Randomized Greedy Inference for Joint Segmentation, Tagging and Parsing

Yuan Zhang, Chengtao Li, Regina Barzilay, Kareem Darwish

MIT, QCRI
Error Propagation in Pipeline Models

Dependency Accuracy on Arabic (SPMRL 2013)

- Pipeline: 82.4%
- Gold POS: 90.5%

Difference: 8.1%
Our Approach: Joint Model with Randomized Greedy

Dependency Accuracy on Arabic (SPMRL 2013)

- Pipeline: 82.4%
- Joint: 87.2%
- Gold POS: 90.5%
Randomized Greedy in Dependency Parsing

• Key idea: greedy hill-climbing with random restarts
• Highly effective inference procedure
Randomized Greedy in Dependency Parsing

• Key idea: greedy hill-climbing with random restarts
• Highly effective inference procedure

![Bar chart comparing Turbo and Our Full results]

- Turbo: 88.73%
- Our Full: 89.24%
Randomized Greedy in Dependency Parsing

- Key idea: greedy hill-climbing with random restarts
- Highly effective inference procedure

Finding Global Optimum on English

<table>
<thead>
<tr>
<th></th>
<th>Len. ≤ 15</th>
<th>Len. &gt; 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbo</td>
<td>100%</td>
<td>99.3%</td>
</tr>
<tr>
<td>Our Full</td>
<td>88.73%</td>
<td>89.24%</td>
</tr>
</tbody>
</table>
Randomized Greedy in Dependency Parsing

- Key idea: greedy hill-climbing with random restarts
- Highly effective inference procedure

Finding Global Optimum on English

<table>
<thead>
<tr>
<th></th>
<th>Len. ≤ 15</th>
<th>Len. &gt; 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbo</td>
<td>88.73%</td>
<td>100%</td>
</tr>
<tr>
<td>Our Full</td>
<td>89.24%</td>
<td>99.3%</td>
</tr>
</tbody>
</table>

- Analysis: parsing is easy on average

# Optima on English Dataset

% sentences: 50% 70% 90%

21 121 2000
Randomized Greedy in Dependency Parsing

- Key idea: greedy hill-climbing with random restarts
- Highly effective inference procedure

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Finding Global Optimum on English</th>
<th># Optima on English Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbo</td>
<td>Len. ≤ 15: 100%  Len. &gt; 15: 99.3%</td>
<td>2000</td>
</tr>
<tr>
<td>Our Full</td>
<td>88.73% 89.24%</td>
<td>50% 70% 90%</td>
</tr>
</tbody>
</table>

- Analysis: parsing is easy on average

Scalable for more complex joint inference?
Randomized Greedy for Joint Prediction

Dependency Accuracy on Arabic (SPMRL 2013)

- Pipeline: 82.4%
- Joint: 87.2%
- Gold POS: 90.5%

- Advantages:
  - No constraints on the scoring function
  - Easy language adaptation
  - Easy parallelization
Core Idea

- **Climb** to the optimal assignment for \((s, t, y)\) in a few small greedy steps

**Randomized Hill-climbing**

For \(k = 1\) to \(K\)

1) Sample segmentation \(s\), POS tags \(t\) and a dependency tree \(y\)
2) Greedily improve the POS tags and the tree
3) Repeat (2) until converge

Select the assignment with the highest score
Sample Segmentation and POS Tag

- Sample from first-order distribution
  \[ p(s) \propto \exp\{\theta \cdot f(s)\}, \quad p(t) \propto \exp\{\theta \cdot f(s,t)\} \]
Sample using a random walk-based algorithm (Wilson, 1996)
Update each POS to maximize the full scoring function

\[ t_{i,j} \leftarrow \arg\max_{t_{i,j}} \{ \theta \cdot f(s, t_{i,j}, t_{(i,j)}, y) \} \]
• Update each POS to maximize the full scoring function

\[ t_{i,j} \leftarrow \arg\max_{t_{i,j}} \{ \theta \cdot f(s, t_{i,j}, t_{-(i,j)}, y) \} \]
• Update each dependency to maximize the full scoring function

\[ y_{i,j} \leftarrow \operatorname{argmax}_{y_{i,j}} \{ \theta \cdot f(s, t, y_{i,j}, y_{-(i,j)}) \} \]
Hill-climbing with Restarts

- Overcome local optima via **restarts**
- Parallelize each run during hill-climbing
Learning Algorithm

