Syntactic Category Learning
as Prototype-Driven Clustering

Jordan Kodner
University of Pennsylvania

SCiL, Jan. 4, 2018
Salt Lake City, UT
Learning Syntactic Categories

- Abstract labels corresponding to nodes in syntactic trees
  
  **N** - noun  **V** - verb  **PREP** - preposition  etc.

- Cognitive equivalent of part-of-speech tags
Learning Syntactic Categories

- Abstract labels corresponding to nodes in syntactic trees
  
  N - noun  V - verb  PREP - preposition etc.

- Cognitive equivalent of part-of-speech tags
- Early learning (post-segmentation)
- Categories are learned from distributional cues
- Syntactic frames (Mintz 2003)

“The __ is” → N  “Where __ you” → V  “pick __ the” → PREP

“D __ V” → N  “Q __ P” → V  “V __ D” → PREP
Chicken-and-Egg Problem

Children learn POS on the basis of (POS) context
But POS context depends on learning POS
Chicken-and-Egg Problem

Children learn POS on the basis of (POS) context
But POS context depends on learning POS

- Semantic bootstrapping (Pinker 1984)
- Innately anchor some words into real world concepts
- e.g., actions should be verbs, objects should be nouns, etc.
Analogues in POS-tagging

- Unsupervised tagging
- Processing distributional cues with statistical methods
- Results in clustering and labeling
Analogues in POS-tagging

- Unsupervised tagging
- Processing distributional cues with statistical methods
- Results in clustering and labeling
- The Cutting Problem

Parkes et al. 1998
Analogues in POS-tagging

- Unsupervised tagging
- Processing distributional cues with statistical methods
- Results in clustering and labeling
- The Cutting Problem

- Blue - $VBD_1$, $VBD_2$, $VBZ$
Analogues in POS-tagging

- Unsupervised tagging
- Processing distributional cues with statistical methods
- Results in clustering and labeling
- The Cutting Problem

- Blue - VBD₁, VBD₂, VBZ
- Gold - VBD, VBD/VBZ

Parkes et al. 1998
Prototype-Driven POS-Tagging

- cf. Haghighi & Klein 2006
- Minimally-supervised approach
- Words are tagged by similarity with *prototypes*
- 3 most common words per tag in WSJ corpus
- Markov Random Field model
Prototype-Driven POS-Tagging

- cf. Haghighi & Klein 2006
- Minimally-supervised approach
- Words are tagged by similarity with prototypes
- 3 most common words per tag in WSJ corpus
- Markov Random Field model

Word-internal features
- uni-, bi-, trigram char suffixes
- initial_capital
- contains_hyphen
- contains_digit

Word-external features
- left and right contexts (len=2)

(Edge feature)
- tag trigrams
Prototype-Driven POS-Tagging *Synt. Cat. Learning*

- cf. Haghighi & Klein 2006
- Minimally-supervised approach
- Words are tagged by similarity with *prototypes*
- 3 most common words per tag in WSJ corpus
- Markov Random Field model

**Word-internal features**
- uni-, bi-, trigram char suffixes
- *initial_capital*
- *contains_hyphen*
- *contains_digit*

**Word-external features**
- left and right contexts (len=2)

*(Edge feature)*
- tag trigrams
Prototype-Driven POS-Tagging

- cf. Haghighi & Klein 2006
- Minimally-supervised approach
- Words are tagged by similarity with prototypes
- 3 most common words per tag in WSJ corpus
- Markov Random Field model

Word-internal features
- uni-, bi-, trigram char suffixes
- initial\_capital
- contains\_hyphen
- contains\_digit

Word-external features
- left and right contexts (len=2)
- tag trigrams
Prototype-Driven POS Tagging

- cf. Haghighi & Klein 2006
- Minimally-supervised approach
- Words are tagged by similarity with prototypes
- 3 most common words per tag in WSJ corpus
- Markov Random Field model

