EVALB, Improving CKY Parsing, Hw3

Scott Farrar
CLMA, University of Washington
farrar@u.washington.edu

January 28, 2010

Today's lecture

1. Evaluating parsers
2. Hw3
3. Optimization: tips and tricks
   - 1. Size of the grammar
   - 2. Limit rules added to chart
   - 3. Sentence length

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Parsing: dev/train/test paradigm

The Wall Street Journal (WSJ) section of the Penn Treebank (PTB), for all its faults, provides a very useful resource for comparing parser performance.

Parsing: dev/train/test paradigm

The Wall Street Journal (WSJ) section of the Penn Treebank (PTB), for all its faults, provides a very useful resource for comparing parser performance.

In building a probabilistic parser, there are four kinds of resources that are commonly used esp. in the ACL related literature:

Parsing: dev/train/test paradigm

The Wall Street Journal (WSJ) section of the Penn Treebank (PTB), for all its faults, provides a very useful resource for comparing parser performance.

In building a probabilistic parser, there are four kinds of resources that are commonly used esp. in the ACL related literature:

1. **training data**: large number of annotated sentences (sec. 2–21 of PTB has 39,830 sentences)

Parsing: dev/train/test paradigm

The Wall Street Journal (WSJ) section of the Penn Treebank (PTB), for all its faults, provides a very useful resource for comparing parser performance.

In building a probabilistic parser, there are four kinds of resources that are commonly used esp. in the ACL related literature:

1. **training data**: large number of annotated sentences (sec. 2–21 of PTB has 39,830 sentences)
2. **development data**: small number of annotated sentences used to "tweak" parser (sec. 22, of PTB)

Scott Farrar CLMA, University of Washington farrar@u.washington.ed

Parsing: dev/train/test paradigm

The Wall Street Journal (WSJ) section of the Penn Treebank (PTB), for all its faults, provides a very useful resource for comparing parser performance.

In building a probabilistic parser, there are four kinds of resources that are commonly used esp. in the ACL related literature:

1. **training data**: large number of annotated sentences (sec. 2–21 of PTB has 39,830 sentences)
2. **development data**: small number of annotated sentences used to "tweak" parser (sec. 22, of PTB)
3. **test data**: small-medium number of un-annotated sentences used as input to parser (sec. 23 of PTB has 2416 sentences, $\sim 6\%$ of training set)

Parsing: dev/train/test paradigm

The Wall Street Journal (WSJ) section of the Penn Treebank (PTB), for all its faults, provides a very useful resource for comparing parser performance.

In building a probabilistic parser, there are four kinds of resources that are commonly used esp. in the ACL related literature:

1. **training data**: large number of annotated sentences (sec. 2–21 of PTB has 39,830 sentences)
2. **development data**: small number of annotated sentences used to "tweak" parser (sec. 22, of PTB)
3. **test data**: small-medium number of un-annotated sentences used as input to parser (sec. 23 of PTB has 2416 sentences, $\sim 6\%$ of training set)
4. **gold standard**: annotated version of test data, with no errors (hidden till parser is developed)

Recall our discussion first day of class

Definition

objective criterion: that which a parser tries to maximize.

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Recall our discussion first day of class

Definition

objective criterion: that which a parser tries to maximize.

Definition

tree accuracy: (harsh) exact match criterion; 1 for perfect match, otherwise 0.

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Recall our discussion first day of class

Definition

objective criterion: that which a parser tries to maximize.

Definition

tree accuracy: (harsh) exact match criterion; 1 for perfect match, otherwise 0.

Non-exact matches can be very useful for some tasks: named entity extraction, information retrieval, document clustering

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

PARSEVAL

Definition

PARSEVAL measures: standard metrics for evaluation using the component pieces of a parse; a way to give partial credit.

