Multi-Task Structured Prediction for Entity Analysis: Search Based Learning Algorithms

Chao Ma, †Janardhan Rao Doppa, Prasad Tadepalli, Hamed Shahbazi, Xiaoli Fern

Oregon State University
†Washington State University

Nov. 17th, 2017
Entity Analysis in Language Processing

Many NLP tasks process mentions of entities – things, people, organizations, etc.

- Named Entity Recognition
- Coreference Resolution
- Entity Linking
  - Semantic Role Labeling
  - Entity Relation Extraction

We focus on three of them in this work
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago…
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago...

Coreference:

\[ y_{i} = \{1, 2 \ldots i\} \]
Coreference Resolution

He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago...

Coreference:

\[ y_i = \{1, 2 \ldots i\} \]

Left-linking Tree formulation for coreference resolution:

\[ y_{\text{coref}} = (\ ? \ , \ ? \ , \ ? \ , \ ? \ , \ ? \ , \ ? \ , \ ? \ , \ ? \ ) \]
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago...

Coreference:

\[ y_i = \{1, 2 \ldots i\} \]

Left-linking Tree formulation for coreference resolution:

\[ y_{\text{coref}} = (1, ?, ?, ?, ?, ?, ?, ?) \]
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago...

Coreference: 
\[ y_i = \{1, 2 \ldots i\} \]

Left-linking Tree formulation for coreference resolution:

\[ y_{\text{coref}} = (1, 1, \text{?}, \text{?}, \text{?}, \text{?}, \text{?}) \]
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago...

**Coreference:**

\[ y_i = \{1, 2 \ldots i\} \]

**Left-linking Tree formulation for coreference resolution:**

\[ y_{coref} = (1, 1, 2, ?, ?, ?, ?) \]
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago…

Coreference:
\[ y_i = \{1, 2 \ldots i\} \]

Left-linking Tree formulation for coreference resolution:

\[ y_{coref} = (1, 1, 2, 4, ?, ?, ?, ?) \]
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago...

Coreference:
\[ y_i = \{1, 2 \ldots i\} \]

Left-linking Tree formulation for coreference resolution:

\[ y_{coref} = (1, 1, 2, 4, 5, ?, ?, ?) \]
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago…

Coreference:
\[ y_i = \{1, 2 \ldots i\} \]

Left-linking Tree formulation for coreference resolution:

\[ y_{coref} = (1, 1, 2, 4, 5, 6, ?) \]
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago…

Coreference:

\[ y_i = \{1, 2 \ldots i\} \]

Left-linking Tree formulation for coreference resolution:

\[ y_{\text{coref}} = (1, 1, 2, 4, 5, 6, 7) \]
Coreference Resolution

He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago...

Coreference: $y_i = \{1, 2 \ldots i\}$

Left-linking Tree formulation for coreference resolution:

$$y_{\text{coref}} = (1,1,2,4,5,6,7)$$

ACML 2017
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago…

Coreference: $y_{\text{coref}} = \{1, 2 \ldots i\}$
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago...

Coreference: \( y_{\text{coref}} = \)
\[ y_i = \{1, 2 \ldots i\} \]

Named Entity Recognition: \( y_{\text{ner}} = \)
\[ y_i = \{\text{ORG, PER, GPE, LOC, FAC, VEL, WEA}\} \]
He left [Columbia] in 1983 with a BA degree, ... after graduating from [Columbia University], he worked as a community organizer in Chicago...

**Coreference:**

\[ y_{\text{coref}} = \begin{cases} 1, 2 \ldots i \end{cases} \]

**Named Entity Recognition:**

\[ y_{\text{ner}} = \begin{cases} \text{ORG, PER, GPE, LOC, FAC, VEL, WEA} \end{cases} \]

**Entity Linking:**

\[ y_{\text{link}} = \begin{cases} \text{https://en.wikipedia.org/wiki/Columbia\_University,} \\
\text{https://en.wikipedia.org/wiki/Columbia\_District,} \\
\text{https://en.wikipedia.org/wiki/Columbia\_British\_Columbia,} \\
\text{https://en.wikipedia.org/wiki/Columbia\_College\_Columbia\_University,} \\
\text{...} \end{cases} \]
Single Task Structured Prediction

Typical (Single-Task) Structured Prediction:

\[ f(x, y) = w \cdot \phi(x, y) \]

\[ \hat{y} = \text{argmax}_y f(x, y) \]

Intractable in most cases

Candidate Methods:
- Graphical models
- Structured Perceptron
- Structured SVM
- Belief Propagation
- Integer Linear Programming (ILP)
- Beam Search

