

Lecture 3: Learning Transformation Rules

Kyle Richardson

kyle@ims.uni-stuttgart.de

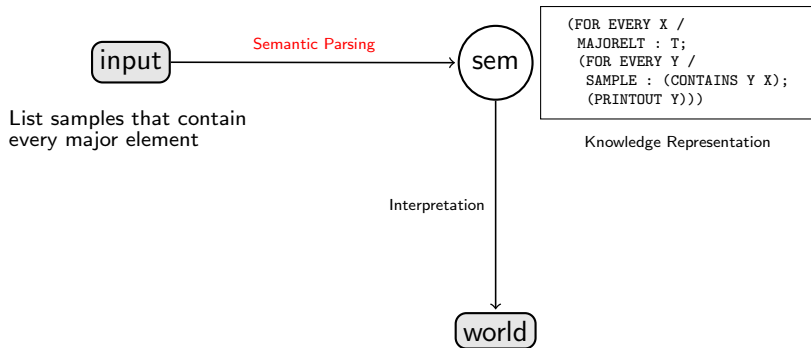
July 4, 2016

Lecture Plan

- ▶ **paper:** ?
- ▶ **general topics:** transformation and rewrite rules, the SILT algorithm, the CKY algorithm, PCFGs, the EM algorithm.

The Big Picture (reminder)

► Standard processing pipeline



$$\llbracket sem \rrbracket = \{S10019, S10059, \dots\}$$

Previous session: Learning from meaning representations

data: $(x = \text{two times two plus three}, y = (\text{plus } (\text{mult } 2 \ 2) \ 3))$

- ▶ **Compositional model** : a semantic context-free grammar.
- ▶ **Learning model**: linear classifier on derivation trees.

Previous session: Learning from meaning representations

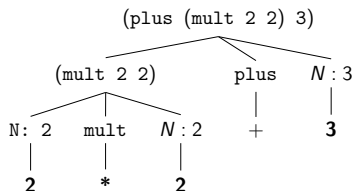
data: ($x = \text{two times two plus three}, y = (\text{plus } (\text{mult } 2 \ 2) \ 3)$)

- ▶ **Compositional model** : a semantic context-free grammar.
- ▶ **Learning model:** linear classifier on derivation trees.
- ▶ **Missing**
 - ▶ **Rule extraction** : (local) rules that get us from $x \rightarrow y$
 - ▶ **Parsing algorithm:** generate derivations for a given input x using such rules.

Transformation and Rewrite Rules

- Decompose translation into a set of *local* transformations.

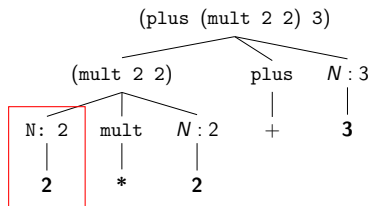
data: $(x = \text{two multiplied by two plus three}, y = (\text{plus } (\text{mult } 2 \ 2) \ 3))$



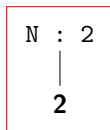
Transformation and Rewrite Rules

- Decompose translation into a set of *local* transformations.

data: $(x = \text{two multiplied by two plus three}, y = (\text{plus } (\text{mult } 2 \ 2) \ 3))$



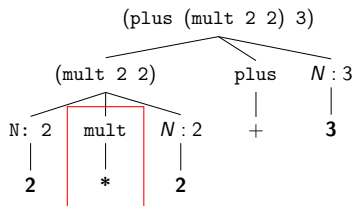
r1: 'two' →



Transformation and Rewrite Rules

- Decompose translation into a set of *local* transformations.

data: $(x = \text{two multiplied by two plus three}, y = (\text{plus (mult 2 2) 3}))$

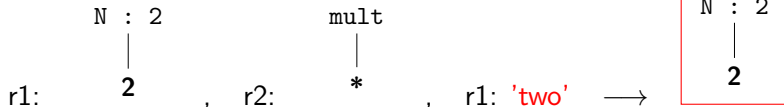
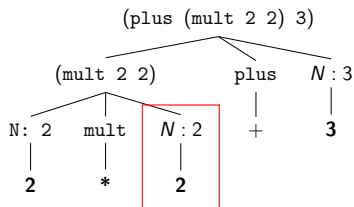


r1: $\begin{array}{c} N : 2 \\ | \\ 2 \end{array}$, r2: 'multiplied by' \longrightarrow $\begin{array}{c} \text{mult} \\ | \\ * \end{array}$

Transformation and Rewrite Rules

- Decompose translation into a set of *local* transformations.

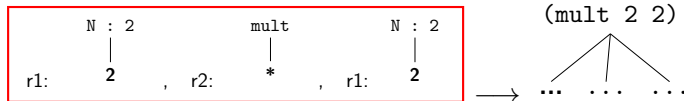
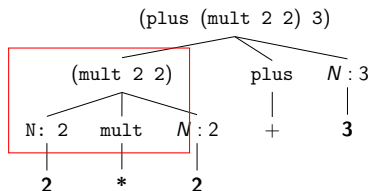
data: $(x = \text{two multiplied by two plus three}, y = (\text{plus } (\text{mult } 2 \ 2) \ 3))$



Transformation and Rewrite Rules

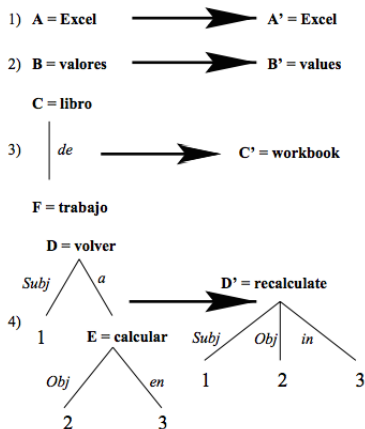
- Decompose translation into a set of *local* transformations.

data: $(x = \text{two multiplied by two plus three}, y = (\text{plus } (\text{mult } 2 \ 2) \ 3))$



Not a new idea: Early machine translation

($X = \text{Excel vuelve a calcular valores en libro de trabajo}$, $Y = \text{Excel recalculates values in workbook}$)



?

Rule-based Semantic Interpretation

Idea: Locally rewrite syntactic structure to semantic representations.

a. Jane did not hop.

b.
$$\left[\begin{array}{ll} \text{PRED} & \text{'hop<SUBJ>'} \\ \text{SUBJ} & \left[\text{PRED} \text{'Jane'} \right] \\ \text{ADJUNCT} & \left\{ \left[\text{PRED} \text{'not'} \right] \right. \\ & \left. \left[\text{ADJUNCT-TYPE} \text{ neg} \right] \right\} \end{array} \right]$$

c. *context_head*(*t*,not:10)

context_head(*ctx*(*hop*:17),*hop*:17)

in_context(*t*,role(mod(degree),*ctx*(*hop*:17),not:10,normal))

in_context(*ctx*(*hop*:17),role(Theme,*hop*:17,*Jane*:1))

word(*Jane*:1,*Jane*,noun,0,1,*t*,[[9482706]])

word(*hop*:17,*hop*,verb,0,17,*ctx*(*hop*:17),[[1948772], [2076532], [1823521], [2076385],
[2076247], [2076113]])

word(not:10,not,adv,0,10,*t*,[[24548]])

?

Learning Transformation Rules: ?

- ▶ Learn string \rightarrow tree transformation rules from $(text, MR)$ pairs
- ▶ **Components**

Learning Transformation Rules: ?

- ▶ Learn string \rightarrow tree transformation rules from $(text, MR)$ pairs
- ▶ **Components**
 - ▶ Text-meaning pairs (*robocup* and *geoquery*)

Learning Transformation Rules: ?

