Probabilistic Context-Free-Grammars

Miguel Ballesteros
7-11

Using some materials by Joakim Nivre from UU

Parsing with PCFGs

- Grammar Formalism
- Parsing Model
- Parsing Algorithms
- Learning with a treebank.
- Learning without a treebank.

PCFG

- The PCFG model is without doubt the most important formal model in syntactic parsing today,
- it is widely used in itself
- many later developments start from it.

Grammar Formalism

- A probabilistic context-free grammar (PCFG) is a simple extension of a context-free grammar in which every production rule is associated with a probability.

Grammar Formalism

- PCFG
  - G = (V, T, P, S, Q)
    - V: a finite set of variables, non-terminal symbols.
    - T: a finite set of terminal symbols (equiv. To Σ in FSAs)
    - P: a set of context free production rules, each of the form
      - $A \rightarrow \alpha$, where $A \in V$, $\alpha \in (V \cup T)^*$
    - S: a start non-terminal $S \in V$

Grammar Formalism

- PCFG
  - G = (V, T, P, S, Q)
    - V: a finite set of variables, non-terminal symbols.
    - T: a finite set of terminal symbols (equiv. To Σ in FSAs)
    - P: a set of context free production rules, each of the form
      - A → α , where A ϵ V, α ϵ (V U T)*
    - S: a start non-terminal S X ={x 1 , . . . , x m }. V
    - Q: P → [0, 1]

Grammar Formalism

- Q: P → [0, 1]

- It is a function that assigns a probability to each member of P.

S	→	NP VP .	1.00	JJ	→	Economic	0.33
VP	→	VP PP	0.33	JJ	→	little	0.33
VP	→	VBD NP	0.67	JJ	→	financial	0.33
NP	→	NP PP	0.14	NN	→	news	0.5
NP	→	JJ NN	0.57	NN	→	effect	0.5
NP	→	JJ NNS	0.29	NNS	→	markets	1.0
PP	→	IN NP	1.0	VBD	→	had	1.0
PU	→	.	1.0	IN	→	on	1.0

FIGURE 1. Probabilistic context-free grammar for a fragment of English.

Grammar Formalism

- As usual, we use L(G) to denote the string language generated by G.
  - Set of strings x over the alphabet for which there exists a derivation S =>* x using rules in P.
- We also have T(G) to denote the tree language generated by G:
  - the set of parse trees corresponding to valid derivations of strings in L(G)

Grammar Formalism

- We will also use:
  - ***Yield(y)*** for the terminal string associated with y (y is a parse tree).
  - ***Count(i,y)*** for the number of times that the ith-production rule pi ϵ P is used in the derivation of y.
  - ***LHS(i)*** for the nonterminal symbol in the left-hand-side of pi
  - ***qi:*** for the probability of rule pi (qi = Q(pi))

Grammar Formalism

- The probability of a parse tree y ϵ T(G) is defined as the product of probabilities of all rule application in the derivation of y:

$$P(y) = \prod_{i=1}^{|p|} q_i^{COUNT(i,y)}$$

Grammar Formalism

- The probability of a parse tree y ϵ T(G) is defined as the product of probabilities of all rule application in the derivation of y:

$$P(y) = \prod_{i=1}^{|p|} q_i^{COUNT(i,y)}$$

This follows from basic probability theory on the **assumption** that the application of a rule in the derivation of a tree is independent of all other rule application in that tree.

Grammar Formalism

The basic probability theory basically says that the joint probability of several events, Can be computed as the product of the probability of each event, given that They are independent.

…. we are assuming, for example that having a NP in lhs of a rule is independent of having a VP in the rhs of the true.

This does not sound as a good assumption... but we will later study how to fix this, and avoid this independence assumption

This follows from basic probability theory on the **assumption** that the application of a rule in the derivation of a tree is independent of all other rule application in that tree.

Grammar Formalism

- Since the yield of a parse tree uniquely determines the string associated with the tree.
  - The joint probability of a tree y ϵ T(G) and a string x ϵ T(G) P(x,y) is either
    - 0
    - Or equal to the probability of y.
  - Depending on whether or not the string matches the yield.

Grammar Formalism

- Since the yield of a parse tree uniquely determines the string associated with the tree.
  - The joint probability of a tree y ϵ T(G) and a string x ϵ T(G) P(x,y) is either
    - 0
    - Or equal to the probability of y.

the

In other words, in the string exists in the language, it has some kind of syntactic structure and thus we we can provide a parse tree for it, otherwise, we just ignore it and return nothing.