Word-internal features:
- uni-, bi-, trigram character suffixes
- initial_cap
- contains_hyphen
- contains_digit

Word-external features:
- left and right contexts (len=2)
  (Edge feature)
- tag trigrams
Prototype-Driven POS-Tagging

- cf. Haghighi & Klein 2006
- Minimally-supervised approach
- Words are tagged by similarity with prototypes
- 3 most common words per tag in WSJ corpus
- Markov Random Field model

Word-internal features
- uni-, bi-, trigram char suffixes
- initial_cap
- contains_hyphen
- contains_digit

Word-external features
- left and right contexts (len=2)

Semantic Bootstrapping

Syntactic Frames (len=1)
Prototype-Driven Syntactic Category Learning

Semantic Bootstrapping
- Selection of initial “seeds”
- $s$ seeds per category
Prototype-Driven Syntactic Category Learning

Semantic Bootstrapping
● Selection of initial “seeds”
● $s$ seeds per category

Syntactic Frames
● Only track distributions of left and right contexts
● Models early learning
● Only train for frequent types
Prototype-Driven Syntactic Category Learning

Semantic Bootstrapping
- Selection of initial “seeds”
- $s$ seeds per category

Syntactic Frames
- Only track distributions of left and right contexts
- Models early learning
- Only train for frequent types

Cognitive Modeling
- Algorithmic representation
- Simple computations
- Does not require a specific tag set
Prototype-Driven Syntactic Category Learning

**Semantic Bootstrapping**
- Selection of initial “seeds”
- $s$ seeds per category

**Syntactic Frames**
- Only track distributions of left and right contexts
- Models early learning
- Only train for frequent types

**Cognitive Modeling**
- Algorithmic representation
- Simple computations
- Does not require a specific tag set
- Vanilla agglomerative clustering
- KL-distance between context vecs
- Apply iteratively as vocab size increases
Labeling Algorithm

1) Calculate distance matrix on top $k$ types
2) Agglomerative clustering of top $k$
3) Label seed leaves
4) For each join in rank order,
   If one subtree is assigned and the other unassigned,
   Assign all unassigned leaves the most frequent tag in the assigned subtree
Labeling Algorithm

1) Calculate distance matrix on top $k$ types
2) Agglomerative clustering of top $k$
3) Label seed leaves
4) For each join in rank order,
   If one subtree is assigned and the other unassigned,
     Assign all unassigned leaves the most frequent tag in the assigned subtree

- Simple KL-divergence
- Symmetricized: $\text{KL}(P||Q) + \text{KL}(Q||P)$
Labeling Algorithm

1) Calculate distance matrix on top $k$ types
2) Agglomerative clustering of top $k$
3) Label seed leaves
4) For each join in rank order,
   If one subtree is assigned and the other unassigned,
       Assign all unassigned leaves the most frequent tag in the assigned subtree

- Average linkage criterion
- Seconds for $k=1,000$
- Minutes for $k=10,000$
Labeling Algorithm

1) Calculate distance matrix on top $k$ types
2) Agglomerative clustering of top $k$
3) Label seed leaves
4) For each join in rank order,
   If one subtree is assigned and the other unassigned,
   Assign all unassigned leaves the most frequent tag in the assigned subtree

- Label each with its most frequent tag
- No requirement that seeds have single unique tag
Labeling Algorithm

1) Calculate distance matrix on top $k$ types
2) Agglomerative clustering of top $k$
3) Label seed leaves
4) For each join in rank order,
   If one subtree is assigned and the other unassigned,
   Assign all unassigned leaves the most frequent tag in the assigned subtree

Base case: seed leaf & unassigned leaf
Labeling Algorithm

1) Calculate distance matrix on top $k$ types
2) Agglomerative clustering of top $k$
3) Label seed leaves
4) For each join in rank order,
   If one subtree is assigned and the other unassigned,
     Assign all unassigned leaves the most frequent tag in the assigned subtree
Labeling Algorithm