- ▶ Learn string \rightarrow tree transformation rules from $(text, MR)$ pairs
- ▶ **Components**
 - ▶ Text-meaning pairs (*robocup* and *geoquery*)
 - ▶ MR grammar (*compositional model*)

Learning Transformation Rules: ?

- ▶ Learn string \rightarrow tree transformation rules from $(text, MR)$ pairs
- ▶ **Components**
 - ▶ Text-meaning pairs (*robocup* and *geoquery*)
 - ▶ MR grammar (*compositional model*)
 - ▶ **SILT rule induction algorithm**

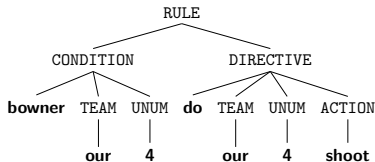
Learning Transformation Rules: ?

- ▶ Learn string \rightarrow tree transformation rules from $(text, MR)$ pairs
- ▶ **Components**
 - ▶ Text-meaning pairs (*robocup* and *geoquery*)
 - ▶ MR grammar (*compositional model*)
 - ▶ **SILT rule induction algorithm**
 - ▶ Greedy matching procedure

Datasets: Learning from MRs

► Robocup :

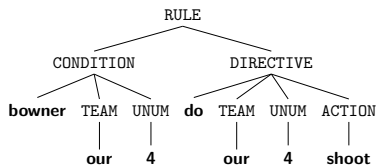
(If our p 4 has the ball, our p 4 should shoot, ((bowner our{4}) (do our{4} (shoot))))



MR Grammar

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bowner TEAM UNUM
DIRECTIVE	→	do TEAM UNUM ACTION
TEAM	→	our
UNUM	→	4
ACTION	→	shoot

Bottom-up, String \rightarrow Tree Rule Matching



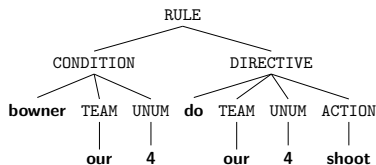
MR Grammar

RULE	\rightarrow	CONDITION DIRECTIVE
CONDITION	\rightarrow	bowner TEAM UNUM
DIRECTIVE	\rightarrow	do TEAM UNUM ACTION
TEAM	\rightarrow	our
UNUM	\rightarrow	4
ACTION	\rightarrow	shoot

Transformation: If our player 4 has the ball, our player 4 should shoot.

Input: If our player 4 has the ball, our player 4 should shoot.

Bottom-up, String \rightarrow Tree Rule Matching



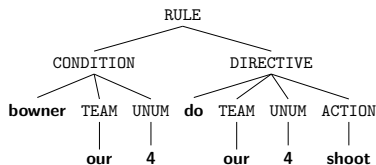
MR Grammar

RULE	\rightarrow	CONDITION DIRECTIVE
CONDITION	\rightarrow	bowner TEAM UNUM
DIRECTIVE	\rightarrow	do TEAM UNUM ACTION
TEAM	\rightarrow	our
UNUM	\rightarrow	4
ACTION	\rightarrow	shoot

Transformation: If **TEAM** player 4 has the ball, **TEAM** player 4 should shoot.

Input: If **our** player 4 has the ball, **our** player 4 should shoot.

Bottom-up, String → Tree Rule Matching



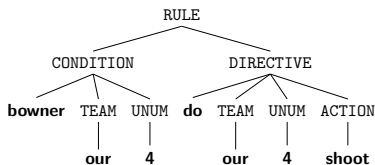
MR Grammar

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bowner TEAM UNUM
DIRECTIVE	→	do TEAM UNUM ACTION
TEAM	→	our
UNUM	→	4
ACTION	→	shoot

Transformation: If TEAM **UNUM** has the ball, TEAM **UNUM** should shoot.

Input: If our **player 4** has the ball, our **player 4** should shoot.

Bottom-up, String → Tree Rule Matching



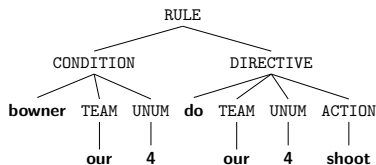
MR Grammar

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bowner TEAM UNUM
DIRECTIVE	→	do TEAM UNUM ACTION
TEAM	→	our
UNUM	→	4
ACTION	→	shoot

Transformation: If TEAM UNUM has the ball, TEAM UNUM should **ACTION**.

Input: If our player 4 has the ball, our player 4 should **shoot**.

Bottom-up, String → Tree Rule Matching



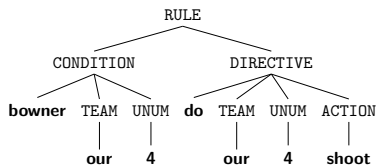
MR Grammar

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bowner TEAM UNUM
DIRECTIVE	→	do TEAM UNUM ACTION
TEAM	→	our
UNUM	→	4
ACTION	→	shoot

Transformation: If COND=[TEAM UNUM has ball], TEAM UNUM should ACTION.

Input: If our player 4 has the ball, our player 4 should shoot.

Bottom-up, String → Tree Rule Matching



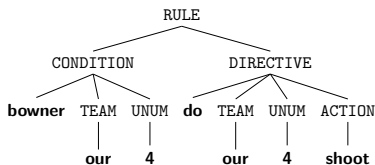
MR Grammar

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bowner TEAM UNUM
DIRECTIVE	→	do TEAM UNUM ACTION
TEAM	→	our
UNUM	→	4
ACTION	→	shoot

Transformation: If **CONDITION**, TEAM UNUM should ACTION.

Input: If **our player 4 has the ball**, our player 4 should shoot.

Bottom-up, String → Tree Rule Matching



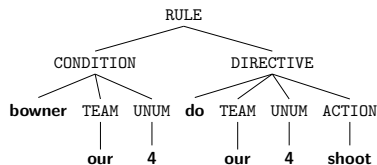
MR Grammar

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bowner TEAM UNUM
DIRECTIVE	→	do TEAM UNUM ACTION
TEAM	→	our
UNUM	→	4
ACTION	→	shoot

Transformation: If CONDITION, **DIR=[TEAM UNUM should ACTION]**.

Input: If our player 4 has the ball, **our player 4 should shoot.**

Bottom-up, String → Tree Rule Matching



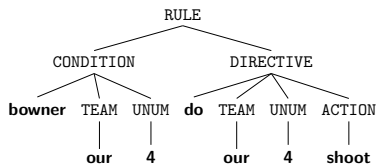
MR Grammar

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bowner TEAM UNUM
DIRECTIVE	→	do TEAM UNUM ACTION
TEAM	→	our
UNUM	→	4
ACTION	→	shoot

Transformation: If CONDITION, **DIRECTIVE**.

Input: If our player 4 has the ball, **our player 4 should shoot.**

Bottom-up, String \rightarrow Tree Rule Matching



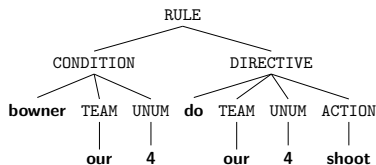
MR Grammar

RULE	\rightarrow	CONDITION DIRECTIVE
CONDITION	\rightarrow	bowner TEAM UNUM
DIRECTIVE	\rightarrow	do TEAM UNUM ACTION
TEAM	\rightarrow	our
UNUM	\rightarrow	4
ACTION	\rightarrow	shoot

Transformation: **RULE=[If CONDITION DIRECTIVE].**

Input: **If our player 4 has the ball, our player 4 should shoot.**

Bottom-up, String \rightarrow Tree Rule Matching



MR Grammar

RULE	\rightarrow	CONDITION DIRECTIVE
CONDITION	\rightarrow	bowner TEAM UNUM
DIRECTIVE	\rightarrow	do TEAM UNUM ACTION
TEAM	\rightarrow	our
UNUM	\rightarrow	4
ACTION	\rightarrow	shoot

Transformation: **RULE**

Input: **If our player 4 has the ball, our player 4 should shoot.**

SILT: Semantic Interpretation by Learning Transformations

- ▶ Extract mapping rules from strings to production rules.
- ▶ Bottom-up, find generalization between *positive* examples.
- ▶ Rank by **goodness** of fit over all examples.