Grammar Formalism

- The probability of a string can be obtained by adding up the probabilities of all parse trees compatible with the string:

$$P(x) = \sum_{y \in T(G):\ \text{yield}(y)=x} P(y)$$

Grammar Formalism

- A PCFG is **proper iff** P defines a proper probability distribution over every subset of rules that have the same left-hand side A ϵ N:

$$\sum_{p \epsilon P:\ LHS(p)=A} Q(p) = 1$$

Grammar Formalism

- A PCFG is **consistent iff** it defines a proper probability over the set of trees it generates:

$$\sum_{y \epsilon T(G)} P(y) = 1$$

Consistency can also be defined in terms of the probability distribution over strings generated by the grammar.

Grammar Formalism

- A PCFG is **consistent iff** it defines a proper probability over the set of trees it generates:

$$\sum_{y \epsilon T(G)} P(y) = 1$$

It seems obvious and always possible, but it is a frequent situation to see grammars that are not capable of being consistent.

Parsing Model

- PCFGs have many applications in natural language processing, for example, in language modeling
  - for speech recognition or statistical machine translation, where they can be used to model the probability distribution of a string in a language.
- Here, we are going to look at them from the parsing perspective.

Parsing Model

- The input space is the set of all strings over T; $X = T^*$.
- The output space is the set of all parse trees over P; $Y = P^*$
  - Parse trees can be identified with sequences of production rules given some canonical form of derivation, such as leftmost derivation.
- The generative component is the context-free grammar defining a set of parse trees for each input;

$$GEN(x) = \{y \in T(G) \mid yield(y) = x\}.$$

- The evaluative component is the function Q defining a probability distribution over the parse trees for an input

$$EVAL(Y) = P(y) = \prod_{i=1}^{|p|} q_i^{COUNT(i,y)}$$

Parsing Model

- Note that the score P(y) is equal to the joint probability P(x,y) of the input sentence and the output tree.

- However, in a parsing model, it seems more natural to use the conditional probability P(y|x) since the sentence is given as input to the model.

$$P(y|x) = \frac{P(x,y)}{\sum_{y' \in \mathrm{GEN(x)}} P(y')}$$

Parsing Model

- Note that the score P(y) is equal to the joint probab
output

Joint probability P(x,y) is actually proportional to conditional probability P(y|x) , meaning that the probabilities are going to be different, but the ranking is going to be the same.

(the reason is that the denominator is going to be constant for this)

- Howev
natural
since th
model.

$$P(y|x) = \frac{P(x,y)}{\sum_{y' \in \mathrm{GEN(x)}} P(y')}$$

Parsing Model

- This PCFG generates two trees for the sentence

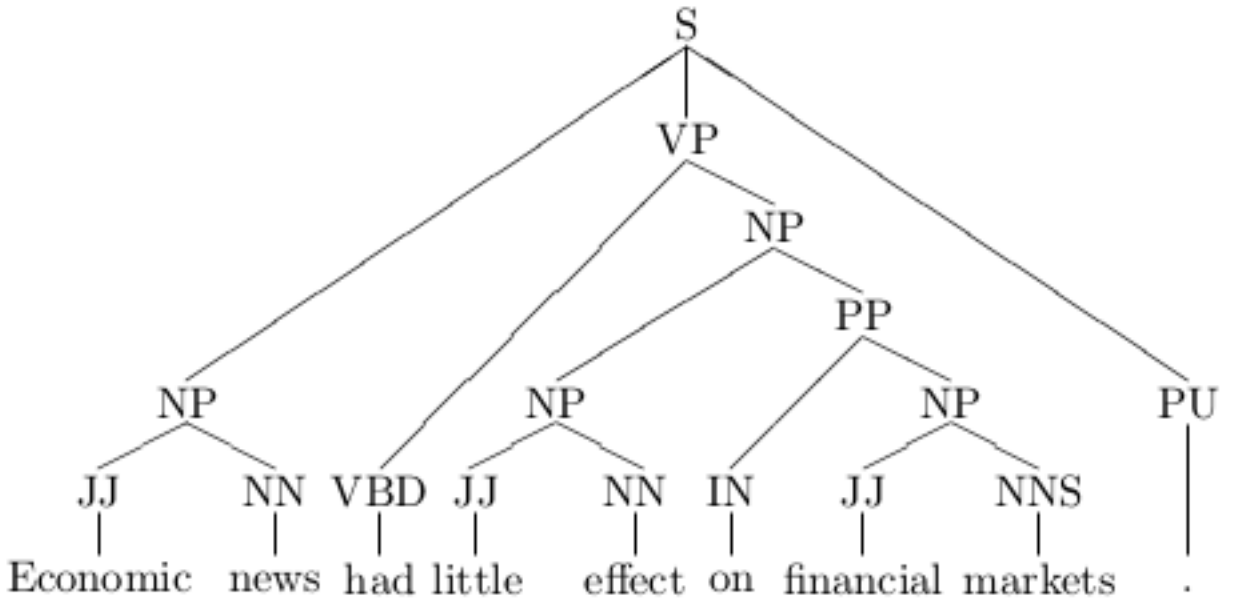
Economic news had little effect on financial markets.