This Work
Structured SVM Learning with Search-based Inference

$y^*$

$x$

$w$

Beam Search Inference

Loss-Augmented Inference

Weight Learner

Updated weights

$y'$

New $y'$ for $x$ and updated constraints after adding $y'$

Dual Coordinate Descent (DCD) Learning Algorithm

ACMIL 2017
Multi-Task Structured Prediction

Multi-Task Structured Prediction (MTSP):

Input $x$

$\text{Intra-task Features}$

$\begin{align*}
    f_1 : X &\rightarrow Y_1 \\
    &= w_1 \cdot \phi_1(x, y)
\end{align*}$

$\begin{align*}
    f_2 : X &\rightarrow Y_2 \\
    &= w_2 \cdot \phi_2(x, y')
\end{align*}$

$\begin{align*}
    f_3 : X &\rightarrow Y_3 \\
    &= w_3 \cdot \phi_3(x, y'')
\end{align*}$

Output $y_1$, $y_2$, $y_3$

$\text{models}$

$\text{How to exploit the interdependencies between tasks?}$
Multi-Task Structured Prediction

Introduce Inter-task Features:

\[ f_1 : X \rightarrow Y_1 = w_1 \cdot \phi_1(x, y) \]
\[ f_2 : X \rightarrow Y_2 = w_2 \cdot \phi_2(x, y') \]
\[ f_3 : X \rightarrow Y_3 = w_3 \cdot \phi_3(x, y'') \]

\[ \phi_{(1,2)}(x, y, y') \]
\[ \phi_{(2,3)}(x, y', y'') \]
\[ \phi_{(1,3)}(x, y, y'') \]
## Pipeline Architecture

Learning $k (= 3)$ independent models, one after another;

<table>
<thead>
<tr>
<th>Before Start:</th>
<th>Models</th>
<th>Predict Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>$w_2$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>$w_{(1,2)}$</td>
<td>$w_3$</td>
<td>$y_2$</td>
</tr>
<tr>
<td>$w_{(1,3)}$</td>
<td>$w_{(2,3)}$</td>
<td>$y_3$</td>
</tr>
</tbody>
</table>

Define a order:  Task 1 $\rightarrow$ Task 2 $\rightarrow$ Task 3
Pipeline Architecture

Learning $k$ ($= 3$) independent models, one after another;

Before Start:

<table>
<thead>
<tr>
<th>Models</th>
<th>Predict Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>$y_1$</td>
</tr>
<tr>
<td>$w_2$</td>
<td>$y_2$</td>
</tr>
<tr>
<td>$w_{(1,2)}$</td>
<td>$y_3$</td>
</tr>
<tr>
<td>$w_3$</td>
<td></td>
</tr>
<tr>
<td>$w_{(1,3)}$</td>
<td></td>
</tr>
<tr>
<td>$w_{(2,3)}$</td>
<td></td>
</tr>
</tbody>
</table>

Task 1:

Use feature $\phi_1(x, y)$

SSVM Learner

$x$  

$w_1$

$y_1$
Pipeline Architecture

Learning $k$ (= 3) independent models, one after another;

Before Start:

Task 1:

- **Use feature** $\phi_1(x, y)$
- **Use feature** $\phi_2(x, y), \phi_{(1,2)}(x, y, y')$

Task 2:

- **Use feature** $\phi_1(x, y)$
- **Use feature** $\phi_{(1,2)}(x, y, y')$

Models

Predict Output
Pipeline Architecture

Learning $k$ (= 3) independent models, one after another;

Before Start:

Task 1:

- Use feature $\phi_1(x, y)$

Task 2:

- Use feature $\phi_2(x, y), \phi_{(1,2)}(x, y, y')$

Task 3:

- Use feature $\phi_3(x, y), \phi_{(1,3)}(x, y, y''), \phi_{(2,3)}(x, y', y'')$
Each group of bars represents one task. In each group, we show the accuracy when the task is placed at first (1st bar), or at last (2nd and 3rd bar).

- The task performs better when it is placed last in order.
- There is no ordering that allows the pipeline to reach peak performance on all the three tasks.
Joint Architecture

Task 1 & 2 & 3:

Use all features $\phi_1(x, y), \phi_2(x, y), \phi_3(x, y), \phi_{(1,2)}(x, y, y'), \phi_{(1,3)}(x, y, y''), \phi_{(2,3)}(x, y', y'')$

SSVM Learner

Big Problem: Huge branching factor for search
Pruning

A pruner is a classifier to prune the domain of each variable using state features.

