Better! 
Faster! 
Stronger!*!

Learning to balance accuracy and efficiency when predicting linguistic structures

(*theorems)

Hal Daumé III  |  UMD CS, UMIACS, Linguistics  |  me@hal3.name  |  @haldaume3
## NLP as transduction

<table>
<thead>
<tr>
<th>Task</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Translation</td>
<td>Ces deux principes se tiennent à la croisée de la philosophie, de la politique, de l’économie, de la sociologie et du droit.</td>
<td>Both principles lie at the crossroads of philosophy, politics, economics, sociology, and law.</td>
</tr>
<tr>
<td>Document Summarization</td>
<td>Argentina was still obsessed with the Falkland Islands even in 1994, 12 years after its defeat in the 74-day war with Britain. The country's overriding foreign policy aim continued to be winning sovereignty over the islands.</td>
<td>The Falkland islands war, in 1982, was fought between Britain and Argentina.</td>
</tr>
<tr>
<td>Syntactic Analysis</td>
<td>The man ate a big sandwich.</td>
<td>The man ate a big sandwich.</td>
</tr>
</tbody>
</table>

...many more...
Why are complex predictions slow?

- Parsing
  \[ \text{# trees} \sim O(2^{\text{|sentence|}}) \]
- Translation
  \[ \text{# trans} \sim O(2^{\text{|foreign| \times |english|}}) \]
- Summarization
  \[ \text{# sums} \sim O(2^{\text{|document|}}) \]

What about dynamic programming....?
- Often not possible (features are too complicated)
- Even when possible, polynomial-time is too painful:
  Parsing is \( O(|\text{grammar| \times |sentence|^3}) \)
  but \( |\text{grammar|} \) is

Concretely, \( n^3 = 15000 \)
\( |\text{grammar|} = \frac{1}{2} \) million
Case study: prioritized parsing

Time (seconds)

Sentence Length

0 1 minute

10 minutes

30 minutes

0 4 8 12 16 20

Sentence Length

0 500 1000 1500 2000 2500
Case study: prioritized parsing

![Graph showing time vs sentence length](image)

- Time (seconds) on the y-axis, ranging from 0 to 2500.
- Sentence Length on the x-axis, ranging from 0 to 20.
- Graph shows different time segments: 1 minute, 10 minutes, and 30 minutes.
- Input and Output points are indicated.
- The graph illustrates how time increases as sentence length increases.
Learning to be *fast!*

Quality = tradeoff(accuracy, time)
Prioritized search (e.g., parsing)

I can can cans

0 1 2 3 4

NP Aux Vrb NP

0 I can can
1 can
2 VP[1,3]
3 S[0,2]
4 VP[2,4]
Prioritized search (eg., parsing)
Prioritized search (eg., parsing)

Diagram showing a tree structure with nodes labeled as NP, VP, Pro, Aux, Vrb, and Noun. The tree branches are labeled with elements such as "I", "can", "cans", "S[0,2]", "VP[1,3]", and "VP[1,4]".
Prioritized search (eg., parsing)
Prioritized search (eg., parsing)

\[ S \]

\[ S[0,4] \]

\[ S[1,3] \]

\[ S[0,2] \]

I can can cans

NP      VP        NP

Pro  Aux   Vrb   Noun

0                   1                    2                   3                   4

VP

NP      VP        NP

Aux      Vrb

can  can  cans

(Jiang+Teich+er+D., NIPS 2012)
Prioritized search (eg., parsing)

```
9 stop
1 S[1,3]
1 S[0,2]
```
What do better priorities buy us?