`evalb` is an implementation of the PARSEVAL measures
The `evalb` program uses several PARSEVAL measures:

- labeled precision (LP)
- labeled recall (LR)
- F-measure
- cross bracketing

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

PARSEVAL: Labeled precision

Definition

Labeled Precision (LP): the average of how many brackets in the resulting parse tree **match** those in the gold standard (same span). Focusing in on specific problems can increase precision. Broadening your methodology can decrease precision. Labeled precision includes the node label as well.

$$LP = \frac{\#of\ correct\ constituents\ in\ candidate\ parse\ of\ s}{\#of\ total\ constituents\ in\ candidate\ parse\ of\ s}$$

PARSEVAL: Labeled recall

Definition

Labeled Recall (LR): the average of how many brackets in the gold standard are in the resulting parse. Did you get them all? **Coverage**. Focusing in on specific problems can decrease recall, because other problems may get ignored. Labeled recall includes the node label as well.

$$LR = \frac{\#of\ correct\ constituents\ in\ candidate\ parse\ of\ s}{\#of\ correct\ constituents\ in\ reference\ parse\ of\ s}$$

P, R errors

Example

PP attachment error

```
(S (NP (A a)) (VP(B b)    (PP (C c))) )  gold
(S (NP (A a)) (VP(B b) ) (PP (C c)))
```

P, R errors

Example

PP attachment error

```
(S (NP (A a)) (VP(B b)    (PP (C c))) )  gold
(S (NP (A a)) (VP(B b) ) (PP (C c)))
```

Constituents in gold: $S(0,3)$, $NP(0,1)$, $VP(1,3)$, $PP(2,3)$

P, R errors

Example

PP attachment error

```
(S (NP (A a)) (VP(B b)    (PP (C c))) )  gold
(S (NP (A a)) (VP(B b) ) (PP (C c)))
```

Constituents in gold: $S(0,3)$, $NP(0,1)$, $VP(1,3)$, $PP(2,3)$
Constituents in cand: $S(0,3)$, $NP(0,1)$, $VP(1,2)$, $PP(2,3)$

P, R errors

Example

PP attachment error

```
(S (NP (A a)) (VP(B b)    (PP (C c))) )  gold
(S (NP (A a)) (VP(B b) ) (PP (C c)))
```

Constituents in gold: $S(0,3)$, $NP(0,1)$, $VP(1,3)$, $PP(2,3)$
Constituents in cand: $S(0,3)$, $NP(0,1)$, $VP(1,2)$, $PP(2,3)$

Precision

$P = \frac{3}{4}$

P, R errors

Example

PP attachment error

```
(S (NP (A a)) (VP(B b)    (PP (C c))) )  gold
(S (NP (A a)) (VP(B b) ) (PP (C c)))
```

Constituents in gold: $S(0,3)$, $NP(0,1)$, $VP(1,3)$, $PP(2,3)$
Constituents in cand: $S(0,3)$, $NP(0,1)$, $VP(1,2)$, $PP(2,3)$

Precision

$P = \frac{3}{4}$

Recall

$P = \frac{3}{4}$

PARSEVAL: F-measure

Definition

F-measure is the weighted aggregation of precision and recall (harmonic mean).

PARSEVAL: F-measure

Definition

F-measure is the weighted aggregation of precision and recall (harmonic mean).

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

$0 \leq \beta \leq +\infty$

- When β is 1, P and R are weighted equally.
- When β is greater than 1, R is favored.
- When β is less than 1, P is favored.

PARSEVAL: F-measure

Equally weighted P and R

$$F_1 = \frac{(1^2 + 1) * 0.9 * 0.3}{1^2 * 0.9 + 0.3} = 0.45$$

PARSEVAL: F-measure

Equally weighted P and R

$$F_1 = \frac{(1^2 + 1) * 0.9 * 0.3}{1^2 * 0.9 + 0.3} = 0.45$$

Harmonic Mean

F_1 is the same as the harmonic mean:

$$HM(a_1, a_2, a_3, ..., a_n) = \frac{n}{\frac{1}{a_1}\frac{1}{a_2}\frac{1}{a_3}\cdots\frac{1}{a_n}}$$

$$\frac{2}{\frac{1}{0.9} + \frac{1}{0.3}} = 0.45$$

PARSEVAL: F-measure

Favoring R

$$F_2 = \frac{(2^2 + 1) * 0.9 * 0.3}{2^2 * 0.9 + 0.3} = 0.346$$

Favoring P

$$F_{.5} = \frac{(.5^2 + 1) * 0.9 * 0.3}{.5^2 * 0.9 + 0.3} = 0.643$$

What is F_0?