Score-agnostic Pruning

- Can accelerate the training time;
- May or may not improve the testing accuracy;

Score-sensitive Pruning

- Can improve the testing accuracy;
- No training speedup, but evaluation does speed up.
Cyclic Architecture

Pipeline architecture

Task 1 $\rightarrow$ Task 2 $\rightarrow$ Task 3

Connect the tail of pipeline to the head?
Cyclic Architecture

Unshared-Weight-Cyclic Training

Step 1: Define a order: Task 1 → Task 2 → Task 3

Step 2: Predict initial outputs: \( y_1 \), \( y_2 \), \( y_3 \)
Cyclic Architecture

*Unshared-Weight-Cyclic Training*

Step 1: Define a order: Task 1 → Task 2 → Task 3

Step 2: Predict initial outputs:

Use features
\[ \phi_1(x,y), \]
\[ \phi_{(1,2)}(x,y,y'), \]
\[ \phi_{(1,3)}(x,y,y'') \]

Predict initial outputs:
\[ y_1 \quad y_2 \quad y_3 \]
Cyclic Architecture

Unshared-Weight-Cyclic Training

Step 1: Define a order: Task 1 → Task 2 → Task 3

Step 2: Predict initial outputs: \( y_1 \), \( y_2 \), \( y_3 \)

Use features:
- \( \phi_1 (x,y) \)
- \( \phi_{(1,2)} (x,y,y') \)
- \( \phi_{(1,3)} (x,y,y'') \)

Predict initial outputs:
- \( y_2 \)
- \( y_3 \)

Use features:
- \( \phi_2 (x,y) \)
- \( \phi_{(1,2)} (x,y,y') \)
- \( \phi_{(2,3)} (x,y',y'') \)

Unshared-Weight-Cyclic Training
Cyclic Architecture

Unshared-Weight-Cyclic Training

Step 1: Define a order: Task 1 $\rightarrow$ Task 2 $\rightarrow$ Task 3

Step 2: Predict initial outputs:

\[
\begin{align*}
\phi_1(x, y), \\
\phi_{(1,2)}(x, y, y'), \\
\phi_{(1,3)}(x, y, y'')
\end{align*}
\]

Use features $\phi_2(x, y), \phi_{(1,2)}(x, y, y'), \phi_{(2,3)}(x, y', y'')$

Weights are independent

\[
\begin{align*}
\phi_3(x, y), \\
\phi_{(1,3)}(x, y, y''), \\
\phi_{(2,3)}(x, y', y'')
\end{align*}
\]
Cyclic Architecture

Shared-Weight-Cyclic Training

Step 1: Define a order: Task 1 $\rightarrow$ Task 2 $\rightarrow$ Task 3

Step 2: Predict initial outputs: $y_1$ $y_2$ $y_3$
Cyclic Architecture

Shared-Weight-Cyclic Training

Step 1: Define a order: Task 1 → Task 2 → Task 3

Step 2: Predict initial outputs:

Use features
\[ \phi_1(x,y), \]
\[ \phi_{(1,2)}(x,y,y'), \]
\[ \phi_{(1,3)}(x,y,y'') \]

Task 1 Turn

SSVM Learner

Initial Outputs:
\[ y_1 \]
\[ y_2 \]
\[ y_3 \]
Cyclic Architecture

Shared-Weight-Cyclic Training

Step 1: Define a order: Task 1 → Task 2 → Task 3

Step 2: Predict initial outputs: $y_1$, $y_2$, $y_3$

Use features

$\phi_1(x,y)$,
$\phi_{(1,2)}(x,y,y')$,  
$\phi_{(1,3)}(x,y,y'')$

Task 1 Turn

SSVM Learner

Task 2 Turn

SSVM Learner

$w_1$, $w_2$, $w_3$, $w_{(1,2)}$, $w_{(1,3)}$, $w_{(2,3)}$
Cyclic Architecture

**Shared-Weight-Cyclic Training**

**Step 1:** Define an order: Task 1 $\rightarrow$ Task 2 $\rightarrow$ Task 3

**Step 2:** Predict initial outputs: $y_1$, $y_2$, $y_3$

Use features $\phi_1(x,y)$, $\phi_{(1,2)}(x,y,y')$, $\phi_{(1,3)}(x,y,y'')$

Use features $\phi_2(x,y)$, $\phi_{(1,2)}(x,y,y')$, $\phi_{(2,3)}(x,y',y'')$

Weights are shared

ACML 2017
Experimental Setup

Datasets:

ACE2005
AC-to-Wiki annotation

Train/Dev/Test
338/144/117

ACE-KBP2015

Train/Dev/Test
132/36/167

Knowledge Base:

Wikipedia
(2015 dump)

Freebase
(2014 dump)

