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

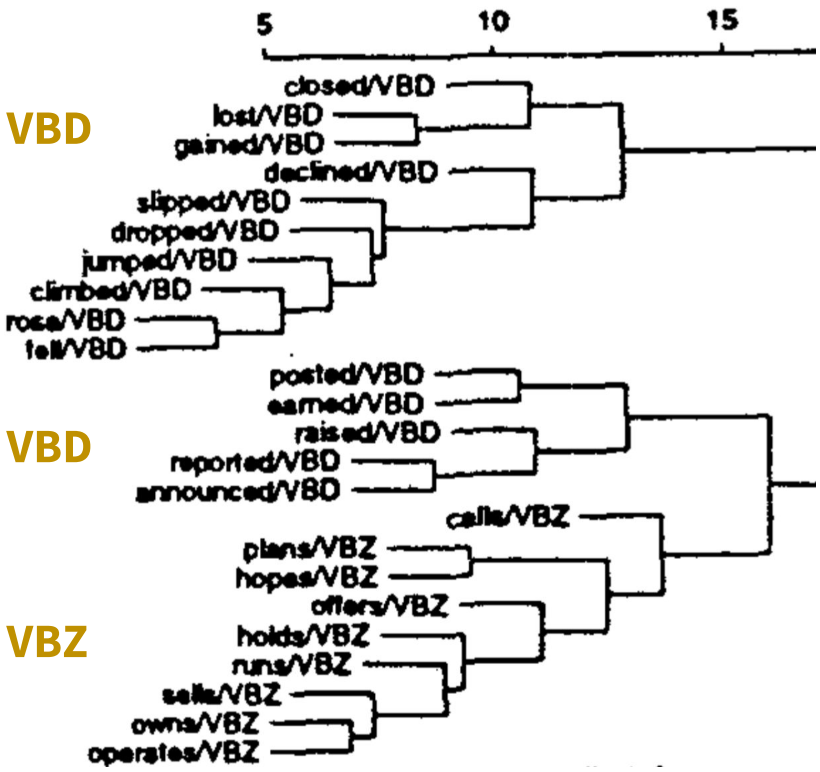
- Semantic bootstrapping (Pinker 1984)
- Innately anchor some words into real world concepts
- e.g., actions should be verbs, objects should be nouns, etc.