• Follow common max-margin framework

\[ \theta \cdot f(x, \hat{s}, \hat{t}, \hat{y}) \geq \theta \cdot f(x, s, t, y) + \text{Err}(s, t, y) - \xi \]

- \( \hat{s}, \hat{t}, \hat{y} \) are gold values of segmentation, POS tags and dependencies

• Adopt **passive-aggressive** online learning framework (Crammer et al. 2006)

• Decode with our randomized greedy algorithm
Generating Lattice Structure: Arabic

- Use MADA to generate top-$k$ morphological analyses
- Convert analyses to equivalent lattice

Word *Emlyp*:

- *Emly/NOUN* + *p/NSUFF*
- *Emly/ADJ* + *p/NSUFF*
- *Eml/NOUN* + *y/NSUFF* + *p/PRON*
Generating Lattice Structure: Chinese

- Use Stanford word segmenter to generate top-$k$ segmentation
- Convert segmentation to equivalent lattice
Experimental Setup

• Datasets
  – Chinese Penn Treebank 5.0 (CTB5)
  – Modern Standard Arabic (MSA): the SPMRL 2013 dataset
  – Mixed Arabic dataset
    • Training: MSA
    • Testing: Classical Arabic
    • Different vocabulary but similar grammar

• Evaluation Metric
  – F-score for segmentation, POS tagging and dependency parsing
  – TedEval (Tsarfaty et al. 2012) for the SPMRL dataset
    • A joint evaluation of segmentation and parsing quality
Baselines

• State-of-the-art
  – The SPMRL 2013 dataset: pipeline system (Björkelund et al. 2013)
  – CTB5: transition-based model (Zhang et al. 2014)

• Pipeline variants of our model
  – Predicted POS tags and segmentations by the same systems that we use to generate candidates
Features

• Segmentation
  – Morphemes/words scores, character-based features

• POS tagging
  – Up to 5-gram features, character-based features

• Dependency parsing
  – Up to 3rd-order (three arcs) features used in standard parsing

➢ Note: scoring function combines all features and capture cross-task interaction
Comparison to State-of-the-art Systems

<table>
<thead>
<tr>
<th></th>
<th>SPMRL</th>
<th>CTB5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Published F-Score</td>
<td>82.4%</td>
<td>81.7%</td>
</tr>
<tr>
<td>Ours</td>
<td>87.2%</td>
<td>82.0%</td>
</tr>
</tbody>
</table>

- SPMRL: pipeline model (Björkelund et al. 2013)
- CTB5: transition-based model (Zhang et al. 2014)
Comparison to State-of-the-art Models

TedEval Score on the SPMRL Dataset

- 91.7% for Björkelund et al. 2013
- 93.9% for Ours

- 27% error reduction on the TedEval score
Joint vs. Pipeline Model

POS Tagging F-Score

- **SPMRL**
  - Pipeline: 95.8%
  - Joint: 97.4%
  - Error reduction: 38%

- **CTB5**
  - Pipeline: 93.4%
  - Joint: 94.5%

- **Mixed**
  - Pipeline: 82.4%
  - Joint: 84.4%

- 38% error reduction on the SPMRL dataset
Impact on Seen and OOV Words

POS F-score Absolute Improvement (Joint vs. Pipeline)

- **SPMRL**
  - Seen: 1.6%
  - OOV: 2.8%

- **CTB5**
  - Seen: 0.8%
  - OOV: 7.7%

- **Mixed**
  - Seen: 0.4%
  - OOV: 10.3%
Randomized Greedy in Dependency Parsing

- Key idea: greedy hill-climbing with random restarts
- Highly effective inference procedure

Analysis: parsing is easy on average

Scalable for more complex joint inference?
Convergence Properties: Dependency Parsing

![Graph showing score convergence over restarts]

- Score convergence over the number of restarts.
- The graph indicates that the score stabilizes around 1.0 as the number of restarts increases.
- The convergence is rapid, with significant improvements observed within the first 50 restarts.
Convergence Properties: Dependency Parsing

![Graph showing the convergence of dependency parsing scores with increasing number of restarts. The score quickly stabilizes around 1 after a few restarts.](image-url)
Joint Model vs. Dependency Parsing

- Both tasks exhibit similar convergence
Joint Model vs. Dependency Parsing

- Both tasks exhibit similar convergence
Conclusion

• Randomized greedy algorithm scales up for joint prediction tasks

• Our model outperforms the state-of-the-art systems and its pipeline variant on both Arabic and Chinese

Source code available at:
https://github.com/yuanzh/SegParser
Thank You!