Evaluation:

Coref. NER Linking

Within.Coref Cross.Coref NER & Linking

MUC BCube CEAFm

average

CoNLL

Hamming

CoNLL

Hamming

Hamming

Combined accuracy of NER and Linking

All metrics are accuracies (larger is better)
### Joint Architecture Performance

#### ACE05 Test Set Performance

<table>
<thead>
<tr>
<th>Algs.</th>
<th>Coreference</th>
<th>NER</th>
<th>Link</th>
<th>Train time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berkeley</td>
<td><strong>81.41</strong></td>
<td><strong>74.7</strong></td>
<td><strong>72.93</strong></td>
<td><strong>76.35</strong></td>
</tr>
<tr>
<td>a. Results of Joint Architecture without Pruning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STSP</td>
<td><strong>80.28</strong></td>
<td><strong>73.26</strong></td>
<td><strong>71.58</strong></td>
<td><strong>75.04</strong></td>
</tr>
<tr>
<td>Joint w. Rand Init</td>
<td><strong>80.23</strong></td>
<td><strong>73.79</strong></td>
<td><strong>72.03</strong></td>
<td><strong>75.35</strong></td>
</tr>
<tr>
<td>Joint w. Good init</td>
<td><strong>82.18</strong></td>
<td><strong>76.57</strong></td>
<td><strong>74.00</strong></td>
<td><strong>77.58</strong></td>
</tr>
</tbody>
</table>

#### TAC15 Test Set Performance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Berkeley</td>
<td><strong>88.9</strong></td>
<td><strong>74.8</strong></td>
<td><strong>72.8</strong></td>
<td><strong>82.98</strong></td>
<td><strong>80.8</strong></td>
<td><strong>6m29s</strong></td>
</tr>
<tr>
<td>a. Results of Joint Architecture without Pruning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STSP</td>
<td><strong>87.3</strong></td>
<td><strong>76.2</strong></td>
<td><strong>70.9</strong></td>
<td><strong>81.21</strong></td>
<td><strong>78.8</strong></td>
<td><strong>2m41s</strong></td>
</tr>
<tr>
<td>Joint w. Rand Ini</td>
<td><strong>87.1</strong></td>
<td><strong>71.17</strong></td>
<td><strong>68.33</strong></td>
<td><strong>81.31</strong></td>
<td><strong>78.4</strong></td>
<td><strong>7m19s</strong></td>
</tr>
<tr>
<td>Joint w. Good. Ini</td>
<td><strong>89.72</strong></td>
<td><strong>76.98</strong></td>
<td><strong>74.43</strong></td>
<td><strong>82.8</strong></td>
<td><strong>81.3</strong></td>
<td><strong>6m11s</strong></td>
</tr>
</tbody>
</table>

1. **Joint-Good-Init > STSP**
   
   Interdependency, captured by inter-task features, does benefit the system.

2. **Joint-Good-Init > Joint-Rand-Init**
   
   Search-based inference for large structured prediction problems suffers from local optima and is mitigated by a good initialization.

3. Search-based MTSP is competitive or better than the state-of-the-art system.
Results  

Joint Architecture Performance

<table>
<thead>
<tr>
<th>ACE05 Test Set Performance</th>
<th>TAC15 Test Set Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algs.</strong></td>
<td><strong>Coreference</strong></td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td>a. Results of Joint Architecture without Pruning</td>
<td>81.41</td>
</tr>
<tr>
<td>STSP</td>
<td>80.28</td>
</tr>
<tr>
<td>Joint w. Rand Init</td>
<td>80.23</td>
</tr>
<tr>
<td>Joint w. Good init</td>
<td>82.18</td>
</tr>
<tr>
<td>b. Results of Joint Architecture with Pruning</td>
<td></td>
</tr>
<tr>
<td>Score-agnostic</td>
<td>81.10</td>
</tr>
<tr>
<td>Score-sensitive</td>
<td>82.81</td>
</tr>
<tr>
<td>Rank-1st</td>
<td>87</td>
</tr>
<tr>
<td>Berkeley</td>
<td>88.9</td>
</tr>
<tr>
<td>Joint w. Rand. Ini</td>
<td>87.1</td>
</tr>
<tr>
<td>Joint w. Good. Ini</td>
<td>89.72</td>
</tr>
<tr>
<td>b. Results of Joint Architecture with Pruning</td>
<td></td>
</tr>
<tr>
<td>Score-agnostic</td>
<td>89.54</td>
</tr>
<tr>
<td>Score-sensitive</td>
<td>89.33</td>
</tr>
</tbody>
</table>