PARSEVAL: F-measure

Favoring R

$$F_2 = \frac{(2^2 + 1) * 0.9 * 0.3}{2^2 * 0.9 + 0.3} = 0.346$$

Favoring P

$$F_{.5} = \frac{(.5^2 + 1) * 0.9 * 0.3}{.5^2 * 0.9 + 0.3} = 0.643$$

What is F_0?

$$F_0 = \frac{(0^2 + 1) * 0.9 * 0.3}{0^2 * 0.9 + 0.3} = 0.9 = P$$

PARSEVAL: cross-bracketing

Definition

cross-bracketing: the average of how many constituents in the resulting parse tree cross over the brackets in the gold standard.

Example

```
Candidate
     (   (    (                )        )   )

Gold std
     (   (         (                   )   )   )
      w1   w2    w3    w4    w5    w6    w7    w8
```

One cross-bracket error.

PARSEVAL: cross-bracketing

Example

```
Candidate
     (   (     (                    )   )   )

Gold std
     (   (          (                )   )   )
      w1   w2    w3    w4    w5    w6    w7    w8
```

Also one cross-bracket error.

PARSEVAL: Perfect results

Sent. ID	Len.	Stat.	Recal	Prec.	Matched Bracket	Bracket gold	Bracket test	Cross Bracket	Words	Correct Tag Tags	Accracy
1	8	0	100.00	100.00	6	6	6	0	8	8	100.00
2	41	0	100.00	100.00	33	33	33	0	41	41	100.00
3	36	0	100.00	100.00	27	27	27	0	36	36	100.00
4	37	0	100.00	100.00	24	24	24	0	37	37	100.00
5	31	0	100.00	100.00	29	29	29	0	31	31	100.00
6	17	0	100.00	100.00	13	13	13	0	17	17	100.00
.....											
2413	23	0	100.00	100.00	13	13	13	0	23	23	100.00
2414	37	0	100.00	100.00	31	31	31	0	37	37	100.00
2415	15	0	100.00	100.00	14	14	14	0	15	15	100.00
2416	13	0	100.00	100.00	8	8	8	0	13	13	100.00
			100.00	100.00	49749	49749	49749	2416	60548	60548	100.00

PARSEVAL: Perfect results

```
=== Summary ===

-- All --
Number of sentence        =   2416
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =   2416
Bracketing Recall         = 100.00
Bracketing Precision      = 100.00
Complete match            = 100.00
Average crossing          =   0.00
No crossing               = 100.00
2 or less crossing        = 100.00
Tagging accuracy          = 100.00
```

PARSEVAL: Perfect results

```
-- len<=40 --
Number of sentence        =   2160
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =   2160
Bracketing Recall         = 100.00
Bracketing Precision      = 100.00
Complete match            = 100.00
Average crossing          =   0.00
No crossing               = 100.00
2 or less crossing        = 100.00
Tagging accuracy          = 100.00
No. of matched brackets   = 39896
No. of gold brackets      = 39896
No. of test brackets      = 39896
```

P, R errors

Example

Cross bracket error

```
(S (NP (A a)       (B b) ) (VP(C c) (PP (D d)))) gold
(S (NP (A a) ) (VP (B b)      (C c) (PP (D d))))
```

P, R errors

Example

Cross bracket error

```
(S (NP (A a)       (B b) ) (VP(C c) (PP (D d)))) gold
(S (NP (A a) ) (VP (B b)      (C c) (PP (D d))))
```

Constituents (gold): $S(0,4)$, $NP(0,2)$, $VP(2,4)$, $PP(3,4)$
Constituents (cand): $S(0,4)$, $NP(0,1)$, $VP(1,4)$, $PP(3,4)$

P, R errors

Example

Cross bracket error

```
(S (NP (A a)       (B b) ) (VP(C c) (PP (D d)))) gold
(S (NP (A a) ) (VP (B b)      (C c) (PP (D d))))
```

Constituents (gold): $S(0,4)$, $NP(0,2)$, $VP(2,4)$, $PP(3,4)$
Constituents (cand): $S(0,4)$, $NP(0,1)$, $VP(1,4)$, $PP(3,4)$

Precision

$P = \frac{2}{4}$

P, R errors

Example

Cross bracket error

```
(S (NP (A a)       (B b) ) (VP(C c) (PP (D d)))) gold
(S (NP (A a) ) (VP (B b)      (C c) (PP (D d))))
```

Constituents (gold): $S(0,4)$, $NP(0,2)$, $VP(2,4)$, $PP(3,4)$
Constituents (cand): $S(0,4)$, $NP(0,1)$, $VP(1,4)$, $PP(3,4)$

