Mildly Context-Sensitive Grammar Formalisms:

LCFRS: Data-driven Parsing

Laura Kallmeyer
Heinrich-Heine-Universität Düsseldorf

Sommersemester 2011

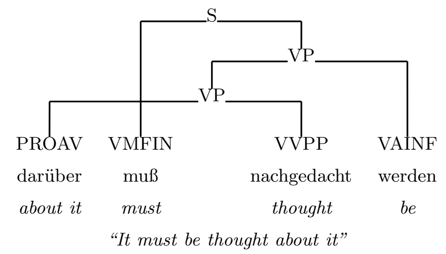
Overview

1. Introduction
2. Weighted Deductive Parsing
3. PLCFRS Parsing
4. Grammar Extraction
5. Evaluation

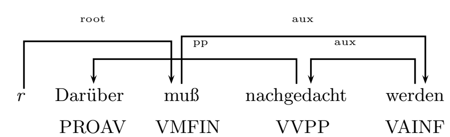
Introduction (1)

Discontinuous constituents and non-projective dependencies are rather frequent, in particular in so-called free word order languages.

Fronting example from German:



Introduction (2)



Appr. 25% of the sentences in Negra display discontinuous constituents.

Introduction (3)

- CFGs cannot describe discontinuous constituents.
- Therefore, if we want to learn a grammar model that includes discontinuous constituents, we need a formalism with an extended domain of locality.
- [Kallmeyer and Maier, 2010, Maier, 2010, Maier and Kallmeyer, 2010] use Linear Context-Free Rewriting Systems.

LCFRS can be conceived as a natural extension of CFG and many of the PCFG parsing techniques can be applied to probabilistic LCFRS.

Weighted Deductive Parsing (1)

Idea of weighted deductive parsing [Nederhof, 2003]:

- Give a deductive definition of the probability of a parse tree.
- Use Knuth's algorithm to compute the best parse tree for category S and a given input w.

Advantage:

- Yields the best parse without exhaustive parsing.
- Can be used to parse any grammar formalism as long as an appropriate weighted deductive system can be defined.

Weighted Deductive Parsing (2)

- A probabilistic context-free grammar (PCFG) is a CFG whose productions are equipped with probabilities.
- It holds that for every non-terminal A, the sum of the probabilities of all A-rules is 1.

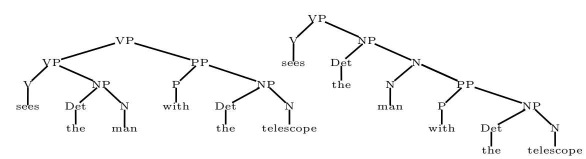
Example:

.8	VP → V NP	1	V → sees
.2	VP → VP PP	1	Det → the
1	NP → Det N	1	P → with
1	PP → P NP	.6	N → man
.1	N → N PP	.3	N → telescope

Weighted Deductive Parsing (3)

Goal: for a given input, find the parse tree with the highest probability.

- Probability of a parse tree: product of the probabilities of the rules used to generate the parse tree.
- Probability of a category A spanning a string w: maximal probability of parse trees with root label A and yield w.



$p = 0.6 \cdot 0.8 \cdot 0.2 \cdot 0.3 = 0.0288$ $\qquad$ $p = 0.6 \cdot 0.8 \cdot 0.1 \cdot 0.3 = 0.0144$

p(VP,sees the man with the telescope) $= 0.0288$

Weighted Deductive Parsing (4)

Example: Bottom-up CFG parsing (CYK) with Chomsky Normal Form.

For an input $w = w_1 \cdots w_n$ with $|w| = n$,

1. Item form $[A, i, j]$ with A a non-terminal, $0 \leq i \leq j \leq n$.
2. Deduction rules:

   Scan: $$\frac{}{[A, i-1, i]} \quad A \to w_i$$

   Complete: $$\frac{[B, i, j], [C, j, k]}{[A, i, k]} \quad A \to B\ C$$

3. Goal item: $[S, 0, n]$.

Weighted Deduction Parsing (5)

Extension to a weighted deduction system:

- Each item has an additional weight. Intuition: weight = costs to build an item.
- The deduction rules specify how to compute the weight of the consequent item form the weights of the antecedent items.