1) Calculate distance matrix on top $k$ types
2) Agglomerative clustering of top $k$
3) Label seed leaves
4) For each join in rank order,
   If one subtree is assigned and the other unassigned,
     Assign all unassigned leaves the most frequent tag in the assigned subtree

General case: assigned and unassigned subtrees

10% N
90% V

?
Labeling Algorithm

1) Calculate distance matrix on top $k$ types
2) Agglomerative clustering of top $k$
3) Label seed leaves
4) For each join in rank order,
   If one subtree is assigned and the other unassigned,
   Assign all unassigned leaves the most frequent tag in the assigned subtree

General case: assigned and unassigned subtrees

10% N  100% V
90% V
Iterative Algorithm

For increasing $k$,
1) Perform *Labeling Algorithm*
2) Define confidence for each assignment
3) Add high confidence assignments to seed set
4) Assign highest confidence tag to each word
Iterative Algorithm

For increasing $k$,

1) Perform *Labeling Algorithm*

2) Define confidence for each assignment

3) Add high confidence assignments to seed set

4) Assign highest confidence tag to each word

- For example, $k = (100, 500, 900, 1000)$
- How to set $k$ sequence?
Iterative Algorithm

For increasing $k$,
1) Perform *Labeling Algorithm*
2) Define confidence for each assignment
3) Add high confidence assignments to seed set
4) Assign highest confidence tag to each word

- Purity of the assigning subtree
- High purity trees more likely to represent true clusters
- Range: $[0,1]$
Iterative Algorithm

For increasing $k$,
1) Perform *Labeling Algorithm*
2) Define confidence for each assignment
3) Add high confidence assignments to seed set
4) Assign highest confidence tag to each word

- Parameter from 0 (add all) to $>1$ (add none)
- Goal is to grow seed set as much as possible while retaining high accuracy
- **How to set confidence threshold?**
Iterative Algorithm

For increasing $k$,
1) Perform *Labeling Algorithm*
2) Define confidence for each assignment
3) Add high confidence assignments to seed set

4) Assign highest confidence tag to each word

- Words not added to the seed set are re-assigned at each iteration
- For the final assignment, choose the tag that the system was most confident about at any point
Seed Selection

- Seeds account for ~1-10% of types
- Automated testing: three most frequent types per tag
- Cognitively motivated: three salient types per tag

<table>
<thead>
<tr>
<th>Tag Set</th>
<th># Tags</th>
<th>Max # Seeds</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHILDES (Brown)</td>
<td>55</td>
<td>165</td>
</tr>
<tr>
<td>Universal Treebank</td>
<td>12</td>
<td>36</td>
</tr>
<tr>
<td>Wall Street Journal</td>
<td>45</td>
<td>135</td>
</tr>
<tr>
<td>Chinese Treebank</td>
<td>35</td>
<td>105</td>
</tr>
</tbody>
</table>
## WSJ Seed Examples

<table>
<thead>
<tr>
<th>Part of Speech</th>
<th>Example</th>
<th>Part of Speech</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>year, market, company</td>
<td>WP</td>
<td>what, who, whom</td>
</tr>
<tr>
<td>NNP</td>
<td>Mr., U.S., Corp.</td>
<td>WRB</td>
<td>where, when, how</td>
</tr>
<tr>
<td>NNS</td>
<td>years, shares, sales</td>
<td>RB</td>
<td>not, also, n’t</td>
</tr>
<tr>
<td>VBD</td>
<td>said, was, were</td>
<td>IN</td>
<td>for, of, in</td>
</tr>
<tr>
<td>VBZ</td>
<td>has, says, is</td>
<td>JJ</td>
<td>other, last, new</td>
</tr>
<tr>
<td>VBN</td>
<td>made, been, expected</td>
<td>JJS</td>
<td>least, largest, most</td>
</tr>
<tr>
<td>Part of Speech</td>
<td>Example 1</td>
<td>Example 2</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>year, market, company</td>
<td>what, who, whom</td>
<td></td>
</tr>
<tr>
<td>NNP</td>
<td>Mr., U.S., Corp.</td>
<td>where, when, how</td>
<td></td>
</tr>
<tr>
<td>NNS</td>
<td>years, shares, sales</td>
<td>not, also, n’t</td>
<td></td>
</tr>
<tr>
<td>VBD</td>
<td>said, was, were</td>
<td>for, of, in</td>
<td></td>
</tr>
<tr>
<td>VBZ</td>
<td>has, says, is</td>
<td>other, last, new</td>
<td></td>
</tr>
<tr>
<td>VBN</td>
<td>made, been, expected</td>
<td>least, largest, most</td>
<td></td>
</tr>
<tr>
<td>FW</td>
<td>de, kanji, Perestroika</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Salient Seeds Examples