Precision

$P = \frac{2}{4}$

Recall

$P = \frac{2}{4}$

Cross-bracket

1 cross-bracket error

Explanation of PARSEVAL

Have a look at the parameters files in dropbox/.../571/tools/EVALB

See Manning & Schutze (1999), p. 433

Today's lecture

1. Evaluating parsers
2. Hw3
3. Optimization: tips and tricks
   - 1. Size of the grammar
   - 2. Limit rules added to chart
   - 3. Sentence length

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Homework 3

See website

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Homework 3

See website

CNF grammar

There's no need to use your 2CNF code, but knowing how the grammar was transformed is important.

- Unary rules: S_VP, NP_NP, etc.
- Non-binary rules: VP', NP', etc.

Hw3 Grammar

Collapsed Unaries

$S\_VP \rightarrow VB\ NP$

was originally:

Hw3 Grammar

Collapsed Unaries

$S\_VP \rightarrow VB\ NP$

was originally:

$S \rightarrow VP$

$VP \rightarrow VB\ NP$

Hw3 Grammar

Binarized Productions

$VP \rightarrow VP' \; PP$

where

$VP' \rightarrow VB \; PP$

was originally:

Hw3 Grammar

Binarized Productions

$VP \rightarrow VP' \; PP$

where

$VP' \rightarrow VB \; PP$

was originally:

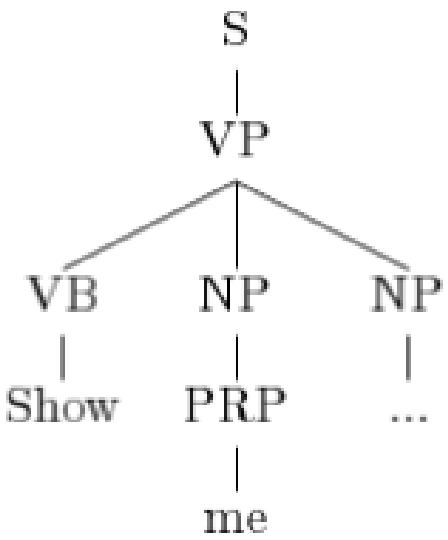
$VP \rightarrow VB \; PP \; PP$

Hw3 Grammar

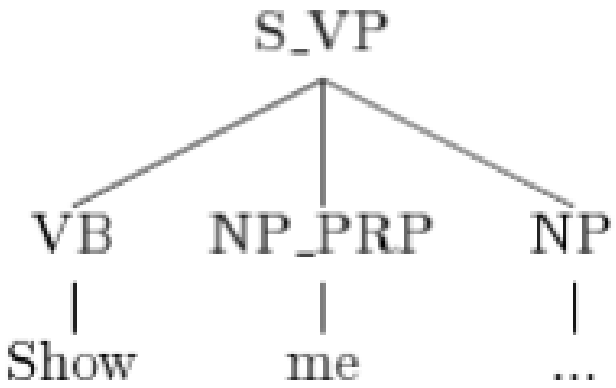
Combination

S_VP'

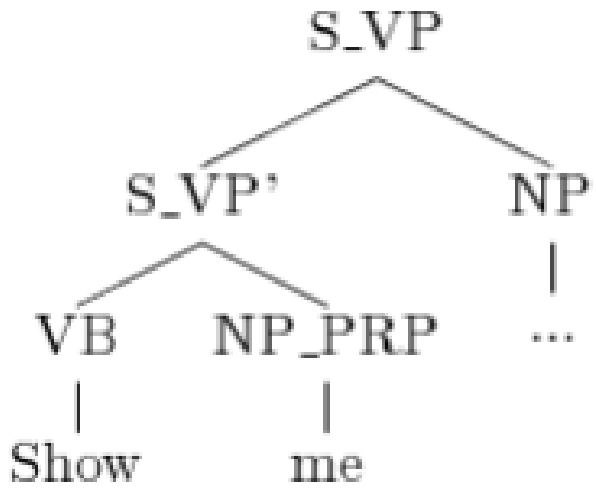
was originally:



Hw3 Grammar



Hw3 Grammar



Amendment to Task 4

Accuracy

You're asked to improve upon the baseline parser so that you get a better EVALB score.