SILT: General Algorithm

- 1: **Function** SILT ($T = \{(x_1, y_1), \dots, (x_n, y_n)\}, G_{MR}$)
- 2: **Set** $\Pi = T$ with parsed representations using G_{MR}
- 3: **Set** $\mathcal{P}_\Pi =$ productions with positive examples in T
- 3: **Set** $\mathcal{N}_\Pi =$ productions with negative examples in T
- 4: **Set** $L = \{\}$
- 5: **While** no more good rules **do**
- 6: $R^* = \text{FINDBESTRULES}(G_{MR}, \mathcal{P}_\Pi, \mathcal{N}_\Pi)$
- 7: $L = L \cup R^*$
- 8: apply rules in L to sentences in T
- 9: **Return** L

SILT: General Algorithm

- 1: **Function** SILT ($T = \{(x_1, y_1), \dots, (x_n, y_n)\}, G_{MR}$)
- 2: **Set** $\Pi = T$ with parsed representations using G_{MR}
- 3: **Set** $\mathcal{P}_\Pi =$ productions with positive examples in T
- 3: **Set** $\mathcal{N}_\Pi =$ productions with negative examples in T
- 4: **Set** $L = \{\}$
- 5: **While** no more good rules **do**
- 6: $R^* = \text{FINDBESTRULES}(G_{MR}, \mathcal{P}_\Pi, \mathcal{N}_\Pi)$
- 7: $L = L \cup R^*$
- 8: apply rules in L to sentences in T
- 9: **Return** L

$T = \{(x_1 = \text{If our p 4 has the ball our p4 should shoot, } y_2 = ((\text{bowner our}\{4\}) (\text{do our}\{4\} (\text{shoot}))),$
 $(x_2 = \text{when p 4 has possession p4 must pass, } y_2 = ((\text{bowner our}\{4\}) (\text{do our}\{4\} (\text{pass}))), \dots\}$

SILT: General Algorithm

- 1: **Function** SILT ($T = \{(x_1, y_1), \dots, (x_n, y_n)\}, G_{MR}$)
- 2: **Set** $\Pi = T$ with parsed representations using G_{MR}
- 3: **Set** $\mathcal{P}_\Pi =$ productions with positive examples in T
- 3: **Set** $\mathcal{N}_\Pi =$ productions with negative examples in T
- 4: **Set** $L = \{\}$
- 5: **While** no more good rules **do**
- 6: $R^* = \text{FINDBESTRULES}(G_{MR}, \mathcal{P}_\Pi, \mathcal{N}_\Pi)$
- 7: $L = L \cup R^*$
- 8: apply rules in L to sentences in T
- 9: **Return** L

$T = \{(x_1 = \text{If our p 4 has the ball our p4 should shoot}, y_2 = ((\text{bowner our}\{4\}) (\text{do our}\{4\} (\text{shoot}))),$
 $(x_2 = \text{when p 4 has possession p4 must pass}, y_2 = ((\text{bowner our}\{4\}) (\text{do our}\{4\} (\text{pass}))), \dots\}$

$\mathcal{P}_\Pi[\text{ACTION} \rightarrow \text{shoot}] = \{x_1, \dots\}$

$\mathcal{N}_\Pi[\text{ACTION} \rightarrow \text{shoot}] = \{x_2, \dots\}$

$\mathcal{P}_\Pi[\text{ACTION} \rightarrow \text{pass}] = \{x_2, \dots\}$

$\mathcal{N}_\Pi[\text{ACTION} \rightarrow \text{pass}] = \{x_1, \dots\}$

$\mathcal{P}_\Pi[\text{UNUM} \rightarrow \mathbf{4}] = \{x_1, x_2, \dots\}$

SILT: Find Best Rules

```
1: Function FINDBESTRULES ( $G_{MR}, \mathcal{P}_{\Pi}, \mathcal{N}_{\Pi}$ )
2:   Set  $R = \{\}$ 
3:   Foreach  $\pi \in G_{MR}$  do
4:     Set  $R_{\pi}$  to be maximally-specific rules from  $\mathcal{P}_{\Pi}$ 
5:     Repeat for  $k = 1000$ 
6:       Choose  $r_1, r_2$  at random
7:        $g = \text{GENERALIZE}(r_1, r_2, \pi)$ 
8:       Add  $g$  to  $R_{\pi}$ 
9:      $R = R \cup R_{\pi}$ 
10:   $r^* = \text{ARG MAX}_{r \in R} \text{GOODNESS}(r)$ 
11:  Remove positive examples covered by  $r^*$  from  $\mathcal{P}_{\Pi}$ 
12:  Return  $r^*$ 
```

SILT: Find Best Rules

```
1: Function FINDBESTRULES ( $G_{MR}, \mathcal{P}_{\Pi}, \mathcal{N}_{\Pi}$ )
2:   Set  $R = \{\}$ 
3:   Foreach  $\pi \in G_{MR}$  do
4:     Set  $R_{\pi}$  to be maximally-specific rules from  $\mathcal{P}_{\Pi}$ 
5:     Repeat for  $k = 1000$ 
6:       Choose  $r_1, r_2$  at random
7:        $g = \text{GENERALIZE}(r_1, r_2, \pi)$ 
8:       Add  $g$  to  $R_{\pi}$ 
9:      $R = R \cup R_{\pi}$ 
10:     $r^* = \text{ARG MAX}_{r \in R} \text{GOODNESS}(r)$ 
11:    Remove positive examples covered by  $r^*$  from  $\mathcal{P}_{\Pi}$ 
12:  Return  $r^*$ 
```

Example

line 3: $\pi = \text{UNUM} \rightarrow 4, \mathcal{P}_{\Pi}[\text{UNUM} \rightarrow 4] = \{x_1, x_2\}$

SILT: Find Best Rules

```
1: Function FINDBESTRULES ( $G_{MR}, \mathcal{P}_{\Pi}, \mathcal{N}_{\Pi}$ )
2:   Set  $R = \{\}$ 
3:   Foreach  $\pi \in G_{MR}$  do
4:     Set  $R_{\pi}$  to be maximally-specific rules from  $\mathcal{P}_{\Pi}$ 
5:     Repeat for  $k = 1000$ 
6:       Choose  $r_1, r_2$  at random
7:        $g = \text{GENERALIZE}(r_1, r_2, \pi)$ 
8:       Add  $g$  to  $R_{\pi}$ 
9:      $R = R \cup R_{\pi}$ 
10:     $r^* = \text{ARG MAX}_{r \in R} \text{GOODNESS}(r)$ 
11:    Remove positive examples covered by  $r^*$  from  $\mathcal{P}_{\Pi}$ 
12:  Return  $r^*$ 
```

Example

line 3: $\pi = \text{UNUM} \rightarrow 4$, $\mathcal{P}_{\Pi}[\text{UNUM} \rightarrow 4] = \{x_1, x_2\}$
 $R_{\pi} = \{x_1 \Rightarrow \text{UNUM} \rightarrow 4, x_2 \Rightarrow \text{UNUM} \rightarrow 4, \dots\}$

