Post Lovelace...
...Pre PvsNP

Post von Humboldt...
...Pre transformation grammar
Why are complex predictions slow?

- Parsing  \# trees \sim O(2^{|\text{sentence}|})
- Translation  \# trans \sim O(2^{\text{foreign} \times \text{english}})
- Summarization  \# 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 \( O(|\text{grammar}| \times |\text{sentence}|^3) \)
  but \( |\text{grammar}| \) is huge

Concretely, \( n^3 = 15000 \)
\( |\text{grammar}| = \frac{1}{2} \text{ million} \)
Learning to be fast!

Quality = tradeoff(accuracy, time)
Dependency parsing

NLP algorithms use a kitchen sink of features
Case study: dependency parsing

The diagram shows the average time (seconds) for different sentence lengths in the following categories:

- Tagging
- Features
- Parsing

For sentences less than 10, the time is consistently less than 0.1 seconds. For sentences between 10 and 20, the time increases to approximately 0.12 seconds. For sentences between 20 and 30, the time increases significantly, reaching around 3.73 seconds for the Tagging category. The Features category stays relatively consistent across these lengths, while the Parsing category shows a more gradual increase, reaching around 32.80 seconds for sentences longer than 50.
Dynamic feature selection

Example

Classifier

h

Remaining Fts

FunnyIP
MailFrom
MailFromWeird
BlacklistedIP
SubjectLength
SubjectWords
BodySize
HasAttachment
...
BodyWords

Predicting linguistic structures

He+Eisner+D, NIPS 2012
Dynamic feature selection

Example

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FunnyIP</td>
<td>1</td>
</tr>
</tbody>
</table>

Classifier

Remaining Fts

- MailFrom
- MailFromWeird
- BlacklistedIP
- SubjectLength
- SubjectWords
- BodySize
- HasAttachment
- ...
- BodyWords
Dynamic feature selection

Example

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FunnyIP</td>
<td>1</td>
</tr>
<tr>
<td>SubjectLength</td>
<td>26</td>
</tr>
</tbody>
</table>

Classifier

Remaining Fts

- MailFrom
- MailFromWeird
- BlacklistedIP
- SubjectWords
- BodySize
- HasAttachment
- ... BodyWords
Dynamic feature selection

Example

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FunnyIP</td>
<td>1</td>
</tr>
<tr>
<td>SubjectLength</td>
<td>26</td>
</tr>
<tr>
<td>HasAttachment</td>
<td>-1</td>
</tr>
</tbody>
</table>

Classifier

+4

Remaining Fts

- MailFrom
- MailFromWeird
- BlacklistedIP
- SubjectWords
- BodySize
- ... BodyWords
Dynamic feature selection

Example
FunnyIP 1
SubjectLength 26
HasAttachment -1

Classifier

Remaining Fts
MailFrom
MailFromWeird
BlacklistedIP
SubjectWords
BodySize
...
BodyWords

h

He+Eisner+D, NIPS 2012
DAgger: Dataset Aggregation

- Collect trajectories from expert $\pi^*$
- Dataset $D_0 = \{ (s, \pi^*(s)) \mid s \sim \pi^* \}$
- Train $\pi_1$ on $D_0$
- Collect new trajectories from $\pi_1$
  - But let the expert steer!
- Dataset $D_1 = \{ (s, \pi^*(s)) \mid s \sim \pi_1 \}$
- Train $\pi_2$ on $D_0 \cup D_1$

In general:
- $D_n = \{ (s, \pi^*(s)) \mid s \sim \pi_n \}$
- Train $\pi_n$ on $\bigcup_{i<n} D_i$

If $N = T \log T$,

$L(\pi_n) < T \epsilon_N + O(1)$

for some $n$
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

Pssst! You should choose $\arg\min_a E[l(a)]$
Dependency parsing

NLP algorithms use a kitchen sink of features

[root]
Dependency parsing

NLP algorithms use a kitchen sink of features (He+Eisner+D, EMNLP 2013)
Dependency parsing

Three steps:
1. Compute POS tags
2. Compute \( kn^2 \) features
3. Run directed MST

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

He+Eisner+D, EMNLP 2013
The system we learn to control

$+$ first feature group 5 features per gray edge
51 gray edge with unknown fate...

$\text{This time }$, the firms were ready.

- Undetermined edge
- Current 1-best tree
- Winner edge
- Loser edge
The system we learn to control

5 features per gray edge
51 gray edge with unknown fate...

$ This time, the firms were ready

Undetermined edge
Current 1-best tree
Winner edge
Loser edge

Non-projective decoding

(He+Eisner+D, EMNLP 2013)
The system we learn to control

5 features per gray edge
50 gray edge with unknown fate...

$ This time, the firms were ready

Undetermined edge
Current 1-best tree
Winner edge
Loser edge

Decide winners among the blue edges
The system we learn to control

5 features per gray edge
44 gray edge with unknown fate...