Example:

Scan: $$\frac{}{|log(p)| : [A, i-1, i]} \quad p : A \to w_i$$

Complete: $$\frac{x_1 : [B, i, j], x_2 : [C, j, k]}{x_1 + x_2 + |log(p)| : [A, i, k]} \quad p : A \to B\ C$$

Weighted Deduction Parsing (6)

- There is a linear order $<$ defined on the weights.
- The lower the weight, the better the item.
- For Knuth's algorithm, the weight functions f must be monotone nondecreasing in each variable and $f(x_1, \ldots, x_m) \geq max(x_1, \ldots, x_m)$.

In our example, this is the case:

Complete: $$\frac{x_1 : [B, i, j], x_2 : [C, j, k]}{x_1 + x_2 + |log(p)| : [A, i, k]} \quad p : A \to B\ C$$

$f(x_1, x_2) = x_1 + x_2 + c$ where $c \geq 0$ is a constant.

Weighted Deduction Parsing (7)

Algorithm for computing the goal item with the lowest weight, goes back to Knuth.

Goal: Find possible items with their lowest possible weight.

We need two sets:

- A set $\mathcal{C}$ (the chart) that contains items that have reached their final weight.
- A set $\mathcal{A}$ (the agenda) that contains items that are waiting to be processed as possible antecendents in further rule applications and that have not necessarily reached their final weight.

Initially, $\mathcal{C} = \emptyset$ and $\mathcal{A}$ contains all items that can be deduced from an empty antecedent set. Their weights are the minima of the possible weights.

Weighted Deductive Parsing (8)

```
while A ≠ ∅ do
   remove the best item x : I from A and add it to C
   if I goal item
   then stop and output true
   else
      for all y : I′ deduced from x : I and items in C:
         if there is no z with z : I′ ∈ C or z : I′ ∈ A
         then add y : I′ to A
         else if z : I′ ∈ A for some z
            then update weight of I′ in A to min(y, z)
```


Weighted Deductive Parsing (9)

If the weight functions are as required, then the following is guaranteed:

- Whenever an item is the best in the agenda, you have found its lowest weight.
- Therefore, if this item is a goal item, then you have found the best parse tree for your input.
- If it is no goal item, you can store it in the chart.

$\Rightarrow$ no exhaustive parsing needed.

However: $\mathcal{A}$ needs to be treated as a priority queue which can be expensive.

Weighted Deductive Parsing (10)

.2	S → AB	1	A → a	1	B → bc		
.8	S → CD	1	C → ab	.5	D → c	.5	D → e

Input: abc

Chart	Agenda
	$0 : [A, 0, 1]$, $0 : [B, 1, 3]$, $0 : [C, 0, 2]$, $0.3 : [D, 2, 3]$
$0 : [A, 0, 1]$	$0 : [B, 1, 3]$, $0 : [C, 0, 2]$, $0.3 : [D, 2, 3]$
$0 : [A, 0, 1]$, $0 : [B, 1, 3]$	$0 : [C, 0, 2]$, $0.3 : [D, 2, 3]$, $0.7 : [S, 0, 3]$
$0 : [A, 0, 1]$, $0 : [B, 1, 3]$, $0 : [C, 0, 2]$	$0.3 : [D, 2, 3]$, $0.7 : [S, 0, 3]$
$0 : [A, 0, 1]$, $0 : [B, 1, 3]$, $0 : [C, 0, 2]$, $0.3 : [D, 2, 3]$	$0.4 : [S, 0, 3]$, $0.7 : [S, 0, 3]$

Weighted Deductive Parsing (11)

Extension to parsing:

- Whenever we generate a new item, we store it not only with its weight but also with backpointers to its antecedent items.
- Whenever we update the weight of an item, we also have to update the backpointers.

In order to read off the best parse tree, we have to start from the best goal item and follow the backpointers.

PLCFRS Parsing (1)

A *probabilistic LCFRS* (PLCFRS) is a tuple $\langle N, T, V, P, S, p\rangle$ such that

1. $\langle N, T, V, P, S\rangle$ is a LCFRS and
2. $p : P \to [0..1]$ a function such that for all $A \in N$: $\Sigma_{A(\vec{x})\to\vec{\Phi}\in P}\, p(A(\vec{x}) \to \vec{\Phi}) = 1$.