S	→	NP VP .	1.00	JJ	→	Economic	0.33
VP	→	VP PP	0.33	JJ	→	little	0.33
VP	→	VBD NP	0.67	JJ	→	financial	0.33
NP	→	NP PP	0.14	NN	→	news	0.5
NP	→	JJ NN	0.57	NN	→	effect	0.5
NP	→	JJ NNS	0.29	NNS	→	markets	1.0
PP	→	IN NP	1.0	VBD	→	had	1.0
PU	→	.	1.0	IN	→	on	1.0

FIGURE 1. Probabilistic context-free grammar for a fragment of English.

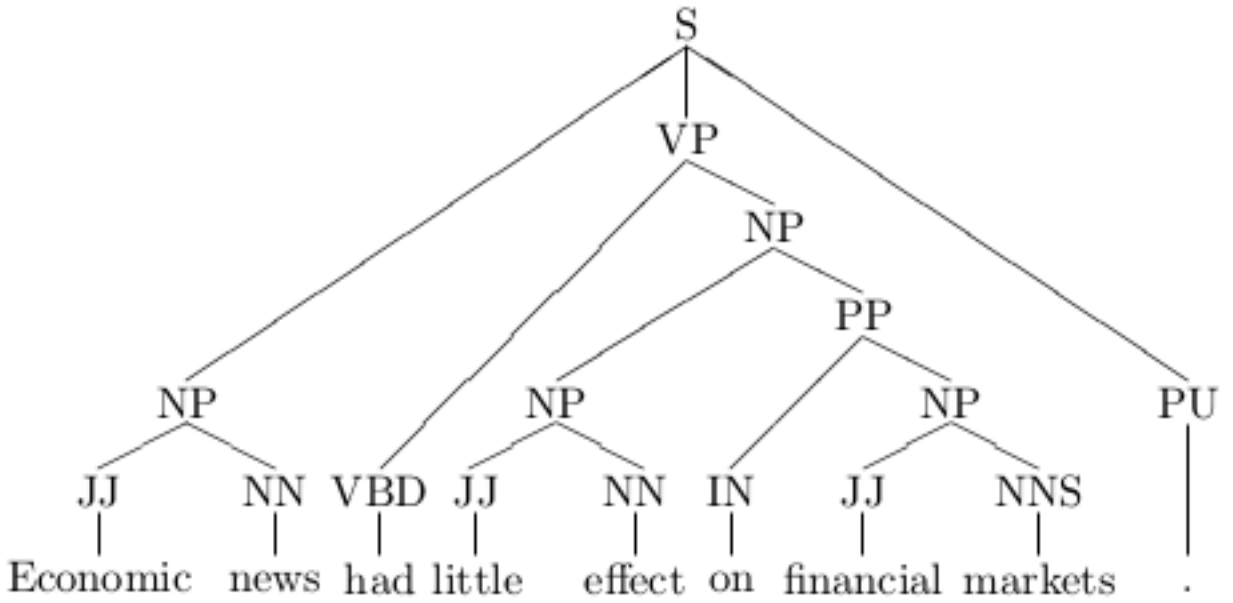
Parsing Model – Tree 1

S	→	NP VP .	1.00	JJ	→	Economic	0.33
VP	→	VP PP	0.33	JJ	→	little	0.33
VP	→	VBD NP	0.67	JJ	→	financial	0.33
NP	→	NP PP	0.14	NN	→	news	0.5
NP	→	JJ NN	0.57	NN	→	effect	0.5
NP	→	JJ NNS	0.29	NNS	→	markets	1.0
PP	→	IN NP	1.0	VBD	→	had	1.0
PU	→	.	1.0	IN	→	on	1.0