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

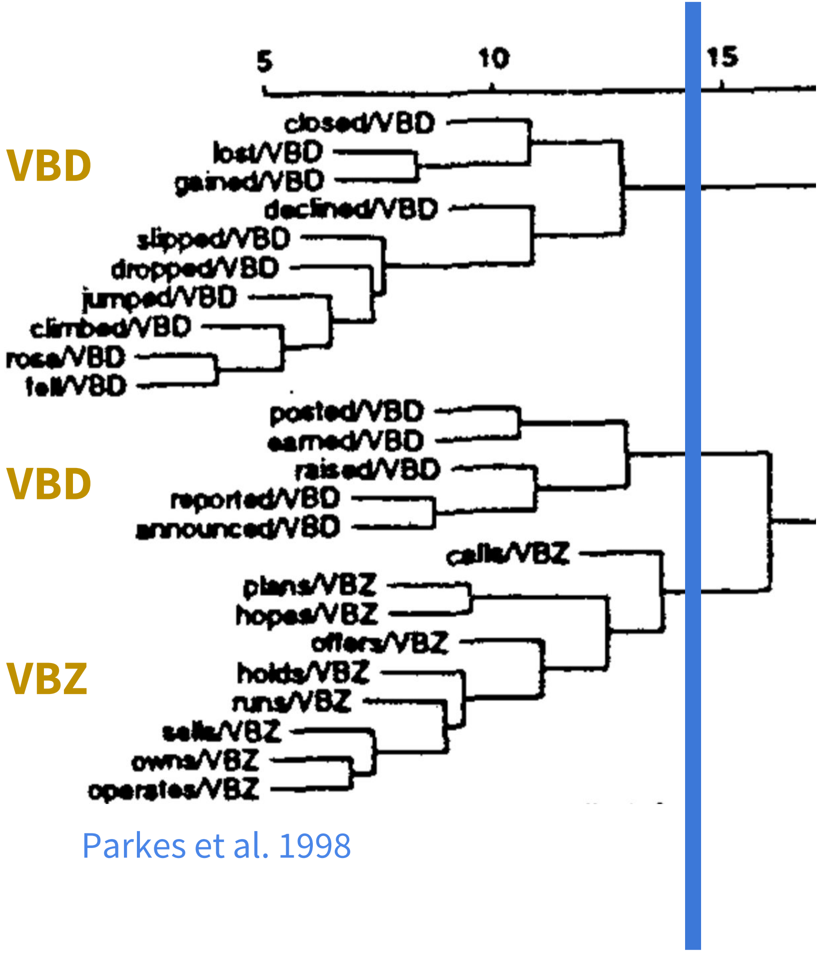


Parkes et al. 1998

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

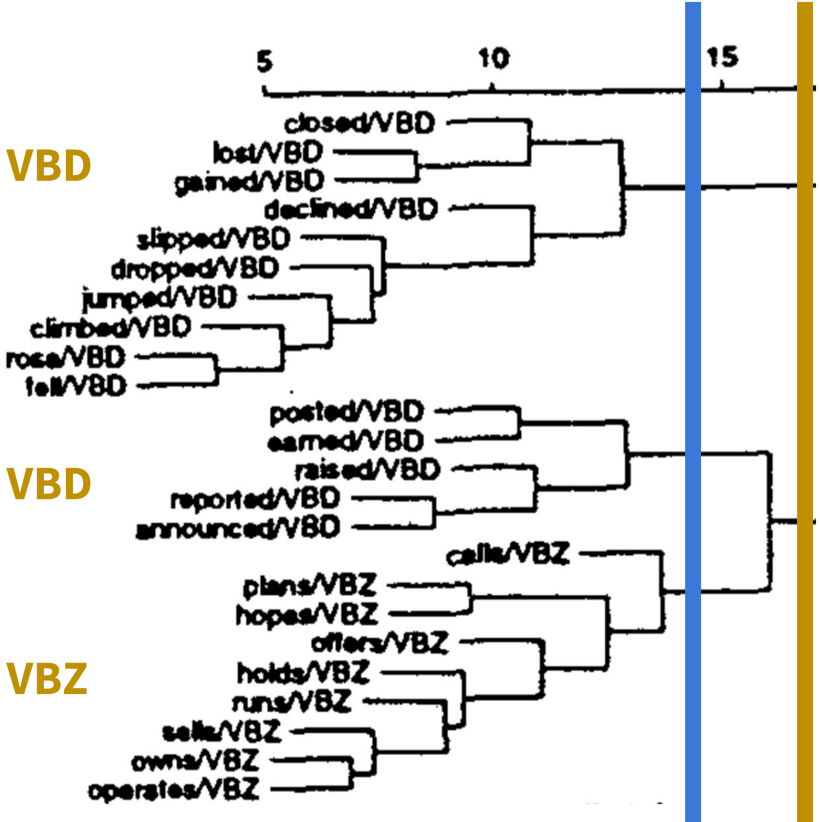


Parkes et al. 1998

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
- 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 PO[illegible]ing~~ *Synt. Cat. Learning*

- cf. Haghighi & Klein 2006
- Minimally-supervised approach
- Words are tagged by similarity with *prototypes*
- ~~3 m[illegible] common words [illegible] tag in WSJ co[illegible]~~
- ~~Mark[illegible] del~~

Word-internal features

uni-, bi-, trigram char suffixes

initial_capital

contains_hyphen

contains_digit

Word-external features

left and right contexts (len=2)

~~**(Ed[illegible]feature)**~~

~~ta[illegible]ns~~

Prototype-Driven ~~PO… …ing~~ *Synt. Cat. Learning*

- cf. Haghighi & Klein 2006
- Minimally-supervised approach
- Words are tagged by similarity with *prototypes*
- ~~3 m… common words … tag in WSJ co…~~
- ~~Mark… …del~~