PLCFRS Parsing (2)

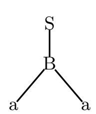
Example:

$0.2 : S(X) \to A(X)$	$0.8 : S(XY) \to B(X, Y)$
$0.7 : A(aX) \to A(X)$	$0.3 : A(a) \to \varepsilon$
$0.8 : B(aX, aY) \to B(X, Y)$	$0.2 : B(a, a) \to \varepsilon$

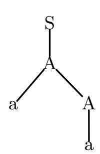
String language is a^+. Words with an even number of as and nested dependencies are more probable than words with a right-linear dependency structure.

The two analyses of aa:

$p = 0.16$



$p = 0.042$



PLCFRS Parsing (3)

Weighted deductive CYK parsing:

Scan: $$\frac{}{0 : [A, \langle\langle i, i+1\rangle\rangle]} \quad A \text{ POS tag of } w_{i+1}$$

Unary: $$\frac{in : [B, \vec{\rho}]}{in + |log(p)| : [A, \vec{\rho}]} \quad p : A(\vec{\alpha}) \to B(\vec{\alpha}) \in P$$

Binary: $$\frac{in_B : [B, \vec{\rho_B}], in_C : [C, \vec{\rho_C}]}{in_B + in_C + |log(p)| : [A, \vec{\rho_A}]}$$

where $p : A(\vec{\rho_A}) \to B(\vec{\rho_B})C(\vec{\rho_C})$ is an instantiated rule.

Goal: $[S, \langle\langle 0, n\rangle\rangle]$

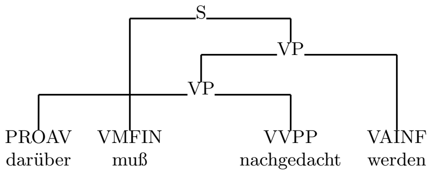
PLCFRS Parsing (4)

Parsing of aa with sample grammar:

chart	agenda
	$0.5 : [A, \langle 0, 1\rangle], 0.5 : [A, \langle 1, 2\rangle], 0.7 : [B, \langle 0, 1\rangle, \langle 1, 2\rangle]$
$0.5 : [A, \langle 0, 1\rangle]$	$0.5 : [A, \langle 1, 2\rangle], 0.7 : [B, \langle 0, 1\rangle, \langle 1, 2\rangle], 1.2 : [S, \langle 0, 1\rangle]$
$0.5 : [A, \langle 0, 1\rangle], 0.5 : [A, \langle 1, 2\rangle]$	$0.65 : [A, \langle 0, 2\rangle], 0.7 : [B, \langle 0, 1\rangle, \langle 1, 2\rangle], 1.2 : [S, \langle 0, 1\rangle], 1.2 : [S, \langle 1, 2\rangle]$
$0.5 : [A, \langle 0, 1\rangle], 0.5 : [A, \langle 1, 2\rangle], 0.65 : [A, \langle 0, 2\rangle]$	$0.7 : [B, \langle 0, 1\rangle, \langle 1, 2\rangle], 1.2 : [S, \langle 0, 1\rangle], 1.2 : [S, \langle 1, 2\rangle], 1.35 : [S, \langle 0, 2\rangle]$
$0.5 : [A, \langle 0, 1\rangle], 0.5 : [A, \langle 1, 2\rangle], 0.65 : [A, \langle 0, 2\rangle], 0.7 : [B, \langle 0, 1\rangle, \langle 1, 2\rangle]$	$0.8 : [S, \langle 0, 2\rangle], 1.2 : [S, \langle 0, 1\rangle], 1.2 : [S, \langle 1, 2\rangle]$

Grammar Extraction: Treebank Grammar (1)

Advantage of LCFRS: It can be straightforwardly extracted from treebanks with crossing branches [Maier and Søgaard, 2008].



PROAV(Darüber)	$\rightarrow$	ε	VMFIN(muß)	$\rightarrow$	ε
VVPP(nachgedacht)	$\rightarrow$	ε	VAINF(werden)	$\rightarrow$	ε

$$\mathrm{S}_1(X_1X_2X_3) \rightarrow \mathrm{VP}_2(X_1, X_3)\ \mathrm{VMFIN}(X_2)$$
$$\mathrm{VP}_2(X_1, X_2X_3) \rightarrow \mathrm{VP}_2(X_1, X_2)\ \mathrm{VAINF}(X_3)$$
$$\mathrm{VP}_2(X_1, X_2) \rightarrow \mathrm{PROAV}(X_1)\ \mathrm{VVPP}(X_2)$$

