Transition-based Semantic Role Labeling
Using Predicate Argument Clustering

Workshop on Relational Models of Semantics

Jinho D. Choi & Martha Palmer
University of Colorado at Boulder
June 23rd, 2011
Dependency-based SRL

- Semantic role labeling
  - Task of identifying arguments of each predicate and labeling them with semantic roles in relation to the predicate.

- Dependency-based semantic role labeling
  - Advantages over constituent-based semantic role labeling.
    - Dependency parsing is faster (2.29 milliseconds / sentence).
    - Dependency structure is more similar to predicate argument structure.
  - Labels headwords instead of phrases.
    - Still can recover the original semantic chunks for the most of time (Choi and Palmer, LAW 2010).
Dependency-based SRL

- Constituent-based vs. dependency-based SRL

He opened the door with his foot at ten.

Agent: He
Theme: the door
Instrument: with
Temporal: at ten
Dependency-based SRL

• Constituent-based vs. dependency-based SRL

He
opened

SBJ OBJ ADV TMP
He door with foot at ten
the with his his at

ARG\(_0\) ARG\(_1\) ARG\(_2\) TMP
He the door with his foot at ten
Motivations

• Do argument **identification** and **classification** need to be in separate steps?
  - They may require two different feature sets.
  - Training them in a pipeline takes less time than as a joint-inference task.
  - We have seen advantages of dealing with them as a joint-inference task in dependency parsing, why not in SRL?
Transition-based SRL

• Dependency parsing vs. dependency-based SRL
  – Both try to find relations between word pairs.
  – Dep-based SRL is a special kind of dep. parsing.
    • It restricts the search only to top-down relations between predicate (head) and argument (dependent) pairs.
    • It allows multiple predicates for each argument.

• Transition-based SRL algorithm
  – Top-down, bidirectional search. → More suitable for SRL
  – Easier to develop a joint-inference system between dependency parsing and semantic role labeling.
Transition-based SRL

• Parsing states
  
  - $(\lambda_1, \lambda_2, p, \lambda_3, \lambda_4, A)$
  - $p$ - index of the current predicate candidate.
  - $\lambda_1$ - indices of lefthand-side argument candidates.
  - $\lambda_4$ - indices of righthand-side argument candidates.
  - $\lambda_2, \lambda_3$ - indices of processed tokens.
  - $A$ - labeled arcs with semantic roles

• Initialization: $([\ ]$, $[\ ]$, $1$, $[\ ]$, $[2, ..., n]$, $\emptyset$)

• Termination: $(\lambda_1, \lambda_2, \emptyset$, $[\ ]$, $[\ ]$, $A)$
Transition-based SRL

- **Transitions**
  - **No-Pred** - finds the next predicate candidate.
  - **No-Arc** - rejects the lefthand-side argument candidate.
  - **No-Arc** - rejects the righthand-side argument candidate.
  - **Left-Arc** - accepts the lefthand-side argument candidate.
  - **Right-Arc** - accepts the righthand-side argument candidate.
John_1 \text{ wants}_2 \text{ to}_3 \text{ buy}_4 \text{ } a_5 \text{ car}_6

- No-Pred
- Left-Arc : John \leftarrow \text{ wants}
- Right-Arc : \text{ wants} \rightarrow \text{ to}
- No-Arc \times 3
- Shift

\begin{align*}
\lambda_1 & : \text{ to} \\
\lambda_2 & : \text{ wants} \\
\lambda_3 & : \text{ wants} \\
\lambda_4 & : \text{ wants} \\
A & : \text{ buy} \rightarrow \text{ car}
\end{align*}
Features

- Baseline features
  - N-gram and binary features (similar to ones in Johansson and Nugues, EMNLP 2008).
  - Structural features.

Subcategorization of “wants”

Path from “John” to “buy”

Depth from “John” to “buy”

1 ↑ LCA ↓ 2
Features

- Dynamic features
  - Derived from previously identified arguments.
  - Previously identified argument label of $w_{arg}$.
  - Label of the very last predicted numbered argument of $w_{pred}$.
  - These features can narrow down the scope of expected arguments of $w_{pred}$.
Experiments

• Corpora
  - CoNLL’09 English data.
  - In-domain task: the Wall Street Journal.
  - Out-of-domain task: the Brown corpus.

• Input to our semantic role labeler
  - Automatically generated dependency trees.
  - Used our open-source dependency parser, ClearParser.

• Machine learning algorithm
  - Liblinear L2-L1 SVM.
Experiments

- **Results**
  - **AI** - Argument Identification.
  - **AC** - Argument Classification.