SILT: Find Best Rules

```
1: Function FINDBESTRULES ( $G_{MR}, \mathcal{P}_{\Pi}, \mathcal{N}_{\Pi}$ )
2:   Set  $R = \{\}$ 
3:   Foreach  $\pi \in G_{MR}$  do
4:     Set  $R_{\pi}$  to be maximally-specific rules from  $\mathcal{P}_{\Pi}$ 
5:     Repeat for  $k = 1000$ 
6:       Choose  $r_1, r_2$  at random
7:        $g = \text{GENERALIZE}(r_1, r_2, \pi)$ 
8:       Add  $g$  to  $R_{\pi}$ 
9:      $R = R \cup R_{\pi}$ 
10:     $r^* = \text{ARG MAX}_{r \in R} \text{GOODNESS}(r)$ 
11:    Remove positive examples covered by  $r^*$  from  $\mathcal{P}_{\Pi}$ 
12:  Return  $r^*$ 
```

Example

line 3: $\pi = \text{UNUM} \rightarrow 4, \mathcal{P}_{\Pi}[\text{UNUM} \rightarrow 4] = \{x_1, x_2\}$

$R_{\pi} = \{x_1 \Rightarrow \text{UNUM} \rightarrow 4, x_2 \Rightarrow \text{UNUM} \rightarrow 4, \dots\}$

line 6: $r_1 = (x_1 \Rightarrow \text{UNUM} \rightarrow 4), r_2 = (x_2 \Rightarrow \text{UNUM} \rightarrow 4)$

SILT: Find Best Rules

```
1: Function FINDBESTRULES ( $G_{MR}, \mathcal{P}_{\Pi}, \mathcal{N}_{\Pi}$ )
2:   Set  $R = \{\}$ 
3:   Foreach  $\pi \in G_{MR}$  do
4:     Set  $R_{\pi}$  to be maximally-specific rules from  $\mathcal{P}_{\Pi}$ 
5:     Repeat for  $k = 1000$ 
6:       Choose  $r_1, r_2$  at random
7:        $g = \text{GENERALIZE}(r_1, r_2, \pi)$ 
8:       Add  $g$  to  $R_{\pi}$ 
9:      $R = R \cup R_{\pi}$ 
10:     $r^* = \text{ARG MAX}_{r \in R} \text{GOODNESS}(r)$ 
11:    Remove positive examples covered by  $r^*$  from  $\mathcal{P}_{\Pi}$ 
12:  Return  $r^*$ 
```

Example

line 3: $\pi = \text{UNUM} \rightarrow 4$, $\mathcal{P}_{\Pi}[\text{UNUM} \rightarrow 4] = \{x_1, x_2\}$

$R_{\pi} = \{x_1 \Rightarrow \text{UNUM} \rightarrow 4, x_2 \Rightarrow \text{UNUM} \rightarrow 4, \dots\}$

line 6: $r_1 = (x_1 \Rightarrow \text{UNUM} \rightarrow 4)$, $r_2 = (x_2 \Rightarrow \text{UNUM} \rightarrow 4)$

line 7: $g = (x_1 \cap x_2) \Rightarrow \text{UNUM} \rightarrow 4$

SILT: Find Best Rules

```
1: Function FINDBESTRULES ( $G_{MR}, \mathcal{P}_{\Pi}, \mathcal{N}_{\Pi}$ )
2:   Set  $R = \{\}$ 
3:   Foreach  $\pi \in G_{MR}$  do
4:     Set  $R_{\pi}$  to be maximally-specific rules from  $\mathcal{P}_{\Pi}$ 
5:     Repeat for  $k = 1000$ 
6:       Choose  $r_1, r_2$  at random
7:        $g = \text{GENERALIZE}(r_1, r_2, \pi)$ 
8:       Add  $g$  to  $R_{\pi}$ 
9:      $R = R \cup R_{\pi}$ 
10:     $r^* = \text{ARG MAX}_{r \in R} \text{GOODNESS}(r)$ 
11:    Remove positive examples covered by  $r^*$  from  $\mathcal{P}_{\Pi}$ 
12:  Return  $r^*$ 
```

Example

line 3: $\pi = \text{UNUM} \rightarrow 4, \mathcal{P}_{\Pi}[\text{UNUM} \rightarrow 4] = \{x_1, x_2\}$

$R_{\pi} = \{x_1 \Rightarrow \text{UNUM} \rightarrow 4, x_2 \Rightarrow \text{UNUM} \rightarrow 4, \dots\}$

line 6: $r_1 = (x_1 \Rightarrow \text{UNUM} \rightarrow 4), r_2 = (x_2 \Rightarrow \text{UNUM} \rightarrow 4)$

line 7: $g = (x_1 \cap x_2) \Rightarrow \text{UNUM} \rightarrow 4$

if our p 4 has the ball ... should shoot \cap when our p 4 ... should pass $=$

SILT: Find Best Rules

```
1: Function FINDBESTRULES ( $G_{MR}, \mathcal{P}_{\Pi}, \mathcal{N}_{\Pi}$ )
2:   Set  $R = \{\}$ 
3:   Foreach  $\pi \in G_{MR}$  do
4:     Set  $R_{\pi}$  to be maximally-specific rules from  $\mathcal{P}_{\Pi}$ 
5:     Repeat for  $k = 1000$ 
6:       Choose  $r_1, r_2$  at random
7:        $g = \text{GENERALIZE}(r_1, r_2, \pi)$ 
8:       Add  $g$  to  $R_{\pi}$ 
9:      $R = R \cup R_{\pi}$ 
10:     $r^* = \text{ARG MAX}_{r \in R} \text{GOODNESS}(r)$ 
11:    Remove positive examples covered by  $r^*$  from  $\mathcal{P}_{\Pi}$ 
12:  Return  $r^*$ 
```

Example

line 3: $\pi = \text{UNUM} \rightarrow 4, \mathcal{P}_{\Pi}[\text{UNUM} \rightarrow 4] = \{x_1, x_2\}$

$R_{\pi} = \{x_1 \Rightarrow \text{UNUM} \rightarrow 4, x_2 \Rightarrow \text{UNUM} \rightarrow 4, \dots\}$

line 6: $r_1 = (x_1 \Rightarrow \text{UNUM} \rightarrow 4), r_2 = (x_2 \Rightarrow \text{UNUM} \rightarrow 4)$

line 7: $g = (x_1 \cap x_2) \Rightarrow \text{UNUM} \rightarrow 4$

if our p 4 has the ball ... should shoot \cap when our p 4 ... should pass $=$

our p 4 $\Rightarrow \text{UNUM} \rightarrow 4$

SILT: Full Algorithm

```
1: Function SILT ( $T = \{(x_1, y_1), \dots, (x_n, y_n)\}, G_{MR}$ )
2:   Set  $\Pi = T$  with parsed representations using  $G_{MR}$ 
3:   Set  $\mathcal{P}_\Pi =$  productions with positive examples in  $T$ 
3:   Set  $\mathcal{N}_\Pi =$  productions with negative examples in  $T$ 
4:   Set  $L = \{\}$ 
5:   While no more good rules do
6:      $R^* = \text{FINDBESTRULES}(G_{MR}, \mathcal{P}_\Pi, \mathcal{N}_\Pi)$ 
7:      $L = L \cup R^*$ 
8:     apply rules in  $L$  to sentences in  $T$ 
9:   Return  $L$ 
```

```
1: Function FINDBESTRULES ( $G_{MR}, \mathcal{P}_\Pi, \mathcal{N}_\Pi$ )
2:   Set  $R = \{\}$ 
3:   Foreach  $\pi \in G_{MR}$  do
4:     Set  $R_\pi$  to be maximally-specific rules from  $\mathcal{P}_\Pi$ 
5:     Repeat for  $k = 1000$ 
6:       Choose  $r_1, r_2$  at random
7:        $g = \text{GENERALIZE}(r_1, r_2, \pi)$ 
8:       Add  $g$  to  $R_\pi$ 
9:      $R = R \cup R_\pi$ 
10:     $r^* = \text{ARG MAX}_{r \in R} \text{GOODNESS}(r)$ 
11:    Remove positive examples covered by  $r^*$  from  $\mathcal{P}_\Pi$ 
12:  Return  $r^*$ 
```


Experiment results

- ▶ **Main metric:** Does my parser generate the gold representations?
- ▶ **Cross-validation:** Test on multiple test sets (variation of the standard train-test setup)
 - ▶ **Motivation (in this case):** Small datasets. A single test set might not be a good representative sample.