Grammar Extraction: Treebank Grammar (2)

For a given treebank tree $\langle V, E, r, l\rangle$ where V is the set of nodes, $E \subset V \times V$ the set of immediate dominance edges, $r \in V$ the root node and $l : V \rightarrow N \cup T$ the labeling function, the algorithm constructs the following rules: Let us assume that $w_1, \dots, w_n$ are the terminal labels of the leaves in $\langle V, E, r\rangle$ with a linear precedence relation $w_i \prec w_j$ for $1 \leq i < j \leq n$. We introduce a variable X_i for every w_i, $1 \leq i \leq n$.

- For every pair of nodes $v_1, v_2 \in V$ with $\langle v_2, v_2\rangle \in E$, $l(v_2) \in T$, we add $l(v_1)(l(v_2)) \rightarrow \varepsilon$ to the rules of the grammar.

Grammar Extraction: Treebank Grammar (3)

- For every node $v \in V$ with $l(v) = A_0 \notin T$ such that there are exactly m nodes $v_1, \dots, v_m \in V$ $(m \geq 1)$ with $\langle v, v_i\rangle \in E$ and $l(v_i) = A_i \notin T$ for all $1 \leq i \leq m$, we now create a rule

  $$A_0(x_1^{(0)}, \dots, x_{dim(A_0)}^{(0)}) \rightarrow A_1(x_1^{(1)}, \dots, x_{dim(A_1)}^{(1)}) \dots A_m(x_1^{(m)}, \dots, x_{dim(A_m)}^{(m)})$$

  where for the A_i, $0 \leq i \leq m$, the following must hold:

  1. The concatenation of all arguments of A_i, $x_1^{(i)} \dots x_{dim(A_i)}^{(i)}$ is the concatenation of all $X \in \{X_i \mid \langle v_i, v_i'\rangle \in E^*$ with $l(v_i') = w_i\}$ such that X_i precedes X_j if $i < j$, and
  2. a variable X_j with $1 \leq j < n$ is the right boundary of an argument of A_i if and only if $X_{j+1} \notin \{X_i \mid \langle v_i, v_i'\rangle \in E^*$ with $l(v_i') = w_i\}$, i.e., an argument boundary is introduced at each discontinuity.

As a further step, in this new rule, all right-hand side arguments of length > 1 are replaced in both sides of the rule with a single new variable.

Finally, all non-terminals A in the rules are equipped with an additional subscript $dim(A)$ which gives us the final non-terminal in our LCFRS.

The probabilities are then computed based on the frequencies of rules in the treebank, using a Maximum Likelihood estimator (MLE).

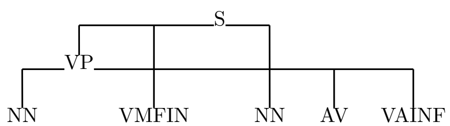
Grammar Extraction: Binarization (1)

Binarization of the extracted LCFRSs: as in the CNF transformation for CFG:

- We introduce a non-terminal for each RHS longer than 2 and split the rule into two rules, using this new intermediate non-terminal.

  This is repeated until all RHS are of length 2.
- Before binarizing, we reorder the RHS such that a head-outward binarization is obtained: First, all elements to the right of the head are listed in reverse order, then all elements to the left of the head in their original order and then the head itself.

Grammar Extraction: Binarization (2)



das muß man jetzt machen

that must one now do

"One has to do that now"

Rule extracted for the S node:

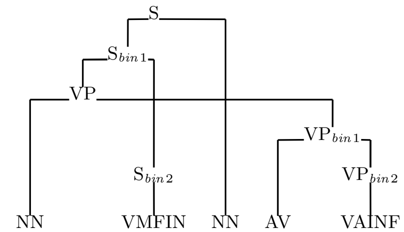
S($XYZU$) → VP(X,U) VMFIN(Y) NN(Z)

Reordering for head-outward binarization:

S($XYZU$) → NN(Z) VP(X,U) VMFIN(Y)

Grammar Extraction: Binarization (3)

Result of binarizing the tree:



Grammar Extraction: Markovization (1)

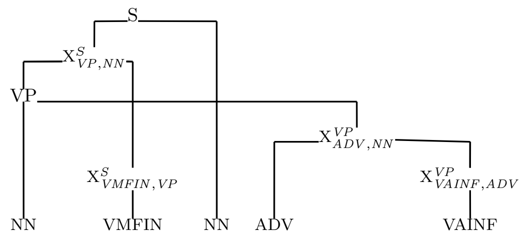
Proposed in [Collins, 1999] for PCFGs.