Word-internal features

~~un… …rigra… …uffixes~~

~~*initial_…*~~

~~*co… …p…*~~

~~*co… …ns_digit*~~

Word-external features

left and right contexts (len=2)

~~**(Ed… feature)**~~

~~ta… …ns~~

Prototype-Driven ~~PO... ...ging~~ *Synt. Cat. Learning*

- ~~cf. Haghighi & Klein 2006~~
- ~~Minimally-supervised approach~~
- Words are tagged by similarity with ***prototypes***
- ~~3 m... common words p... tag in WS...~~
- ~~Mark...~~

Semantic Bootstrapping

~~Word-int... features~~

~~un... ...rigra... ...uffixes~~

~~*initial_...*~~

~~*co...*~~

~~*con...ins_digit*~~

Word-external features

left and right contexts ~~(len=2)~~

~~(Ed... ...eatu...e)~~

~~ta... ...ns~~

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: KL(P||Q) + 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

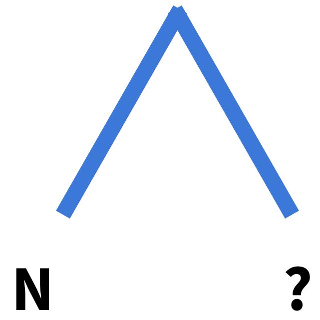
- Label each with its most frequent tag
- No requirement that seeds have single unique tag

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

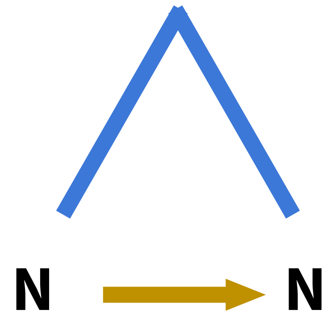
Base case: seed leaf & unassigned leaf



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

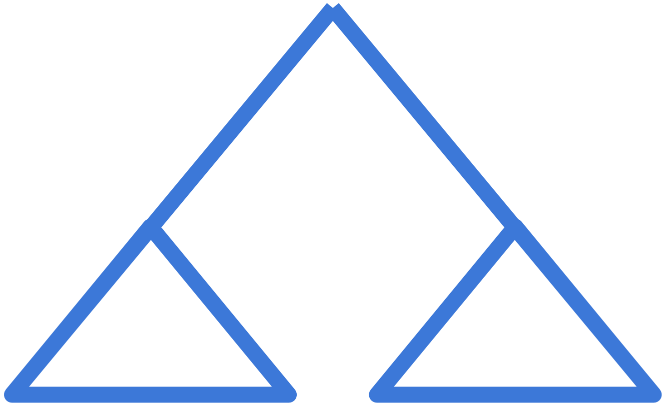
Base case: seed leaf & unassigned leaf



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

General case: assigned and unassigned subtrees



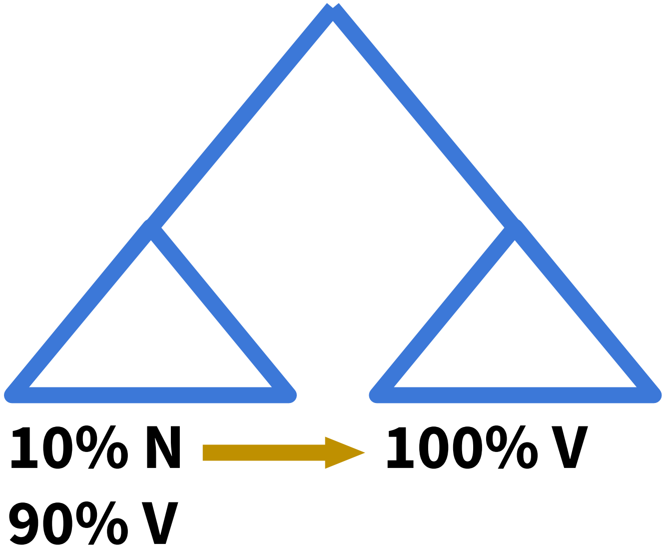
10% N
90% V

?