Parsing Model – Tree 1

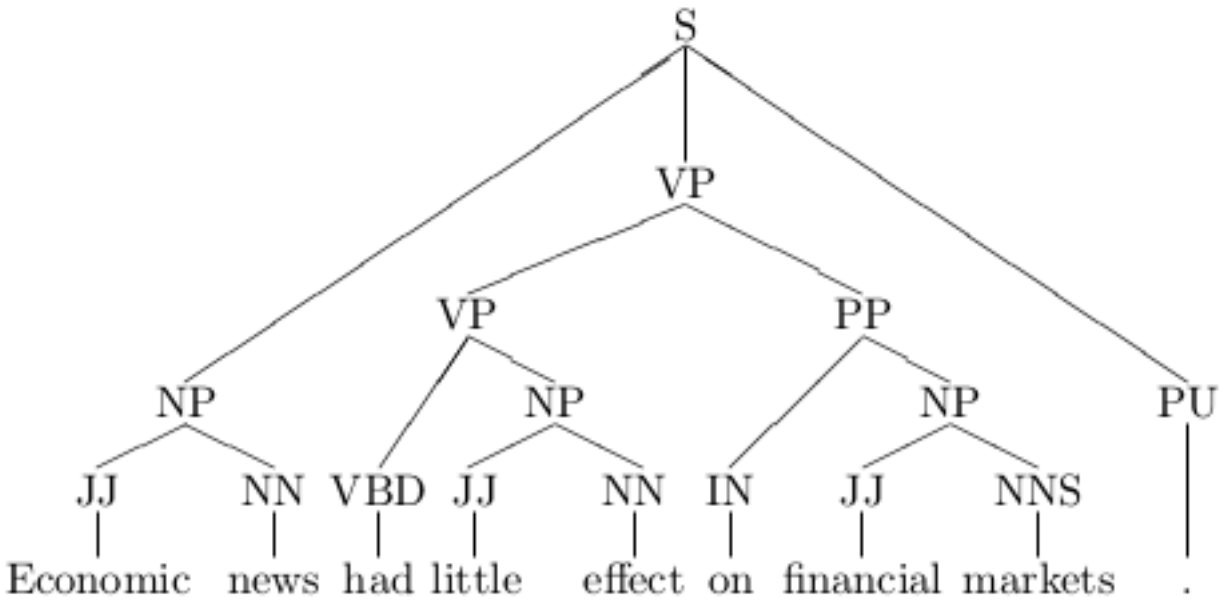
S	→	NP VP .	1.00	JJ	→	Economic	0.33
VP	→	VP PP	0.33	JJ	→	little	0.33
VP	→	VBD NP	0.67	JJ	→	financial	0.33
NP	→	NP PP	0.14	NN	→	news	0.5
NP	→	JJ NN	0.57	NN	→	effect	0.5
NP	→	JJ NNS	0.29	NNS	→	markets	1.0
PP	→	IN NP	1.0	VBD	→	had	1.0
PU	→	.	1.0	IN	→	on	1.0