- Introduce only a single new non-terminal for the new rules obtained during binarization;
- add vertical and horizontal context from the original trees to each occurrence of this new non-terminal.

- As vertical context, we add the first v labels on the path from the root node of the tree that we want to binarize to the root of the entire treebank tree.
- As horizontal context, during binarization of a rule $A(\vec{\alpha}) \to A_0(\vec{\alpha_0}) \dots A_m(\vec{\alpha_m})$, for the new non-terminal that comprises the RHS elements $A_i \dots A_m$ (for some $1 \leq i \leq m$), we add the first h elements of $A_i, A_{i-1}, \dots, A_0$.

Grammar Extraction: Markovization (2)

Markovization with $v = 1, h = 2$



Evaluation (1)

Data: NeGra treebank [Skut et al., 1997], all sentences of length ≤ 30. Separation into the first 90% as training set and the remaining 10% as test set.

	training	test
number of sentences	16,502	1,833
av. sentence length	14.56	14.62
av. tree height	4.62	4.72
av. children per node	2.96	2.94
sentences without gaps	12,481 (75.63%)	1,361 (74.25%)
sent. with one gap	3,320 (20.12%)	387 (21.11%)
sent. with ≥ 2 gaps	701 (4.25%)	85 (4.64%)
max. gap number	6	5

Evaluation (2)

For the evaluation of the constituency parses, we use an EVALB-style metric: For a tree over a string w, a single constituency is represented by a tuple $\langle A, \vec{\rho} \rangle$ with A a node label and $\vec{\rho} \in (Pos(w) \times Pos(w))^{dim(A)}$.

We compute precision, recall and F_1 based on these tuples from gold and parsed test data:

- Precision: $\frac{\text{number of correct parsed constituencies}}{\text{number of parsed constituencies}}$
- Recall: $\frac{\text{number of correct parsed constituencies}}{\text{number of gold constituencies}}$
- F is the harmonic mean of precision and recall: $F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$.

Evaluation (3)

Markovization settings $v = 1$ and $h = 2$.

	PLCFRS	PCFG
LP	73.76	75.37
LR	74.41	75.83
LF_1	74.09	75.60
UP	77.05	78.41
UR	77.72	78.89
UF_1	77.38	78.65

PCFG = same experiment but with a version of NeGra where crossing branches are eliminated before parsing (i.e., a "context-free" version of NeGra).

The parser is implemented in Java and freely available under GPL:

`http://www.wolfgang-maier.net/rparse`

Conclusion

- LCFRS have an extended domain of locality; their non-terminals can span discontinuous tuples of strings.
- LCFRS can be straightforwardly extracted from constituency treebanks with crossing branches.
- Data-driven PLCFRS parsing of constituency treebanks yields competitive results, compared to PCFG parsing.

References

[Collins, 1999] Collins, M. (1999). *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania.

[Kallmeyer and Maier, 2010] Kallmeyer, L. and Maier, W. (2010). Data-driven parsing with probabilistic Linear Context-Free Rewriting Systems. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, China.

[Maier, 2010] Maier, W. (2010). Direct parsing of discontinuous constituents in German. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 58–66, Los Angeles, CA, USA. Association for Computational Linguistics.

[Maier and Kallmeyer, 2010] Maier, W. and Kallmeyer, L. (2010). Discontinuity and non-projectivity: Using mildly context-sensitive formalisms for data-driven parsing. In *Proceedings of the Tenth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10)*, New Haven.

[Maier and Søgaard, 2008] Maier, W. and Søgaard, A. (2008). Treebanks and mild context-sensitivity. In *Proceedings of the 13th Conference on Formal Grammar 2008*, Hamburg, Germany.

[Nederhof, 2003] Nederhof, M.-J. (2003). Weighted Deductive Parsing and Knuth's Algorithm. *Computational Linguistics*, 29(1):135–143.

[Skut et al., 1997] Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1997). An Annotation Scheme for Free Word Order Languages. In *Proceedings of the Fifth Conference on Applied*

Natural Language Processing (ANLP), Washington, D.C.