1. **Joint-Good-Init > STSP**
   Interdependency, captured by inter-task features, does benefit the system.
2. **Joint-Good-Init > Joint-Rand-Init**
   Search-based inference for large structured prediction problems suffers from local optima and is mitigated by a good initialization.
3. Search-based MTSP is competitive or better than the state-of-the-art system.
4. Score-sensitive pruning of joint MTSP performs the best and takes most time.
### ACE05 Test Set Performance

<table>
<thead>
<tr>
<th>Algs.</th>
<th>Coreference</th>
<th>NER</th>
<th>Link</th>
<th>Train time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berkeley</td>
<td>MUC 81.41</td>
<td>74.7</td>
<td>72.93</td>
<td>31min</td>
</tr>
<tr>
<td>a. Results of Joint Architecture without Pruning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STSP</td>
<td>80.28</td>
<td>73.26</td>
<td>71.58</td>
<td>9min</td>
</tr>
<tr>
<td>Joint w. Rand Init</td>
<td>80.23</td>
<td>73.79</td>
<td>72.03</td>
<td>48min</td>
</tr>
<tr>
<td>Joint w. Good init</td>
<td>82.18</td>
<td>76.57</td>
<td>74.00</td>
<td>34min</td>
</tr>
<tr>
<td>b. Results of Joint Architecture with Pruning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score-agnostic</td>
<td>81.10</td>
<td>75.79</td>
<td>74.33</td>
<td>16min</td>
</tr>
<tr>
<td>Score-sensitive</td>
<td>82.81</td>
<td>75.77</td>
<td>74.96</td>
<td>37min</td>
</tr>
<tr>
<td>c. Results of Cyclic Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unshared-Wt-Cyclic</td>
<td>81.83</td>
<td>76.05</td>
<td>73.99</td>
<td>11min</td>
</tr>
<tr>
<td>Shared-Wt-Cyclic</td>
<td>80.97</td>
<td>75.22</td>
<td>73.39</td>
<td>10min</td>
</tr>
</tbody>
</table>

### TAC15 Test Set Performance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank-1st</td>
<td>Accu. 87</td>
<td>Accu. 73.7</td>
<td>Accu. 80</td>
<td>CoNLL 80.8</td>
<td>CEAFm 6m29s</td>
<td></td>
</tr>
<tr>
<td>Berkeley</td>
<td>88.9</td>
<td>74.8</td>
<td>72.8</td>
<td>82.98</td>
<td>80.8</td>
<td>6m29s</td>
</tr>
<tr>
<td>a. Results of Joint Architecture without Pruning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STSP</td>
<td>87.3</td>
<td>76.2</td>
<td>70.9</td>
<td>81.21</td>
<td>78.8</td>
<td>2m41s</td>
</tr>
<tr>
<td>Joint w. Rand. Ini</td>
<td>87.1</td>
<td>71.17</td>
<td>68.33</td>
<td>81.31</td>
<td>78.4</td>
<td>7m19s</td>
</tr>
<tr>
<td>Joint w. Good. Ini</td>
<td>89.72</td>
<td>76.98</td>
<td>74.43</td>
<td>82.8</td>
<td>81.3</td>
<td>6m11s</td>
</tr>
<tr>
<td>b. Results of Joint Architecture with Pruning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score-agnostic</td>
<td>89.54</td>
<td>76.84</td>
<td>74.31</td>
<td>82.99</td>
<td>81.4</td>
<td>4m15s</td>
</tr>
<tr>
<td>Score-sensitive</td>
<td>89.33</td>
<td>77.68</td>
<td>74.63</td>
<td>83.17</td>
<td>81.3</td>
<td>9m2s</td>
</tr>
<tr>
<td>c. Results of Cyclic Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unshared-Wt-Cyclic</td>
<td>89.57</td>
<td>77.68</td>
<td>74.6</td>
<td>82.08</td>
<td>80.5</td>
<td>3m52s</td>
</tr>
<tr>
<td>Shared-Wt-Cyclic</td>
<td>87.95</td>
<td>73.65</td>
<td>71.32</td>
<td>80.54</td>
<td>77.9</td>
<td>2m56s</td>
</tr>
</tbody>
</table>

**ACML 2017**
- Competitive accuracy, and much faster training
- Unshared weights perform better than shared weights
Summary

1. Search-based multi-task structured prediction outperforms prior work based on graphical models on all 3 entity analysis tasks.

2. Studied three learning and inference architectures: pipeline, cyclic, and joint, with trade-offs between accuracy and speed.

3. The joint architecture with score-sensitive pruning performs the best.

4. The cyclic architecture with unshared weights is competitive in accuracy and faster to train.
Thank You!