- Ideally, run until queue is empty
- Ordering only matters because you want to:
  - Only pop items that have a real impact on result
  - Avoid premature pops
  - Stop early
- Goal: learn a priority function that optimizes accuracy/speed trade-off
- Typical solution: hand-built heuristics
- Our solution: learning to optimize $\text{accuracy} - \lambda \times \text{time}$
Prioritized search (eg., parsing)

Learning a heuristic:
\[ \text{priority}(x) = \theta \odot \phi(\text{chart, item}) \]

Example features:
- Inside score of VP
- Are 2 word VPs good?
- \( p(\text{VP} \mid I) \) and \( p(\text{cans} \mid \text{VP}) \)
- Could VP combine with NP?
- VP compete with other spans?
- Crossing constituents?

\[
\begin{array}{c}
2 & \text{VP}[1,3] \\
2 & \text{VP}[1,4] \\
1 & \text{S}[0,2] \\
\end{array}
\]
Learning priorities

Challenges:
The world is non-deterministic
State space is huge
   (25 words $\rightarrow$ $5 \times 10^{14}$ states)
The oracle is way too good
   (25 actions vs 25k actions)
The oracle does not experience
   a trade-off between speed and accuracy

2 VP[1,3]
2 VP[1,4]
1 S[0,2]

Pssst! You should choose the 2nd one!
Parsing Trajectories (I)

Policy Gradient Trajectories

Ground Truth Trajectories (Oracle Trajectories)
## Preliminary Result

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Relative # of pops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apprenticeship Learning</td>
<td>84.2</td>
<td>0.85x</td>
</tr>
<tr>
<td>with Reward Shaping</td>
<td>76.5</td>
<td>0.13x</td>
</tr>
<tr>
<td>Policy Gradient with Boltzmann Exploration</td>
<td>56.4</td>
<td>0.46x</td>
</tr>
<tr>
<td>Uniform cost search</td>
<td>93.3</td>
<td>1.0x</td>
</tr>
<tr>
<td>Pruned Uniform cost search</td>
<td>92.0</td>
<td>0.33x</td>
</tr>
</tbody>
</table>

**Failure Causes:**

Too hard to imitate the oracle with our features!
Parsing Trajectories (II)

Policy Gradient Trajectories

Oracle Trajectories

Oracle-Infused Trajectories

Good Trajectories
Oracle-Infused Policy Gradient

• Goal: “interleaving” oracle actions with policy actions both feasible and sensible

• Let $\delta \in [0,1]$. The oracle-infused policy

$$\pi^+_{\delta}(a|s) = \delta \pi^*(a|s) + (1 - \delta)\pi_t(a|s)$$

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Relative # of pops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle-Infused Policy Gradient</td>
<td>91.2</td>
<td>0.46x</td>
</tr>
<tr>
<td>Apprenticeship Learning</td>
<td>84.2</td>
<td>0.85x</td>
</tr>
<tr>
<td>with Reward Shaping</td>
<td>76.5</td>
<td>0.13x</td>
</tr>
<tr>
<td>Policy Gradient with Boltzmann Exploration</td>
<td>56.4</td>
<td>0.46x</td>
</tr>
<tr>
<td>Uniform cost search</td>
<td>93.3</td>
<td>1.0x</td>
</tr>
<tr>
<td>Pruned Uniform cost search</td>
<td>92.0</td>
<td>0.33x</td>
</tr>
</tbody>
</table>
Pareto Frontier

Change of recall and # of pops

- I+
- UC
- UCₚ
- CTF
- HA*

Has access to multiple levels of grammar

Multiple pruning thresholds

Pareto frontiers: Our I+ parser at different values of λ, against the baselines at different pruning levels. Lower and further right is better.
Many AI tasks can be cast as prioritized search (parsing, planning, inference...)

Non-determinism is a unique property of such sequential decision making processes.

Allows us to reason about trajectories, and learn to trade speed for accuracy.
Dependency parsing

NLP algorithms use a kitchen sink of features
Dependency parsing

NLP algorithms use a kitchen sink of features
Dependency parsing

NLP algorithms use a kitchen sink of features.
Dependency parsing

Edge Features:
- Lex(use→sink)
- POS(verb→noun)
- skip(2)
- skip(Det)
- skip(Noun)
- skip(Det Noun)
- dist=3
- various regexps...
- spelling features
- etc...

Three steps:
1. Compute POS tags
2. Compute $kn^2$ features
3. Run directed MST
Case study: dependency parsing

![Chart showing time vs. sentence length]
Dynamic feature selection

$This\ time$, the firms were ready.
The oracle too good!