DT one, this, three

NN arm, baby, bed

JJ big, black, happy

RB down, not, off

VB break, climb, come

IN in, by, from

PRP him, we, who
Evaluation Metrics

1-to-Many Type Accuracy
- Types have ≥1 gold assignment
- Types are marked correct if their assignments are among their gold assignments
- Seed baseline usually ~1-10%
- Useful metric for syn. category learning as lexicon building

1-to-1 Token Accuracy
- Tokens have 1 gold assignment
- Tokens are marked correct if their assignments match their gold assignments exactly
- Seed baseline potentially >50%
- Useful metric for POS-tagging as the end goal
CHILDES Type Accuracy

- Works well on for smaller $k$
- Deteriorates for bigger $k$
- Seeds by frequency
- Brown tag set
- CHILDES Brown
  - 8,307 types
  - 588,888 tokens

<table>
<thead>
<tr>
<th>$k$</th>
<th># Seeds</th>
<th>Baseline</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>58</td>
<td>58%</td>
<td>94.0%</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>10%</td>
<td>81.2%</td>
</tr>
<tr>
<td>8307</td>
<td>130</td>
<td>1.6%</td>
<td>62.8%</td>
</tr>
</tbody>
</table>
## Large Tag Set vs. Small Tag Set

- **Brown tag set**
- **Reduced 8-tag set**
- **Test 3 and 11 seeds**

<table>
<thead>
<tr>
<th>$k$</th>
<th>Tag Set</th>
<th># Seeds</th>
<th>Baseline</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Brown</td>
<td>100</td>
<td>10%</td>
<td>81.2%</td>
</tr>
<tr>
<td>1000</td>
<td>Reduced</td>
<td>24</td>
<td>2.4%</td>
<td>51.8%</td>
</tr>
<tr>
<td>1000</td>
<td>Reduced</td>
<td>85</td>
<td>8.5%</td>
<td>80.6%</td>
</tr>
<tr>
<td>8307</td>
<td>Brown</td>
<td>130</td>
<td>1.6%</td>
<td>62.8%</td>
</tr>
<tr>
<td>8307</td>
<td>Reduced</td>
<td>24</td>
<td>0.3%</td>
<td>25.3%</td>
</tr>
<tr>
<td>8307</td>
<td>Reduced</td>
<td>85</td>
<td>1.0%</td>
<td>53.3%</td>
</tr>
</tbody>
</table>
Single vs. Iterative Labeling Algorithm

- $k = (100, 200, 500, 900, 1000, 2000, 5000, 8307)$
- Iterative application outperforms regardless of tag set and number of seeds

<table>
<thead>
<tr>
<th>$k$</th>
<th>Tag Set</th>
<th># Seeds</th>
<th>Single</th>
<th>Iterative</th>
</tr>
</thead>
<tbody>
<tr>
<td>8307</td>
<td>Brown</td>
<td>130</td>
<td>44.2%</td>
<td>62.8%</td>
</tr>
<tr>
<td>8307</td>
<td>Reduced</td>
<td>24</td>
<td>9.3%</td>
<td>25.3%</td>
</tr>
<tr>
<td>8307</td>
<td>Reduced</td>
<td>85</td>
<td>44.0%</td>
<td>53.3%</td>
</tr>
</tbody>
</table>
Frequent vs. Salient Seeds

