</td>
<td>0.64</td>
</tr>
<tr>
<td>Wang et al. (2015a)</td>
<td>0.70</td>
<td>0.63</td>
<td>0.66</td>
</tr>
<tr>
<td>Flanigan et al. (2016)</td>
<td>0.70</td>
<td>0.65</td>
<td>0.67</td>
</tr>
<tr>
<td>Wang and Xue (2017)</td>
<td>0.72</td>
<td>0.65</td>
<td>0.68</td>
</tr>
<tr>
<td>Lyu and Titov (2018)</td>
<td>--</td>
<td>--</td>
<td>0.74</td>
</tr>
<tr>
<td>Soft+feats</td>
<td>0.68</td>
<td>0.63</td>
<td>0.65</td>
</tr>
<tr>
<td>Hard+feats</td>
<td>0.69</td>
<td>0.64</td>
<td>0.66</td>
</tr>
</tbody>
</table>
Accuracy on reentrancies

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peng et al., (2018)</td>
<td>0.44</td>
<td>0.28</td>
<td>0.34</td>
</tr>
<tr>
<td>Damonte et al., (2017)</td>
<td>--</td>
<td>--</td>
<td>0.41</td>
</tr>
<tr>
<td>JAMR</td>
<td>0.47</td>
<td>0.38</td>
<td>0.42</td>
</tr>
<tr>
<td>Hard+feats (ours)</td>
<td><strong>0.58</strong></td>
<td>0.34</td>
<td><strong>0.43</strong></td>
</tr>
</tbody>
</table>
Reentrancy example

Sentence: I have no desire to live in any city.

JAMR output:

Peng et al. (2018) output:

Our hard attention output:
Conclusion

- Cache transition system based on a mathematical sound formalism for parsing to graphs.

- The cache transition process can be well-modeled by sequence-to-sequence models.
  - Features from transition states.
  - Monotonic hard attention.
Thank you for listening!

Questions