The probability of this tree T(y), **p=0.0000794**

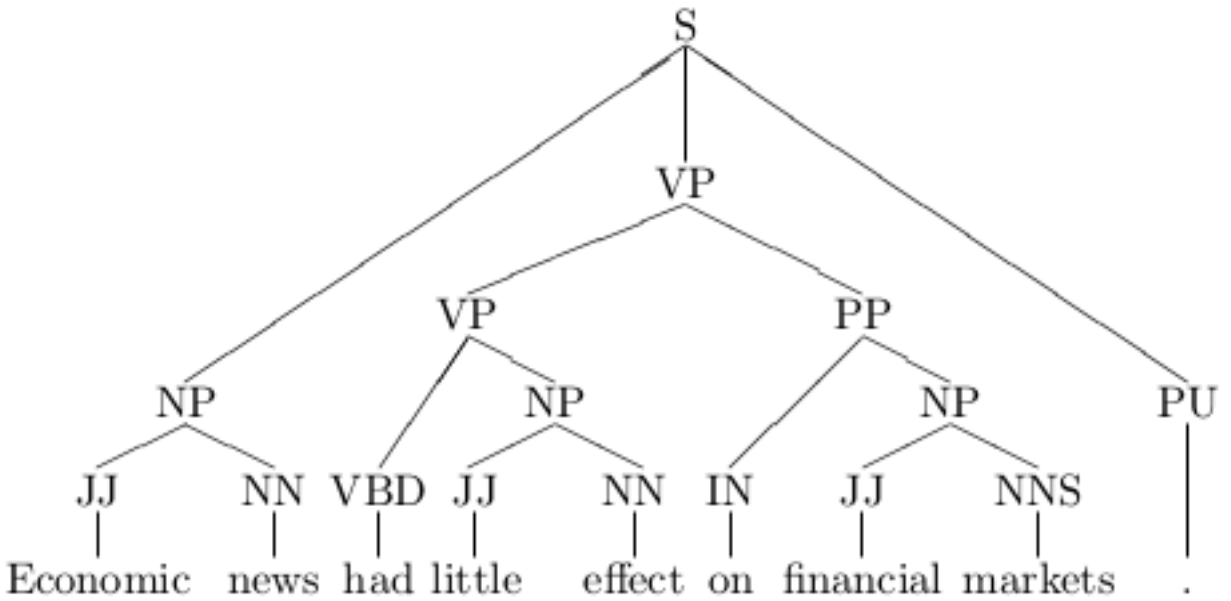
Parsing Model – Tree 2

S	→	NP VP .	1.00	JJ	→	Economic	0.33
VP	→	VP PP	0.33	JJ	→	little	0.33
VP	→	VBD NP	0.67	JJ	→	financial	0.33
NP	→	NP PP	0.14	NN	→	news	0.5
NP	→	JJ NN	0.57	NN	→	effect	0.5
NP	→	JJ NNS	0.29	NNS	→	markets	1.0
PP	→	IN NP	1.0	VBD	→	had	1.0
PU	→	.	1.0	IN	→	on	1.0



Parsing Model – Tree 2

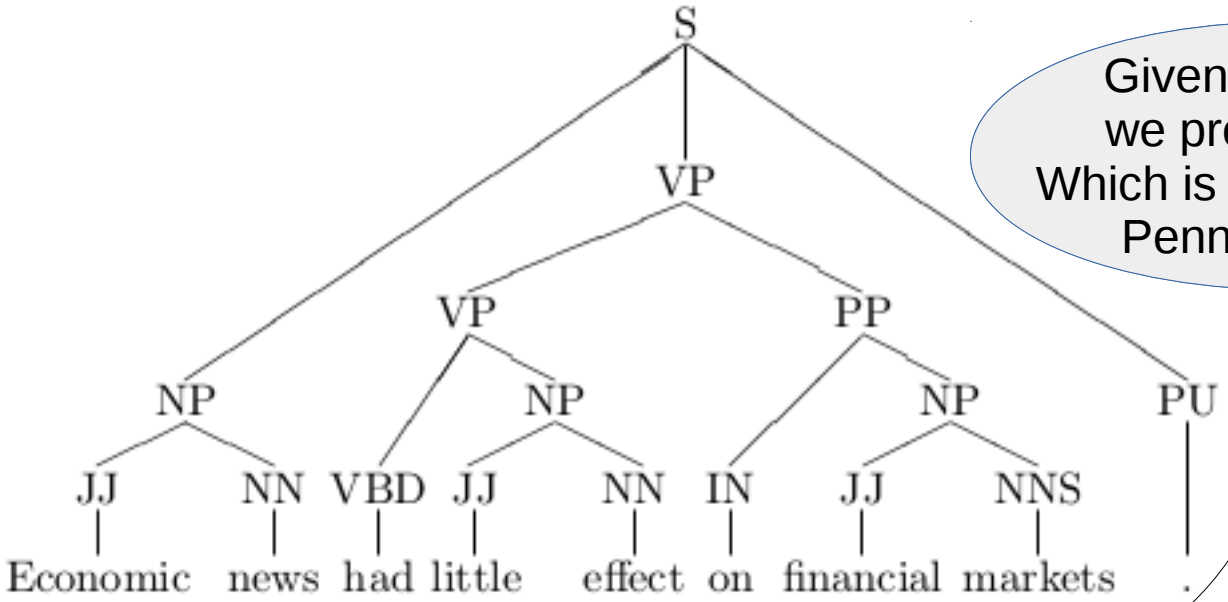
S	→	NP VP .	1.00	JJ	→	Economic	0.33
VP	→	VP PP	0.33	JJ	→	little	0.33
VP	→	VBD NP	0.67	JJ	→	financial	0.33
NP	→	NP PP	0.14	NN	→	news	0.5
NP	→	JJ NN	0.57	NN	→	effect	0.5
NP	→	JJ NNS	0.29	NNS	→	markets	1.0
PP	→	IN NP	1.0	VBD	→	had	1.0
PU	→	.	1.0	IN	→	on	1.0