- Salient performance is lower
- But so is the salient baseline

<table>
<thead>
<tr>
<th></th>
<th>Salient</th>
<th>Frequent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k$</td>
<td># Seeds</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>74</td>
</tr>
<tr>
<td>8307</td>
<td>8307</td>
<td>82</td>
</tr>
</tbody>
</table>
**UTB Type Accuracies**

- Wide range of results
- Divergent results even on relatives
- But, Romance > Germanic for $k=10,000$

<table>
<thead>
<tr>
<th>Corpus</th>
<th># Seeds</th>
<th>$k = 1,000$</th>
<th>$k = 10,000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>French</td>
<td>28</td>
<td>77.92%</td>
<td>62.07%</td>
</tr>
<tr>
<td>German</td>
<td>30</td>
<td>79.04%</td>
<td>26.62%</td>
</tr>
<tr>
<td>Indonesian</td>
<td>30</td>
<td>75.84%</td>
<td>65.21%</td>
</tr>
<tr>
<td>Italian</td>
<td>26</td>
<td>54.26%</td>
<td>37.08%</td>
</tr>
<tr>
<td>Japanese</td>
<td>24</td>
<td>47.78%</td>
<td>48.31%</td>
</tr>
<tr>
<td>Korean</td>
<td>26</td>
<td>33.47%</td>
<td>39.19%</td>
</tr>
<tr>
<td>Portuguese</td>
<td>38</td>
<td>65.40%</td>
<td>49.44%</td>
</tr>
<tr>
<td>Spanish</td>
<td>29</td>
<td>63.41%</td>
<td>46.14%</td>
</tr>
<tr>
<td>Swedish</td>
<td>37</td>
<td>51.10%</td>
<td>33.96%</td>
</tr>
</tbody>
</table>
What is Wrong with Korean (and Japanese)?

- The corpus has an unusually high type/token ratio 36329/69690 = 0.52
- Only 26 of 36 possible seeds occur in the top 1000
- *Eojeol/Bunsetsu* Tokenization
  - Particles and postpositions are not separated
  - Punctuation is not separated
What is Wrong with Korean (and Japanese)?

- The corpus has an unusually high type/token ratio \( \frac{36329}{69690} = 0.52 \)
- Only 26 of 36 possible seeds occur in the top 1000
- *Eojeol/Bunsetsu* Tokenization
  - Particles and postpositions are not separated
  - Punctuation is not separated

환경 문제는 지금 4대강 사업의 최대 쟁점이 되 있다.

“The biggest issue with the Four Rivers Project has become the environmental problem.”
What is Wrong with Korean (and Japanese)?

- **Eojeol/Bunsetsu Tokenization**
  - Particles and postpositions are not separated
  - Punctuation is not separated
- **Prevents useful generalization**

<table>
<thead>
<tr>
<th>Tokenization</th>
<th>Text Strings</th>
<th>Right Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunsetsu</td>
<td>ringo-ga X</td>
<td>ringo-ga: {X}</td>
</tr>
<tr>
<td></td>
<td>ringo-wo Y</td>
<td>ringo-wo: {Y}</td>
</tr>
<tr>
<td></td>
<td>nashi-ga Z</td>
<td>nashi-ga: {Z}</td>
</tr>
<tr>
<td></td>
<td>nashi-wo W</td>
<td>nashi-wo: {W}</td>
</tr>
<tr>
<td>Standalone</td>
<td>ringo ga X</td>
<td>ringo: {ga, wo}</td>
</tr>
<tr>
<td></td>
<td>ringo wo Y</td>
<td>nashi: {ga, wo}</td>
</tr>
<tr>
<td></td>
<td>nashi ga Z</td>
<td>ga: {X, Z}</td>
</tr>
<tr>
<td></td>
<td>nashi wo W</td>
<td>wo: {Y, W}</td>
</tr>
</tbody>
</table>
Extension for Token Accuracy Scoring