$ This time, the firms were ready

Undetermined edge
Current 1-best tree
Winner edge
Loser edge

Remove losers in conflict with the winners
The system we learn to control

5 features per gray edge
44 gray edge with unknown fate...

$\text{This time, the firms were ready.}$

- Undetermined edge
- **Current 1-best tree**
- **Winner edge**
- **Loser edge**

*Remove losers in conflict with the winners*
The system we learn to control

+ next feature group  27  features per gray edge
44  gray edge with unknown fate...

$ This time, the firms were ready
The system we learn to control
Static versus dynamic feature selection
Looking inside the box...

- Later features are helpful
- Most edges win or lose early
- Linear increase in runtime
- Some edges win late
Accuracy is (essentially) unaffected

Accuracy comparison for different languages:
- Bulgarian: 99.30%
- Chinese: 99.80%
- English: 99.90%
- German: 100.00%
- Japanese: 100.30%
- Portuguese: 100.30%
- Swedish: 100.00%

Legend:
- DynFS
- VineP
- Baseline
...but we get a lot faster

[Bar chart showing speedup for different languages: Bulgarian, Chinese, English, German, Japanese, Portuguese, Swedish. The chart compares DynFS, VineP, and Baseline.]
Moving to more general frameworks

- Lots of NLP (+al) problems can be cast at test time as integer linear programs
- ILPs are usually solved using branch and bound

Branch and bound involves a complex heuristic search.
Can we learn to perform this search efficiently?
Some intuition

- A good search strategy should:
  - find a good incumbent solution early
  - identify non-promising nodes before expansion

- “Good” varies depending on your position in the tree:
  - DFS should only be used at nodes that promise to lead to a good feasible solution that may replace the incumbent
  - Best-bound-first search can quickly discard unpromising nodes, but should not be used frequently at the top

- We will learn a heuristic based on features that can capture this intuition
Training and experiments

- Same algorithm as before
- Four (standard) ILP datasets (non-NLP-based)
- Comparison to:
  - DFS (baseline)
  - Gurobi (thousands of person-hours of effort)

Measures:
- Optimality Gap, Integrality Gap, and improvement from initial heuristic solution

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ours (DAgger training)</th>
<th>DFS</th>
<th>Gurobi</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OGap</td>
<td>IGap</td>
<td>Impr</td>
</tr>
<tr>
<td>MIK</td>
<td>0.23</td>
<td>16.63</td>
<td>4.39</td>
</tr>
<tr>
<td>Regions1</td>
<td>0.54</td>
<td>4.53</td>
<td>10.57</td>
</tr>
<tr>
<td>Regions2</td>
<td>1.22</td>
<td>6.76</td>
<td>19.36</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.87</td>
<td>20.28</td>
<td>24.46</td>
</tr>
</tbody>
</table>

We achieve less than 1.2 optimality gap while exploring 0.05%, 1.5%, 5.1% and 47% of the nodes explored by Gurobi!
Simultaneous (machine) interpretation

➢ Dozens of defendants
➢ Judges from four nations (three languages)
➢ Status quo: speak, then translate
➢ After Nuremberg, simultaneous translations became the norm
➢ Long wait → bad conversation

Nuremburg Trials

(Grissom II, Boyd-Graber, He, Morgan & D, EMNLP 2014)
Why simultaneous interpretation is hard

- Human languages have vastly different word orders
  - About half are OV, the other half are VO
  - This comes with a lot more baggage than just verb-final

Running (German/English) Example:

Ich bin mit dem Zug nach Ulm gefahren
I am with the train to Ulm traveled
I (...... waiting.......) traveled by train to Ulm
Model for interpretation decisions

➢ We have a set of actions (predict / translate)
  ➢ Wait
  ➢ Predict clause-verb
  ➢ Predict next word
  ➢ Commit ("speak")

➢ In a changing environment (state)
  ➢ The words we've seen so far
  ➢ Our models' internal predictions

➢ With a well defined oracle
Example of interpretation trajectory

**Observation**
1. Mit dem Zug

**State**
- Verb: gewesen
- Next: und

Ich bin mit dem Zug nach Ulm gefahren
I am with the train to Ulm traveled
I (…… waiting……) traveled by train to Ulm
Evaluating performance and baselines
Evaluating performance and baselines

![Graph showing smoothed average performance vs percent of sentence for different methods: Batch, Monotone, Optimal, Learned. The graph illustrates the performance comparison with smooth lines for each method across the sentence percentage.](Grisson I, Boyd-Graber, He, Morgan & D, EMNLP 2014)
Some other fun stuff...

Come to vowpal wabbit tutorial
1:30-2:45 in Opt wksp, 513ef
• Language is ambiguous (→ statistics) and combinatorial (→ algorithms)
• Arbitrarily flexible algorithms allow us (==NLPers) to solve problems without handcuffs
• As problems get larger, we must not look at the “entire input” to make decisions

Thanks! Questions?