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Amendment to Task 4

Accuracy

You're asked to improve upon the baseline parser so that you get a better EVALB score.

Efficiency

We'll also accept improved parsers that are more efficient, not necessarily more accurate. That is, improve the runtime of the parser without significantly degrading the efficiency.

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Today's lecture

1. Evaluating parsers
2. Hw3
3. Optimization: tips and tricks
   - 1. Size of the grammar
   - 2. Limit rules added to chart
   - 3. Sentence length

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

General strategies

The efficiency of the basic CYK is $O(n^3|P|)$, where n is the average length of sentence and $|P|$ is the number of production rules. You can improve the efficiency by:

General strategies

The efficiency of the basic CYK is $O(n^3|P|)$, where n is the average length of sentence and $|P|$ is the number of production rules. You can improve the efficiency by:

General strategies

The efficiency of the basic CYK is $O(n^3|P|)$, where n is the average length of sentence and $|P|$ is the number of production rules. You can improve the efficiency by:

1. limiting the size of the grammar $|P|$

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

General strategies

The efficiency of the basic CYK is $O(n^3|P|)$, where n is the average length of sentence and $|P|$ is the number of production rules. You can improve the efficiency by:

1. limiting the size of the grammar $|P|$
2. limiting the number of states entered into the CKY chart (prune search space)

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

General strategies

The efficiency of the basic CYK is $O(n^3|P|)$, where n is the average length of sentence and $|P|$ is the number of production rules. You can improve the efficiency by:

1. limiting the size of the grammar $|P|$
2. limiting the number of states entered into the CKY chart (prune search space)
3. reducing n, where n is the length of the input sentence

General strategies

The efficiency of the basic CYK is $O(n^3|P|)$, where n is the average length of sentence and $|P|$ is the number of production rules. You can improve the efficiency by:

1. limiting the size of the grammar $|P|$
2. limiting the number of states entered into the CKY chart (prune search space)
3. reducing n, where n is the length of the input sentence

Trade-off

There is always a speed vs. accuracy trade-off in statistical parsing. Where's the sweet spot?

Limit the size of the grammar

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Limit the size of the grammar

- In a wide-coverage grammar, you will have 1,000s of rule types (CYK requires you to search the rule store over and over).

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Limit the size of the grammar

- In a wide-coverage grammar, you will have 1,000s of rule types (CYK requires you to search the rule store over and over).
- To handle a large number of rules, avoid creating so many rules to begin with.

Limit the size of the grammar

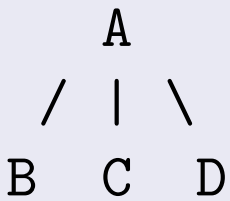
- In a wide-coverage grammar, you will have 1,000s of rule types (CYK requires you to search the rule store over and over).
- To handle a large number of rules, avoid creating so many rules to begin with.
- Conversion to CNF is the major cause of rule proliferation.

Limit the size of the grammar

- In a wide-coverage grammar, you will have 1,000s of rule types (CYK requires you to search the rule store over and over).
- To handle a large number of rules, avoid creating so many rules to begin with.
- Conversion to CNF is the major cause of rule proliferation.
- We can also prune away less important rules using a number of other techniques (recall Giazauskus paper).