Experiment results

- ▶ **Main metric:** Does my parser generate the gold representations?
- ▶ **Cross-validation:** Test on multiple test sets (variation of the standard train-test setup)
 - ▶ **Motivation (in this case):** Small datasets. A single test set might not be a good representative sample.
- ▶ **Recall:** How many inputs received a full analysis/parse?
Precision: out of those, how many were correct?

Results

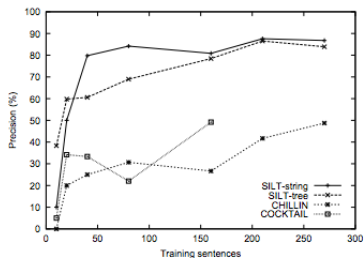


Figure 7: Precision learning curves for CLANG

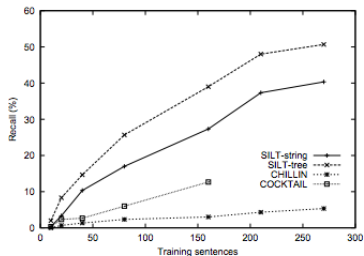


Figure 8: Recall learning curves for CLANG

Why does this work?

- ▶ **goodness**: potential rules compete in terms of their global coverage or *goodness* (bottom-up, or simple to complex)

$$\text{GOODNESS}(r) = \frac{\text{pos}(r)^2}{\text{pos}(r) + \text{neg}(r)}$$

- ▶ **greedy**: *good* rules are applied greedily, MR grammar is deterministic.

Why does this work?

- ▶ **goodness**: potential rules compete in terms of their global coverage or *goodness* (bottom-up, or simple to complex)

$$\text{GOODNESS}(r) = \frac{\text{pos}(r)^2}{\text{pos}(r) + \text{neg}(r)}$$

- ▶ **greedy**: *good* rules are applied greedily, MR grammar is deterministic.
 - ▶ There is no turning back. Errors propagate.
 - ▶ No way to handle or preserve ambiguity.

A non-greedy semantic parser?

Idea: The best local option might not (in general) be the best choice. Try to consider all possibilities.

'at the REGION' \Rightarrow CONDITION \rightarrow ((bpos REGION))

'at the REGION' \Rightarrow CONDITION \rightarrow ((ppos REGION))

If the player is [at the REGION], the goalie should guard the goal.

A non-greedy semantic parser?

Idea: The best local option might not (in general) be the best choice. Try to consider all possibilities.

'at the REGION' \Rightarrow **CONDITION** \rightarrow ((bpos REGION))

'at the REGION' \Rightarrow **CONDITION** \rightarrow ((ppos REGION))

If the **player** is [at the REGION], the goalie should guard the goal.

A non-greedy semantic parser?

Idea: The best local option might not (in general) be the best choice. Try to consider all possibilities.

'at the REGION' \Rightarrow **CONDITION** \rightarrow ((bpos REGION))

'at the REGION' \Rightarrow **CONDITION** \rightarrow ((ppos REGION))

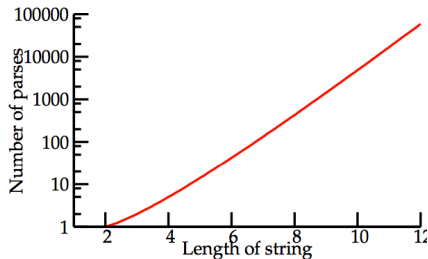
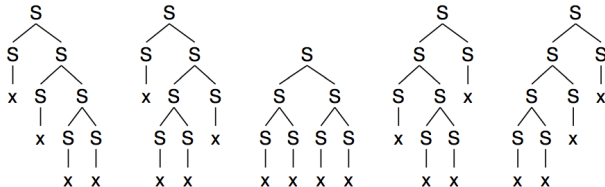
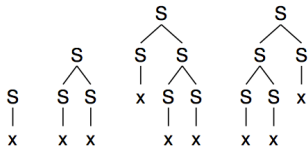
'player at the REGION' \Rightarrow **CONDITION** \rightarrow ((ppos REGION))

If the [**player** is **at the REGION**], the goalie should guard the goal.

Considering all possibilities?

- Ambiguities can grow exponentially with length

$$R = \{S \rightarrow S S, S \rightarrow x\}$$



CKY Parsing Algorithm

- ▶ Polynomial-time, bottom-up parsing algorithm for CFGs, uses *dynamic programming*.
- ▶ **dynamic programming:** breaks problems into smaller sub-problems, local solutions are shared.
- ▶ The basis of virtually all decoding methods we will cover in this class (from MT, parsing, .., **very important!**).

Brief review: Context-Free Grammars

- ▶ **context-free grammar (CFG):**

$$\mathcal{G} = (\Sigma, N, S, R)$$

- ▶ N : set of terminal symbols.
 - ▶ Σ : set of non-terminal symbols.
 - ▶ R : set of rules = $\{N \rightarrow \alpha \mid \alpha \in (N \cup \Sigma)^*\}$
 - ▶ S : start symbol
-
- ▶ **Chomsky Normal-Form (CNF):** rules in R take the form:
 - ▶ $A \rightarrow B C$, where $B, C \in N$
 - ▶ $A \rightarrow a$, where $a \in \Sigma$

CKY: Chart Filling

RULE → CONDITION DIRECTIVE
CONDITION → **bpos** REGION | **ppos** REGION
DIRECTIVE → PLAYER ACTION
UNUM → **4**
ACTION → **shoot** | **pass**
ppos → PLAYER

Transformation rules

PLAYER → 'p4' | 'then' | 'pass' | 'shoots'
REGION → 'at the goal' | 'when'
shoot → 'p4' | 'shoots' | if
pass → 'pass' | 'then should'
 λ → 'if' | 'is' | 'then' | 'when'

when p4 is-at-the-goal p4 shoots.

	1	2	3	4	5
0					
1					
2					
3					
4					

CKY: Chart Filling

RULE \longrightarrow CONDITION DIRECTIVE
 CONDITION \longrightarrow **bpos** REGION | **ppos** REGION
 DIRECTIVE \longrightarrow PLAYER ACTION
 UNUM \longrightarrow **4**
 ACTION \longrightarrow **shoot** | **pass**
 ppos \longrightarrow PLAYER

Transformation rules

PLAYER \longrightarrow 'p4' | 'then' | 'pass' | 'shoots'
REGION \longrightarrow 'at the goal' | 'when'
 shoot \longrightarrow 'p4' | 'shoots' | if
 pass \longrightarrow 'pass' | 'then should'
 λ \longrightarrow 'if' | 'is' | 'then' | 'when'

₀**when**₁ p4 is-at-the-goal p4 shoots.