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



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

Tag Set	# Tags	Max # Seeds
CHILDES (Brown)	55	165
Universal Treebank	12	36
Wall Street Journal	45	135
Chinese Treebank	35	105

WSJ Seed Examples

NN	year, market, company	WP	what, who, whom
NNP	Mr., U.S., Corp.	WRB	where, when, how
NNS	years, shares, sales	RB	not, also, n't
VBD	said, was, were	IN	for, of, in
VBZ	has, says, is	JJ	other, last, new
VBN	made, been, expected	JJS	least, largest, most

WSJ Seed Examples

NN	year, market, company	WP	what, who, whom
NNP	Mr., U.S., Corp.	WRB	where, when, how
NNS	years, shares, sales	RB	not, also, n’t
VBD	said, was, were	IN	for, of, in
VBZ	has, says, is	JJ	other, last, new
VBN	made, been, expected	JJS	least, largest, most
		FW	de, kanji, Perestroika

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

k	# Seeds	Baseline	Score
100	58	58%	94.0%
1000	100	10%	81.2%
8307	130	1.6%	62.8%

Large Tag Set vs. Small Tag Set

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

k	Tag Set	# Seeds	Baseline	Score
1000	Brown	100	10%	81.2%
1000	Reduced	24	2.4%	51.8%
1000	Reduced	85	8.5%	80.6%
8307	Brown	130	1.6%	62.8%
8307	Reduced	24	0.3%	25.3%
8307	Reduced	85	1.0%	53.3%

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

k	Tag Set	# Seeds	Single	Iterative
8307	Brown	130	44.2%	62.8%
8307	Reduced	24	9.3%	25.3%
8307	Reduced	85	44.0%	53.3%

Frequent vs. Salient Seeds

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

Salient				Frequent	
k	# Seeds	Baseline	Score	Baseline	Score
1000	74	7.4%	73.4%	10%	81.2%
8307	82	0.8%	49.5%	1.6%	62.8%

UTB Type Accuracies

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

Corpus	# Seeds	*k* = 1,000	*k* = 10,000
French	28	77.92%	62.07%
German	30	79.04%	26.62%
Indonesian	30	75.84%	65.21%
Italian	26	54.26%	37.08%
Japanese	24	47.78%	48.31%
Korean	26	33.47%	39.19%
Portuguese	38	65.40%	49.44%
Spanish	29	63.41%	46.14%
Swedish	37	51.10%	33.96%

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 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	돼	있다.
	problem- TOP		4- Rivers	project- GEN		issue- NOM		is- PUNCT

"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**

Tokenization	Text Strings	Right Frames
Bunsetsu	ringo-ga X	ringo-ga: {X}
	ringo-wo Y	ringo-wo: {Y}
	nashi-ga Z	nashi-ga: {Z}
	nashi-wo W	nashi-wo: {W}
Standalone	ringo ga X	ringo: {ga, wo}
	ringo wo Y	nashi: {ga, wo}
	nashi ga Z	ga: {X, Z}
	nashi wo W	wo: {Y, W}

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

Tag Set	# Seeds	Baseline	Score
Brown	130	50.2%	82.7%
Reduced	24	28.8%	60.0%
Reduced	85	52.3%	83.5%

WSJ Treebank Token Accuracies

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

		Type		Token			
Algorithm	# Seeds	Baseline	Accuracy	Baseline	Top *k*	All	All+
k=1,000	95	9.5%	57.9%	40.5%	74.3%	54.7%	60.2%
k=10,000	95	1.0%	30.2%	40.5%	63.2%	60.9%	61.4%
H&K PROTO	135	-	-	41.3%	-	68.8%	-

Chinese Treebank Token Accuracies

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

		Type		Token			
Algorithm	# Seeds	Baseline	Accuracy	Baseline	Top *k*	All	All+
k=1,000	74	7.4%	50.4%	29.5%	62.8%	46.9%	50.4%
k=8,842	74	0.8%	27.5%	29.5%	-	54.1%	-
H&K PROTO	99	-	-	34.4%	-	39.0%	-

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

End

Acknowledgements:

- Charles Yang
- Mitch Marcus
- DARPA LORELEI
- NDSEG

Implementation:

`github.com/jkodner05/LowResPOS`