- After the top $k$ are classified, the remaining $n-k$ types are assigned the most common POS of the nearest seed (KL-distance)
Extension for Token Accuracy Scoring

- After the top $k$ are classified, the remaining $n-k$ types are assigned the most common POS of the nearest seed (KL-distance)

Three token accuracy scores:

1) **Top $k$**  
   words outside the top $k$ are not counted

2) **All**  
   words outside the top $k$ are incorrect

3) **All+**  
   words outside the top $k$ are classified by nearest seed
CHILDES Token Accuracies

- $k = 8,307$
- Frequent seeds

<table>
<thead>
<tr>
<th>Tag Set</th>
<th># Seeds</th>
<th>Baseline</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown</td>
<td>130</td>
<td>50.2%</td>
<td>82.7%</td>
</tr>
<tr>
<td>Reduced</td>
<td>24</td>
<td>28.8%</td>
<td>60.0%</td>
</tr>
<tr>
<td>Reduced</td>
<td>85</td>
<td>52.3%</td>
<td>83.5%</td>
</tr>
</tbody>
</table>
## WSJ Treebank Token Accuracies

- H&K PROTO represents closest comparison
- H&K PROTO: Word-external + **internal** features on the same set of WSJ

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Seeds</th>
<th>Type</th>
<th>Baseline</th>
<th>Accuracy</th>
<th>Baseline</th>
<th>Top $k$</th>
<th>All</th>
<th>All+</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k=1,000$</td>
<td>95</td>
<td></td>
<td>9.5%</td>
<td>57.9%</td>
<td>40.5%</td>
<td>74.3%</td>
<td>54.7%</td>
<td>60.2%</td>
</tr>
<tr>
<td>$k=10,000$</td>
<td>95</td>
<td></td>
<td>1.0%</td>
<td>30.2%</td>
<td>40.5%</td>
<td>63.2%</td>
<td>60.9%</td>
<td>61.4%</td>
</tr>
<tr>
<td>H&amp;K PROTO</td>
<td>135</td>
<td></td>
<td>-</td>
<td>-</td>
<td>41.3%</td>
<td>-</td>
<td>68.8%</td>
<td>-</td>
</tr>
</tbody>
</table>
## Chinese Treebank Token Accuracies

- **H&K PROTO**: Word-external *(no internal)* features
- Appears that H&K relies on internal features

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Seeds</th>
<th>Type</th>
<th>Token</th>
<th>Baseline</th>
<th>Top k</th>
<th>All</th>
<th>All+</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Baseline</td>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Baseline</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k=1,000$</td>
<td>74</td>
<td>7.4%</td>
<td>50.4%</td>
<td>29.5%</td>
<td>62.8%</td>
<td>46.9%</td>
<td>50.4%</td>
</tr>
<tr>
<td>$k=8,842$</td>
<td>74</td>
<td>0.8%</td>
<td>27.5%</td>
<td>29.5%</td>
<td>-</td>
<td>54.1%</td>
<td>-</td>
</tr>
<tr>
<td>H&amp;K PROTO</td>
<td>99</td>
<td>-</td>
<td>-</td>
<td>34.4%</td>
<td>-</td>
<td>39.0%</td>
<td>-</td>
</tr>
</tbody>
</table>
Future Directions

Include word-internal features

- Obviously involved
- Especially later
Future Directions

Include word-internal features
● Obviously involved
● Especially later

Include POS contexts
● As opposed to lexical contexts
● e.g., D ___ N → A
Future Directions

Include word-internal features
- Obviously involved
- Especially later

Include POS contexts
- As opposed to lexical contexts
- e.g., D __ N → A

Model multiple category assignments
- Homophony is a thing
Acknowledgements:
● Charles Yang
● Mitch Marcus
● DARPA LORELEI
● NDSEG

Implementation:
github.com/jkodner05/LowResPOS