	1	2	3	4	5
0	λ , REG.				
1					
2					
3					
4					

CKY: Chart Filling

RULE \longrightarrow CONDITION DIRECTIVE
 CONDITION \longrightarrow **bpos** REGION | **ppos** REGION
 DIRECTIVE \longrightarrow PLAYER ACTION
 UNUM \longrightarrow **4**
 ACTION \longrightarrow **shoot** | **pass**
 ppos \longrightarrow PLAYER

Transformation rules

PLAYER \longrightarrow 'p4' | 'then' | 'pass'
 REGION \longrightarrow 'at the goal' | 'when'
shoot \longrightarrow 'p4' | 'shoots' | if
 pass \longrightarrow 'pass' | 'then should'
 λ \longrightarrow 'if' | 'is' | 'then' | 'when'

when $_1p_4_2$ is-at-the-goal p4 shoots.

	1	2	3	4	5
0	$\lambda, \text{REG.}$				
1		sh., PL.			
2					
3					
4					

CKY: Chart Filling

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bpos REGION ppos REGION
DIRECTIVE	→	PLAYER ACTION
UNUM	→	4
ACTION	→	shoot pass
ppos	→	PLAYER

Transformation rules

PLAYER	→	'p4' 'then' 'pass' 'shoots'
REGION	→	'at the goal' 'when'
shoot	→	'p4' 'shoots' if
pass	→	'pass' 'then should'
λ	→	'if' 'is' 'then' 'when'

when p4 ₂**is-at-the-goal**₃ p4 shoots.

	1	2	3	4	5
0	λ , REG.				
1		sh., PL.			
2			REG.		
3					
4					

CKY: Chart Filling

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bpos REGION ppos REGION
DIRECTIVE	→	PLAYER ACTION
UNUM	→	4
ACTION	→	shoot pass
ppos	→	PLAYER

Transformation rules

PLAYER	→	'p4' 'then' 'pass' 'shoots'
REGION	→	'at the goal' 'when'
shoot	→	'p4' 'shoots' if
pass	→	'pass' 'then should'
λ	→	'if' 'is' 'then' 'when'

when p4 is-at-the-goal p4 shoots.

	1	2	3	4	5
0	λ ,REG.				
1		sh.,PL.			
2			REG.		
3				sh.,PL.	
4					sh.,PL.

CKY: Chart Filling

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bpos REGION PLAYER REGION
DIRECTIVE	→	PLAYER ACTION
UNUM	→	4
ACTION	→	shoot pass
ppos	→	PLAYER

Transformation rules

PLAYER	→	'p4' 'then' 'pass' 'shoots'
REGION	→	'at the goal' 'when'
shoot	→	'p4' 'shoots' if
pass	→	'pass' 'then should'
λ	→	'if' 'is' 'then' 'when'

₀when p4 is-at-the-goal₃ p4 shoots.

	1	2	3	4	5
0	λ ,REG.		COND.		
1		sh.,PL.			
2			REG.		
3				sh.,PL.	
4					sh.,PL.

CKY: Chart Filling

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bpos REGION PLAYER REGION
DIRECTIVE	→	PLAYER ACTION
UNUM	→	4
ACTION	→	shoot pass
ppos	→	PLAYER

Transformation rules

PLAYER	→	'p4' 'then' 'pass' 'shoots'
REGION	→	'at the goal' 'when'
shoot	→	'p4' 'shoots' if
pass	→	'pass' 'then should'
λ	→	'if' 'is' 'then' 'when'

when p4 is-at-the-goal₃ **p4 shoots**₅.

	1	2	3	4	5
0	λ , REG.		COND.		
1		sh., PL.			
2			REG.		
3				sh., PL.	DIR.
4					sh., PL.

CKY: Full Algorithm

```
1: Function CKY( $\mathcal{G}, x = a_1, a_2, \dots, a_n$ )
2:   Set  $\mathcal{T} = \emptyset$ 
3:   for all  $j$  from 1 up to  $n$  do
4:     for all rules  $A \rightarrow a_j$  in  $\mathcal{G}_R$  do
5:       add  $[j-1, A, j]$  to  $\mathcal{T}$ 
6:     for all  $i$  from  $j-2$  down to 0 do
7:       for all  $k$  from  $i+1$  up to  $j-1$  do
8:         for all rules  $A \rightarrow BC$  in  $\mathcal{G}_R$  do
9:           if rules  $[i, B, k]$  and  $[k, C, j]$  are in  $\mathcal{T}$  then
10:            add  $[i, A, j]$  to  $\mathcal{T}$ 
11:   if  $[0, \mathcal{G}_S, n]$  is in  $\mathcal{T}$  then
12:     return true
```

- ▶ **lines 3-10:** chart filling routine.
- ▶ **lines 11-12:** recognition (is this a string in my language?)
- ▶ **key:** every smaller span is explored before each larger span.

Non-greedy parsing

- ▶ **goodness:** Is this rule used in valid derivations?
- ▶ **Example:** 'when' \Rightarrow REGION

when p4 is-at-the-goal p4 shoots.

	1	2	3	4	5
0	λ ,REG.		COND.		RULE
1		sh.,PL.			
2			REG.		
3				sh.,PL.	DIR.
4					sh.,PL.

Non-greedy parsing

- ▶ **goodness:** Is this rule used in a valid derivation?
- ▶ **probabilistic:** quantify this in terms of probabilities

Plan for lecture

- ▶ Introduce probabilistic grammars as an alternative tool to solve these problems.
- ▶ **Setting:** rather than extract rules in a greedy fashion, we extract many rules then learn the *good* rules.

Probabilistic Context-Free Grammars

► probabilistic context-free grammar (PCFG):

$$\mathcal{G}_\theta = (\Sigma, N, S, R, \theta)$$

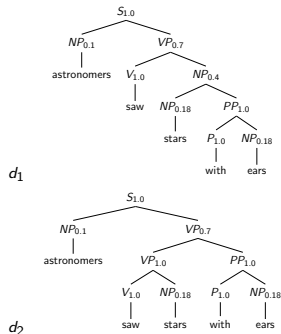
- N : set of terminal symbols.
- Σ : set of non-terminal symbols.
- R : set of rules = $\{N \rightarrow \alpha \mid \alpha \in (N \cup \Sigma)^*\}$
- S : start symbol
- θ : parameters : $\theta_{N \rightarrow \alpha} \rightarrow [0, 1)$ such that

$$\forall R_N \quad \sum_{(N \rightarrow \alpha) \in R_N} \theta_{N \rightarrow \alpha} = 1.0$$

(prob. distribution over lhs rules)

PCFG Basics

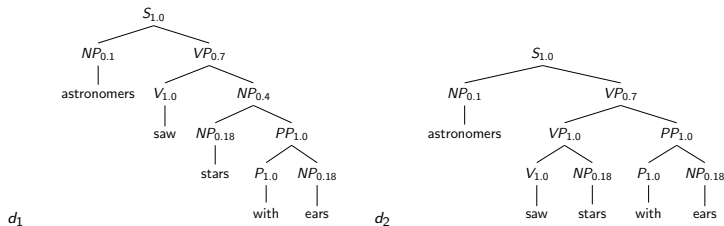
S	→	NP VP (1.0)
PP	→	P NP (1.0)
VP	→	V NP (0.7) VP PP (0.3)
P	→	with (1.0)
V	→	saw (1.0)
NP	→	NP PP (0.4)
NP	→	astronomers (0.1)
NP	→	ears (0.18)
NP	→	saw (0.04)
NP	→	stars (0.18))
NP	→	telescopes (0.1)



► probability of a derivation:

$$p_{\theta}(d) = \prod_{(N \rightarrow \alpha) \text{ in } d} \theta_{N \rightarrow \alpha}$$