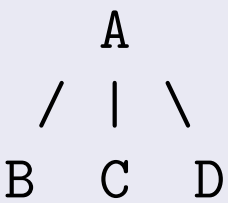
Binarization choices

Original tree in grammar

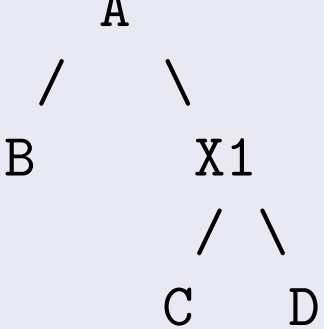


Binarization choices

Original tree in grammar

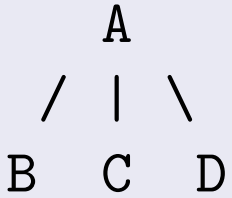


Right-factored



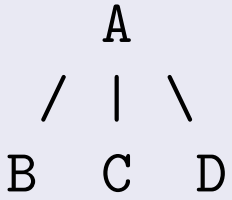
Binarization choices

Original tree in grammar

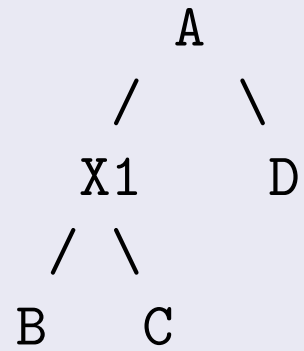


Binarization choices

Original tree in grammar



Left-factored



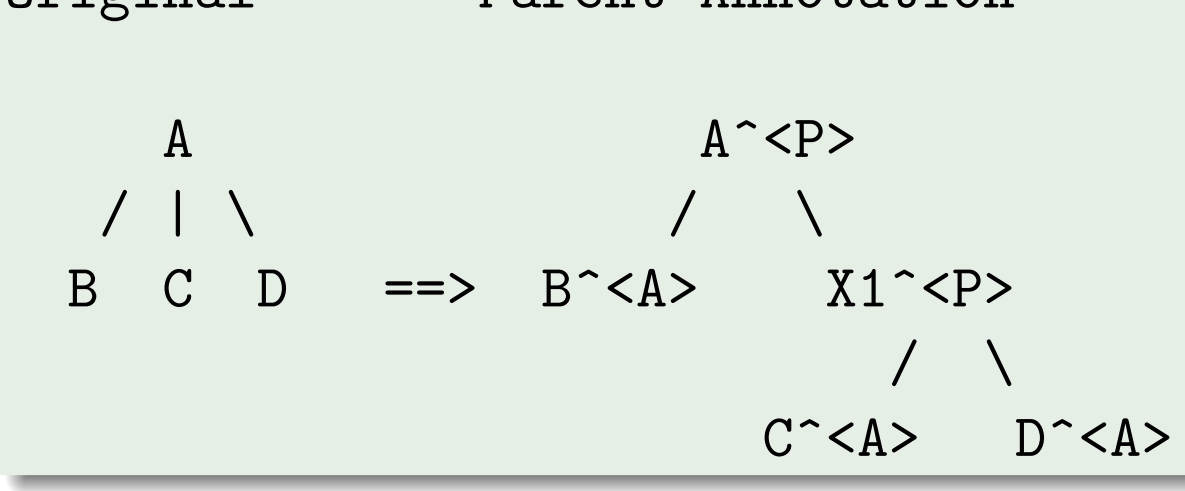
Be sure to see write-up of CNF conversion in the NLTK documentation of `nltk.treetransforms`.

Parent Rule Annotation

Definition

Parent rule annotation refers to the annotation of nodes with information about their ancestor nodes, as if you're giving the nodes a context. Could improve CKY from 74% to 79% accuracy.

Example



Horizontal factoring

Definition

Horizontal factoring refers to the way in which rules in the original grammar can be binarized such that information about the child nodes is encoded in new nodes (in CNF). Also called Markovization, this captures "context" among terminals. As the Markov order increases, the number rules in the converted CFG increases, but more information is captured in rules. Data sparsity is, as usual, a big problem.

Example

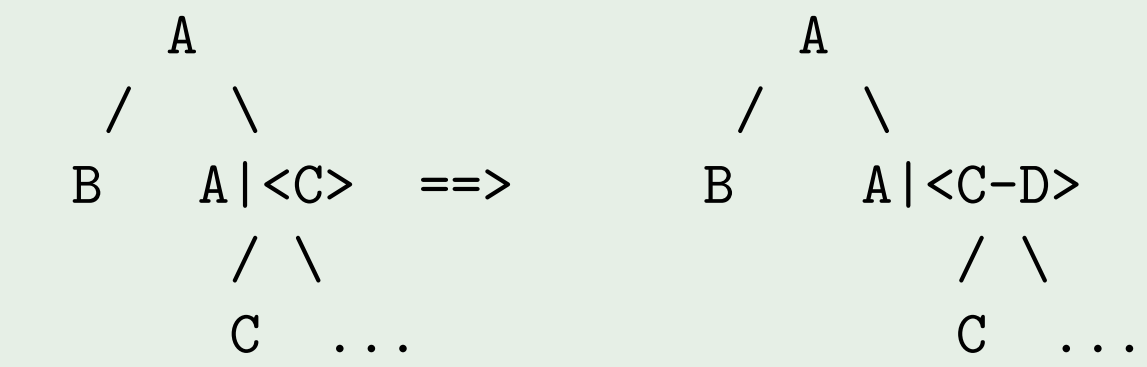
```
Original        Markov order 0

  __A__              A
 / /|\ \            /   \
B C D E F  ==>     B     X1
                        / \
                       C   X2
                           / \
                          D  ....
```

Horizontal factoring

Example

Markov order 1 Markov order 2 etc.



Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Horizontal factoring

Example

Original

```
  __A__
 / /|\ \
B C D E F
```

No smoothing, or order infinity

```
    A
   /   \
  B     A|<C-D-E-F>
       /   \
      C     ...
```

Affects Markov order-N smoothing on rule size

As reported in Mohri and Roark (2006)

Sections 02-23 of PTB-WSJ:

Affects Markov order-N smoothing on rule size

As reported in Mohri and Roark (2006)

Sections 02-23 of PTB-WSJ:

- Markov factor 0: 99 nonterminals, 3803 productions

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Affects Markov order-N smoothing on rule size

As reported in Mohri and Roark (2006)

Sections 02-23 of PTB-WSJ:

- Markov factor 0: 99 nonterminals, 3803 productions
- Markov factor 1: 564 nonterminals, 6354 productions

Affects Markov order-N smoothing on rule size

As reported in Mohri and Roark (2006)

Sections 02-23 of PTB-WSJ:

- Markov factor 0: 99 nonterminals, 3803 productions
- Markov factor 1: 564 nonterminals, 6354 productions
- Markov factor 2: 2492 nonterminals, 11659 productions

Affects Markov order-N smoothing on rule size

As reported in Mohri and Roark (2006)

Sections 02-23 of PTB-WSJ:

- Markov factor 0: 99 nonterminals, 3803 productions
- Markov factor 1: 564 nonterminals, 6354 productions
- Markov factor 2: 2492 nonterminals, 11659 productions
- Markov factor ∞: 10,105 nonterminals, 23,220 productions

Combined Effects of Markov ordering and Parent annotation

PCFG	Time(s)	Words/s	$\mid NTs \mid$	$\mid Prods \mid$	LR	LP	F
Right-factored, M-∞	4848	6.7	10105	23220	69.2	73.8	71.5
Right-factored, M–2	1302	24.9	2492	11659	68.8	73.8	71.3
Right-factored, M–1	445	72.7	564	6354	68.0	73.0	70.5
Right-factored, M–0	206	157.1	99	3803	61.5	65.5	63.3
Parent-annot., Rt-f M-2	7510	4.3	5876	22444	76.2	78.3	77.2

Converting to CNF

Note: different notation used for horizontal annotations.

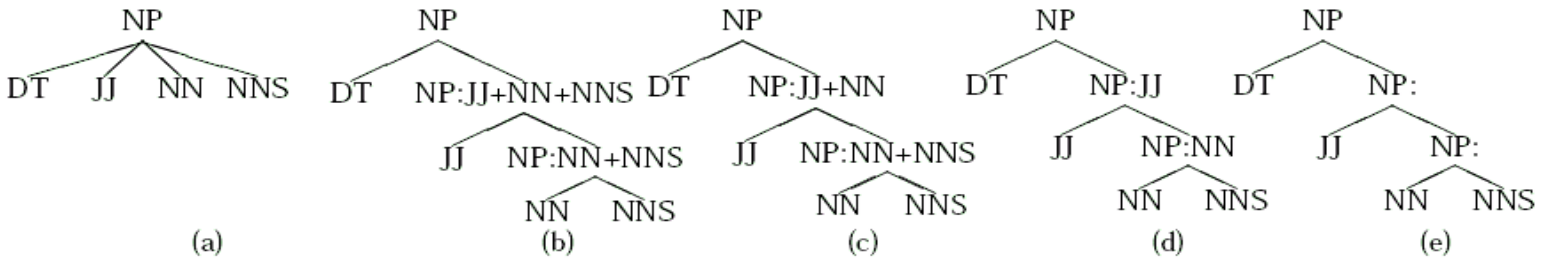


Figure 1: Five representations of an n-ary production, $n = 4$. (a) Original production (b) Right-factored production (c) Right-factored Markov order-2 (d) Right-factored Markov order-1 (e) Right-factored Markov order-0

In general, using no (or ∞ horizontal) factoring will give you better accuracy. But we have a rule explosion problem, so we'll compromise our accuracy for better parser runtime. (See Mohri & Roark 2006).