<table>
<thead>
<tr>
<th>Task</th>
<th>In-domain</th>
<th>Out-of-domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>Baseline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AI</td>
<td>92.57</td>
<td>88.44</td>
</tr>
<tr>
<td>AI+AC</td>
<td>87.20</td>
<td>83.31</td>
</tr>
<tr>
<td>+ Dynamic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AI</td>
<td>92.38</td>
<td>88.76</td>
</tr>
<tr>
<td>AI+AC</td>
<td>87.33</td>
<td>83.91</td>
</tr>
<tr>
<td>JN’08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AI+AC</td>
<td>88.46</td>
<td>83.55</td>
</tr>
</tbody>
</table>
Summary

- Introduced a **transition-based SRL algorithm**, showing near state-of-the-art results.
  - No need to design separate systems for argument identification and classification.
  - Make it easier to develop a joint-inference system between dependency parsing and semantic role labeling.

- **Future work**
  - Several techniques, designed to improve transition-based parsing, can be applied (e.g., dynamic programming, k-best ranking)
  - We can apply more features, such as **clustering information**, to improve labeling accuracy.
Predicate Argument Clustering

- Verb clusters can give more generalization to the statistical models.
  - Clustering verbs using bag-of-words, syntactic structure.
  - Clustering verbs using predicate argument structure.

- Self-learning clustering
  - Cluster verbs in the **test data** using automatically generated predicate argument structures.
  - Cluster verbs in the **training data** using the verb clusters found in the test data as seeds.
  - Re-run our semantic role labeler on the test data using the clustering information.
Predicate Argument Clustering

• Vector representation

  - Semantic role labels, semantic role labels + word lemmas.

| Verb | A0 | A1 | ... | john:A0 | to:A1 | car:A1 | ...
|------|----|----|-----|---------|-------|--------|-----
| want | 1  | 1  | 0s  | 1       | 1     | 0      | 0s  |
| buy  | 1  | 1  | 0s  | 1       | 0     | 1      | 0s  |

\[
s(l_j|v_i) = \frac{1}{1 + \exp(-\text{score}(l_j|v_i))}
\]

score of \(l_j\) being a label of \(v_i\)

\[
s(m_j, l_j) = \begin{cases} 
1 & (w_j \neq \text{noun}) \\
\exp\left(\frac{\text{count}(m_j, l_j)}{\sum_{k \neq j} \text{count}(m_k, l_k)}\right) & \text{max. likelihood of } m_j \text{ co-occurring with } l_j
\end{cases}
\]
Predicate Argument Clustering

• Clustering verbs in the test data
    • Merges *k-best* pairs at each iteration.
    • Uses a threshold to *dynamically* determine the top *k* clusters.
    - We set another threshold for early break-out.

• Clustering verbs in the training data
  - *K-means* clustering.
    • Starts with *centroids* estimated from the clusters found in the test data.
    • Uses a threshold to filter out verbs not close enough to any cluster.
Experiments

- Results

<table>
<thead>
<tr>
<th>Task</th>
<th>In-domain</th>
<th>Out-of-domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td><strong>Baseline</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AI</td>
<td>92.57</td>
<td>88.44</td>
</tr>
<tr>
<td>AI+AC</td>
<td>87.20</td>
<td>83.31</td>
</tr>
<tr>
<td><strong>+ Dynamic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AI</td>
<td>92.38</td>
<td>88.76</td>
</tr>
<tr>
<td>AI+AC</td>
<td>87.33</td>
<td>83.91</td>
</tr>
<tr>
<td><strong>+ Cluster</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AI</td>
<td>92.62</td>
<td>88.90</td>
</tr>
<tr>
<td>AI+AC</td>
<td>87.43</td>
<td>83.92</td>
</tr>
<tr>
<td><strong>JN’08</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AI+AC</td>
<td>88.46</td>
<td>83.55</td>
</tr>
</tbody>
</table>
Conclusion

• Introduced **self-learning clustering technique**, potential for improving labeling accuracy in the new domain.
  - Need to try with large scale data to see a clear impact of the clustering.
  - Can also be improved by using different features or clustering algorithms.

• ClearParser open-source project
Acknowledgements

• We gratefully acknowledge the support of the National Science Foundation Grants CISE-IIIS- RI-0910992, Richer Representations for Machine Translation, a subcontract from the Mayo Clinic and Harvard Children’s Hospital based on a grant from the ONC, 90TR0002/01, Strategic Health Advanced Research Project Area 4: Natural Language Processing, and a grant from the Defense Advanced Research Projects Agency (DARPA/IPTO) under the GALE program, DARPA/CMO Contract No. HR0011-06-C-0022, subcontract from BBN, Inc. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.