PCFG Basics

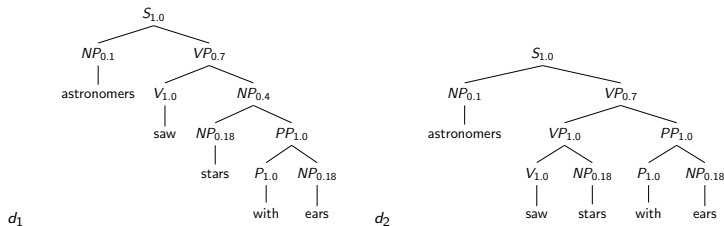


► probability of a derivation:

$$p_{\theta}(d) = \prod_{(N \rightarrow \alpha) \text{ in } d} \theta_{N \rightarrow \alpha}$$

- $p_{\theta}(d_1) = 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 * 1.0 * 1.0 * 0.18 = 0.0009072$
- $p_{\theta}(d_2) = 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 * 1.0 * 1.0 * 0.18 = 0.0015876$

PCFG Basics



► probability of a derivation:

$$p_{\theta}(d) = \prod_{(N \rightarrow \alpha) \text{ in } d} \theta_{N \rightarrow \alpha}$$

► $p_{\theta}(d_1) = 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 * 1.0 * 1.0 * 0.18 = 0.0009072$

► $p_{\theta}(d_2) = 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 * 1.0 * 1.0 * 0.18 = 0.0015876$

► probability of a sentence: \times

$$p_{G_{\theta}}(x = w_1, w_2, \dots, w_n) = \sum_d p_{\theta}(d)$$

Efficient computation of probabilities

- **Probability of a sentence:**

$$p_{\mathcal{G}_\theta}(x = w_1, w_2, \dots, w_n) = \sum_d p_\theta(d)$$

Efficient computation of probabilities

- **Probability of a sentence:**

$$p_{\mathcal{G}_\theta}(x = w_1, w_2, \dots, w_n) = \sum_d p_\theta(d)$$

- requires find all parse derivations d (same problem as before)

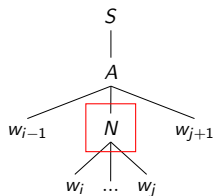
Efficient computation of probabilities

- **Probability of a sentence:**

$$p_{\mathcal{G}_\theta}(x = w_1, w_2, \dots, w_n) = \sum_d p_\theta(d)$$

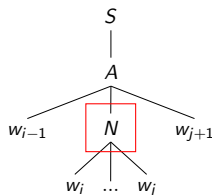
- requires find all parse derivations d (same problem as before)
- We can use dynamic programming again to solve this.

Inside Probabilities



► **definition:** $\beta_{i,j}(N) = P(w_i, \dots, w_j \mid N, \mathcal{G}_\theta)$

Inside Probabilities



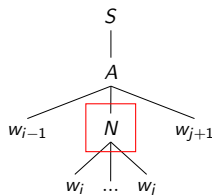
► **definition:** $\beta_{i,j}(N) = P(w_i, \dots, w_j \mid N, \mathcal{G}_\theta)$

► **computed recursively and bottom-up:**

► **Base:**

$$\beta_{i,i+1}(N) = \theta_{N \rightarrow w_{i,i+1}}$$

Inside Probabilities



► **definition:** $\beta_{i,j}(N) = P(w_i, \dots, w_j \mid N, \mathcal{G}_\theta)$

► **computed recursively and bottom-up:**

► **Base:**

$$\beta_{i,i+1}(N) = \theta_{N \rightarrow w_{i,i+1}}$$

► **Else:**

$$\beta_{i,j}(N) = \sum_{B,C} \sum_{i <= k <= j} \theta_{N \rightarrow B \ C} \beta_{i,k}(B) \beta_{k+1,j}(C)$$

Inside Probabilities: Example

S \longrightarrow NP VP (1.0)
 PP \longrightarrow P NP (1.0)
 VP \longrightarrow V NP (0.7) | VP PP (0.3)
 P \longrightarrow **with** (1.0)
 V \longrightarrow **saw** (1.0)
 NP \longrightarrow NP PP (0.4)
 NP \longrightarrow **astronomers** (0.1)
 NP \longrightarrow **ears** (0.18)
 NP \longrightarrow **saw** (0.04)
 NP \longrightarrow **stars** (0.18))
 NP \longrightarrow **telescopes** (0.1)

fragment: ... ₆with₇ ears₈ ...

$$\beta_{6,7}(P) = 1.0$$

$$\beta_{7,8}(NP) = 0.18$$

$$\beta_{6,8}(PP) = 1.0 * 1.0 * 0.18$$

...

$$\beta_{0,n}(S) = \dots$$

► **definition:** $\beta_{i,j}(N) = P(w_i, \dots, w_j \mid N, \mathcal{G}_\theta)$

► **computed recursively and bottom-up:**

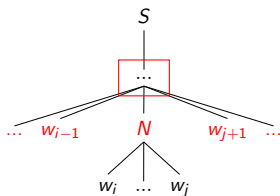
► **Base:**

$$\beta_{i,i+1}(N) = \theta_{N \rightarrow w_{i,i+1}}$$

► **Else:**

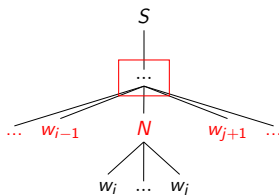
$$\beta_{i,j}(N) = \sum_{B,C} \sum_{i \leq k \leq j} \theta_{N \rightarrow B C} \beta_{i,k}(B) \beta_{k+1,j}(C)$$

Outside Probabilities



► **definition:** $\alpha_{i,j}(N) = P(w_0, \dots, w_{i-1}, N_{i,j}, w_{j+1}, \dots, w_n \mid \mathcal{G}_\theta)$

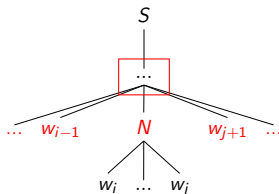
Outside Probabilities



- ▶ **definition:** $\alpha_{i,j}(N) = P(w_0, \dots, w_{i-1}, N_{i,j}, w_{j+1}, \dots, w_n \mid \mathcal{G}_\theta)$
- ▶ **computed recursively and top-down:**
 - ▶ **Base:**

$$\alpha_{0,n}(S) = 1$$

Outside Probabilities



► **definition:** $\alpha_{i,j}(N) = P(w_0, \dots, w_{i-1}, N_{i,j}, w_{j+1}, \dots, w_n \mid \mathcal{G}_\theta)$

► **computed recursively and top-down:**

► **Base:**

$$\alpha_{0,n}(S) = 1$$

► **Else:**

$$\alpha_{i,j}(N) = \sum_{B,C} \sum_{0 \leq k < i} \theta_{B \rightarrow C} \beta_{k,i-1}(C) \alpha_{k,j}(B) + \sum_{B,C} \sum_{n \geq k > j} \theta_{B \rightarrow C} \beta_{j+1,k}(C) \alpha_{i,k}(B)$$

Computing Probabilities: take-aways

- ▶ **Idea:** inside and outside probabilities allows us to efficiently compute probabilities (e.g. probability of a sentence)
- ▶ Can be integrated within a chart-filling procedure (e.g. the CKY algorithm).
- ▶ For a given sentence, we compute the probability of a rule N spanning from i to j :

$$P(w_0, \dots, w_n, N_{i,j} \mid \mathcal{G}_\theta) = \alpha_{i,j}(N)\beta_{i,j}(N)$$

Parameter Estimation and Learning

- ▶ **Objective (english):** We want to find parameters θ that maximize the probability of our training dataset $D (= x_1, \dots, x_n)$.