2. Limit rules entered into chart

Definition

Use **beam threshholding**, named after the evaluation function in a beam search algorithm. Beam search is a way to only explore nodes in a search tree that are most likely to yield an answer. Some strategies:

2. Limit rules entered into chart

Definition

Use **beam threshholding**, named after the evaluation function in a beam search algorithm. Beam search is a way to only explore nodes in a search tree that are most likely to yield an answer. Some strategies:

2. Limit rules entered into chart

Definition

Use **beam threshholding**, named after the evaluation function in a beam search algorithm. Beam search is a way to only explore nodes in a search tree that are most likely to yield an answer. Some strategies:

- Using beam width k (only allow k entries in cell): k=10, 100, or 200

2. Limit rules entered into chart

Definition

Use **beam threshholding**, named after the evaluation function in a beam search algorithm. Beam search is a way to only explore nodes in a search tree that are most likely to yield an answer. Some strategies:

- Using beam width k (only allow k entries in cell): k=10, 100, or 200
- With your small grammar: k=2, 5, or 10

2. Limit rules entered into chart

Definition

Use **beam threshholding**, named after the evaluation function in a beam search algorithm. Beam search is a way to only explore nodes in a search tree that are most likely to yield an answer. Some strategies:

- Using beam width k (only allow k entries in cell): k=10, 100, or 200
- With your small grammar: k=2, 5, or 10
- Remove all production rules with a frequency of 1 from grammar. Then try 2 and possibly 3. (variant)

2. Limit rules entered into chart

Definition

Use **beam threshholding**, named after the evaluation function in a beam search algorithm. Beam search is a way to only explore nodes in a search tree that are most likely to yield an answer. Some strategies:

- Using beam width k (only allow k entries in cell): k=10, 100, or 200
- With your small grammar: k=2, 5, or 10
- Remove all production rules with a frequency of 1 from grammar. Then try 2 and possibly 3. (variant)

2. Limit rules entered into chart

Definition

Use **beam threshholding**, named after the evaluation function in a beam search algorithm. Beam search is a way to only explore nodes in a search tree that are most likely to yield an answer. Some strategies:

- Using beam width k (only allow k entries in cell): k=10, 100, or 200
- With your small grammar: k=2, 5, or 10
- Remove all production rules with a frequency of 1 from grammar. Then try 2 and possibly 3. (variant)

Problem?

You aren't guaranteed to always find an answer (a parse).

More strategies

Heuristics

Within the CYK algorithm, heuristically throw away constituents that probably won't make it into a complete parse. In other words limit the number of nodes saved in each cell of the CYK table.

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

More strategies

Heuristics

Within the CYK algorithm, heuristically throw away constituents that probably won't make it into a complete parse. In other words limit the number of nodes saved in each cell of the CYK table.

Where x and y are constituents:

More strategies

Heuristics

Within the CYK algorithm, heuristically throw away constituents that probably won't make it into a complete parse. In other words limit the number of nodes saved in each cell of the CYK table.

Where x and y are constituents:

- Throw constituent x away if $p(x) < 10^{-200}$.

More strategies

Heuristics

Within the CYK algorithm, heuristically throw away constituents that probably won't make it into a complete parse. In other words limit the number of nodes saved in each cell of the CYK table.

Where x and y are constituents:

- Throw constituent x away if $p(x) < 10^{-200}$.
- Throw x away if $p(x) < 100 * p(y)$ for some y that spans the same set of words.
  Throw away $NP_{i,j}$ b/c $p(NP_{i,j}) = 0.00002571$, and $p(VP_{i,j}) = 0.0003211$

Dealing w. sentence length

- Since the EVALB package doesn't evaluate sentences greater than length 40, there's no need to attempt to parse them.
- Add a step to your algorithm to calculate the length of the input sentence, and then to return a blank line for sentences with length greater than 40.
- Thus, we reduce n, but of course this doesn't really improve the parser, just the eval numbers.

Scott Farrar
CLMA, University of Washington farrar@u.washington.ed

Effects of sentence length

Collins Parser results (Collins 1997) for words ≤ 40

labeled recall	label precision
88.1	88.6

Collins Parser results (Collins 1997) for words ≤ 100

labeled recall	label precision
87.5	88.1