- The oracle knows the label
- Picks feature with highest $y^*$ value
- Ends after selecting one feature
- Coach says how to improve, not the best thing to do

Psst! You should choose $\text{argmin} \ E[l(a)]$ a

If $N = T \log T$, $L(\pi_n) < T \epsilon_N$ for some

Provably smaller than DAgger's epsilon!
## Results across languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Method</th>
<th>First-order</th>
<th></th>
<th></th>
<th>Second-order</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Speed</td>
<td>Cost</td>
<td>UAS(D)</td>
<td>UAS(F)</td>
<td>Speed</td>
<td>Cost</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>DYNFS</td>
<td>3.11</td>
<td>45.4</td>
<td>91.2</td>
<td>91.3</td>
<td>5.20</td>
<td>44.3</td>
</tr>
<tr>
<td></td>
<td>VINEP</td>
<td>3.25</td>
<td>-</td>
<td>90.5</td>
<td>90.7</td>
<td>7.91</td>
<td>-</td>
</tr>
<tr>
<td>Chinese</td>
<td>DYNFS</td>
<td>2.09</td>
<td>43.5</td>
<td>91.0</td>
<td>91.3</td>
<td>2.72</td>
<td>49.4</td>
</tr>
<tr>
<td></td>
<td>VINEP</td>
<td>1.02</td>
<td>-</td>
<td>89.3</td>
<td>89.5</td>
<td>2.03</td>
<td>-</td>
</tr>
<tr>
<td>English</td>
<td>DYNFS</td>
<td>5.58</td>
<td>24.8</td>
<td>91.7</td>
<td>91.9</td>
<td>5.27</td>
<td>49.1</td>
</tr>
<tr>
<td></td>
<td>VINEP</td>
<td>5.23</td>
<td>-</td>
<td>91.0</td>
<td>91.2</td>
<td>11.88</td>
<td>-</td>
</tr>
<tr>
<td>German</td>
<td>DYNFS</td>
<td>4.71</td>
<td>21.0</td>
<td>89.2</td>
<td>89.3</td>
<td>6.02</td>
<td>36.6</td>
</tr>
<tr>
<td></td>
<td>VINEP</td>
<td>3.37</td>
<td>-</td>
<td>89.0</td>
<td>89.2</td>
<td>7.38</td>
<td>-</td>
</tr>
<tr>
<td>Japanese</td>
<td>DYNFS</td>
<td>5.04</td>
<td>16.2</td>
<td>93.7</td>
<td>93.6</td>
<td>8.49</td>
<td>19.9</td>
</tr>
<tr>
<td></td>
<td>VINEP</td>
<td>4.60</td>
<td>-</td>
<td>91.7</td>
<td>92.0</td>
<td>14.90</td>
<td>-</td>
</tr>
<tr>
<td>Portuguese</td>
<td>DYNFS</td>
<td>4.36</td>
<td>32.9</td>
<td>87.3</td>
<td>87.1</td>
<td>6.84</td>
<td>40.4</td>
</tr>
<tr>
<td></td>
<td>VINEP</td>
<td>4.47</td>
<td>-</td>
<td>90.0</td>
<td>90.1</td>
<td>12.32</td>
<td>-</td>
</tr>
<tr>
<td>Swedish</td>
<td>DYNFS</td>
<td>3.60</td>
<td>40.4</td>
<td>88.8</td>
<td>89.0</td>
<td>5.04</td>
<td>47.1</td>
</tr>
<tr>
<td></td>
<td>VINEP</td>
<td>4.64</td>
<td>-</td>
<td>88.3</td>
<td>88.5</td>
<td>13.89</td>
<td>-</td>
</tr>
</tbody>
</table>
We can build systems that learn to trade-off speed vs accuracy (it's hard).

- Requires new algorithms
- Important to model the problem “right”
- Imitation/reinforcement learning applied to non-deterministic search

Orthogonal improvements to most “speedup” type papers

Thanks! Questions?

I'm going to be on the job market soon!