Parameter Estimation and Learning

- ▶ **Objective (english):** We want to find parameters θ that maximize the probability of our training dataset $D (= x_1, \dots, x_n)$.



$$\operatorname{argmax}_{\mathcal{G}_{\theta}} \prod_i^n p_{\mathcal{G}_{\theta}(x_i)}$$

Back to Semantic Parsing

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bpos REGION ppos REGION
DIRECTIVE	→	PLAYER ACTION
UNUM	→	4
ACTION	→	shoot pass
ppos	→	PLAYER

Transformation rules

PLAYER	→	'p4' 'then' 'pass' 'shoots'
REGION	→	'at the goal' 'when'
shoot	→	'p4' 'shoots' if
pass	→	'pass' 'then should'
λ	→	'if' 'is' 'then' 'when'

- ▶ We have a dataset of text and outputs.
- ▶ We have mapping rules that over-generate, want to find a grammar the tells us something about the **goodness** of rule.

Parameter Estimation and Learning

- ▶ **Objective (english):** We want to find parameters θ that maximize the probability of our training dataset $D (= x_1, \dots, x_n)$.



$$\operatorname{argmax}_{\mathcal{G}_{\theta}} \prod_i^n p_{\mathcal{G}_{\theta}}(x_i)$$

- ▶ **Maximum Likelihood Estimation (MLE)** (with full information)

$$\theta_{N \rightarrow \alpha} = \frac{\text{count}(N \rightarrow \alpha)}{\sum_{\alpha'} \text{count}(N \rightarrow \alpha')}$$

Parameter Estimation and Learning

- ▶ **Objective (english):** We want to find parameters θ that maximize the probability of our training dataset $D (= x_1, \dots, x_n)$.



$$\operatorname{argmax}_{\mathcal{G}_{\theta}} \prod_i^n p_{\mathcal{G}_{\theta}}(x_i)$$

- ▶ **Maximum Likelihood Estimation (MLE)** (with full information)

$$\theta_{N \rightarrow \alpha} = \frac{\text{count}(N \rightarrow \alpha)}{\sum_{\alpha'} \text{count}(N \rightarrow \alpha')}$$

- ▶ **problem:** we often don't have full information

Back to Semantic Parsing

RULE	→	CONDITION DIRECTIVE
CONDITION	→	bpos REGION ppos REGION
DIRECTIVE	→	PLAYER ACTION
UNUM	→	4
ACTION	→	shoot pass
ppos	→	PLAYER

Transformation rules

PLAYER	→	'p4' 'then' 'pass' 'shoots'
REGION	→	'at the goal' 'when'
shoot	→	'p4' 'shoots' if
pass	→	'pass' 'then should'
λ	→	'if' 'is' 'then' 'when'

- ▶ We have a dataset of text and outputs.
- ▶ We have mapping rules that over generate, want to find a grammar the tells us something about the **goodness** of rule.
- ▶ We don't have the actually target derivations for each x .

Expectation-Maximization (EM)

- ▶ Iterative technique for doing MLE in cases involving hidden (or latent) variables and incomplete data.
- ▶ Makes an initial (possibly random) guess about parameters, then iteratively repeats two steps:
 - ▶ **e-step:** Estimates counts using current model parameters.
 - ▶ **m-step:** Re-estimate parameters based on these completions.

Expectation-Maximization (EM)

- ▶ Iterative technique for doing MLE in cases involving hidden (or latent) variables and incomplete data.
- ▶ Makes an initial (possibly random) guess about parameters, then iteratively repeats two steps:
 - ▶ **e-step:** Estimates counts using current model parameters.
 - ▶ **m-step:** Re-estimate parameters based on these completions.
- ▶ **converge:** Will eventually converge (not proved here).

Expectation-Maximization (EM)

- ▶ Iterative technique for doing MLE in cases involving hidden (or latent) variables and incomplete data.
- ▶ Makes an initial (possibly random) guess about parameters, then iteratively repeats two steps:
 - ▶ **e-step:** Estimates counts using current model parameters.
 - ▶ **m-step:** Re-estimate parameters based on these completions.
- ▶ **converge:** Will eventually converge (not proved here).
- ▶ **inside-outside algorithm:** counts are based on inside-outside probabilities.

Inside-Outside Algorithm (rough outline)

```
1: Function INSIDE-OUTSIDE( $\mathcal{G}_\theta, D = x_1, \dots, x_n$ )
2:   Until converge do
2:     Set  $\mathcal{C} = \emptyset$ 
3:     for all sentences  $i$  from 1 up  $n$  do
4:       compute inside probabilities
5:       compute outside probabilities
6:       for all rules of the form  $N \rightarrow B \ C$  do
7:          $\mathcal{C}[N \rightarrow B \ C] += \frac{\theta_{N \rightarrow B \ C}}{p_{\mathcal{G}_\theta}(x_i)} \sum_{0 \leq i \leq j \leq k \leq n} \alpha_{i,k}(N) \beta_{i,k}(B) \beta_{j+1,k}(C)$ 
8:       for all rules of the form  $N \rightarrow w \ C$  do
9:          $\mathcal{C}[N \rightarrow w] += \frac{\theta_{N \rightarrow w \ C}}{p_{\mathcal{G}_\theta}(x_i)} \sum_{0 \leq n} \beta_{i,i+1}(N)$ 
10:      for all rules  $N \rightarrow \delta$  from 1 up  $n$  do
11:         $\theta_{N \rightarrow \delta} = \frac{\mathcal{C}[N \rightarrow \delta]}{\sum \delta' \mathcal{C}(N \rightarrow \delta')}$ 
12:   return  $\mathcal{G}_\theta$ 
```

► **e-step:** lines 3-9, **m-step:** lines 10-11.

Big Ideas:

- ▶ Greedy versus non-greedy parsing and extraction of rules
- ▶ **Learning:** two different notions of *goodness*
- ▶ Introduced the basics of (P)CFGs, and one particular parameter estimation method.

Big Ideas:

- ▶ Greedy versus non-greedy parsing and extraction of rules
- ▶ **Learning:** two different notions of *goodness*
- ▶ Introduced the basics of (P)CFGs, and one particular parameter estimation method.
 - ▶ **project idea:** re-implement Kate's model as a PCFG learner.
 - ▶ **presentation papers:** ??

Roadmap

- ▶ **Lecture 2 (today):** rule extraction, decoding (parsing perspective)
- ▶ **Lecture 3:** rule extraction, decoding (MT perspective)
- ▶ **Lecture 4:** structured classification and prediction.

References I

- Börschinger, B., Jones, B. K., and Johnson, M. (2011). Reducing grounded learning tasks to grammatical inference. In *Proceedings of EMNLP-2011*, pages 1416–1425.
- Crouch, D. and King, T. H. (2006). Semantics via f-structure rewriting. In *Proceedings of the LFG06 Conference*, pages 145–165.
<http://www2.parc.com/isl/groups/nltt/papers/lfg06crouchking-PREPRINT.pdf>.
- Kate, R. J., Wong, Y. W., and Mooney, R. J. (2005). Learning to transform natural to formal languages. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1062.
<http://www.aaai.org/Library/AAAI/2005/aaai05-168.php>.
- Kim, J. and Mooney, R. J. (2012). Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *Proceedings of EMNLP-CoNLL-2012*, Jeju Island, Korea.
- Meyers, A., Kosaka, M., and Grishman, R. (2000). Chart-based transfer rule application in machine translation. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 537–543. Association for Computational Linguistics.
- Woods, W. A. (1973). Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4–8, 1973, National Computer Conference and Exposition*, pages 441–450.