The probability of this tree T(y), **p=0.0001871**

Parsing Model – Tree 2

S	→	NP VP .	1.00	JJ	→	Economic	0.33
VP	→	VP PP	0.33	JJ	→	little	0.33
VP	→	VBD NP	0.67	JJ	→	financial	0.33
NP	→	NP PP	0.14	NN	→	news	0.5
NP	→	JJ NN	0.57	NN	→	effect	0.5
NP	→	JJ NNS	0.29	NNS	→	markets	1.0
PP	→	IN NP	1.0	VBD	→	had	1.0
PU	→	.	1.0	IN	→	on	1.0



Given this PCFG,
we prefer this tree
Which is incorrect in the
Penn treebank...

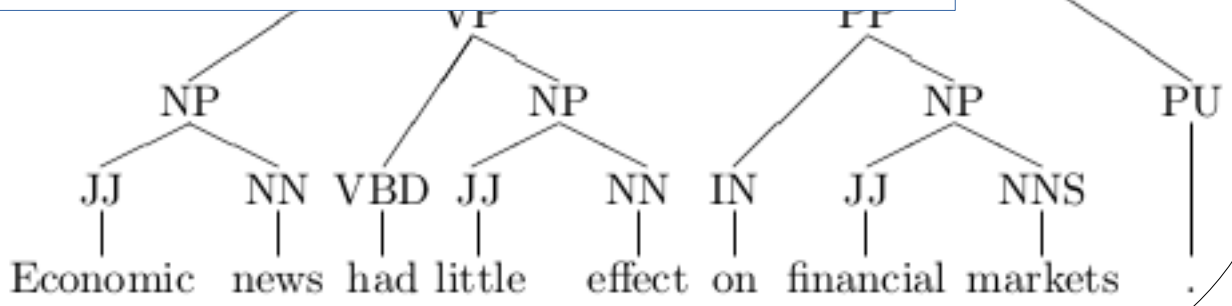
The probability of this tree T(y), **p=0.0001871 vs p=0.0000794**

Parsing Model – Tree 2

This goes in the idea that the optimal interpretation given by a parsing model may differ and normally it will, to the one given by humans=Penn treebank.

onomic	0.33
tle	0.33
ancial	0.33
ws	0.5
ect	0.5
arkets	1.0
d	1.0
	1.0

Given this PCFG, we prefer this tree Which is incorrect in the Penn treebank...



The probability of this tree T(y), **p=0.0001871 vs p=0.0000794**

Parsing Algorithms

- The parsing or decoding problem for a PCFG model is given
  - a specific grammar G
  - an input sentence x,

  1- to compute the set GEN(x) of candidate representations and

  2- to score each candidate by the probability P(y), as defined by the grammar.

Parsing Algorithms

1- to compute the set GEN(x) of candidate representations and

- Simply the parsing problem for CFGs
- Many of the standard algorithms for CFGs that we have studied have a straightforward extension that computes probabilities of trees in the same process:
  - CKY.
  - Earley's.
  - And others.

Parsing Algorithms

  - CKY.
  - Earley's.
  - And others.
- These algorithms are based on dynamic programming, which makes it possible to compute the probability of a substructure (subtree) at the time when it is being composed of smaller substructures,
- We can use Viterbi search to find the highest scoring parse tree in $O(n^3 \cdot |G|)$ time
  - n is the length of the input sentence.
  - |G| is the size of the grammar.

Parsing Algorithms

...

This means that the model defines a complete ranking over all the candidate analyses in GEN(x), but they only compute the best parsing analysis they can find. In other words, they solve:

$$y^* = \underset{y \in GEN(x)}{argmax}\ EVAL(X,Y)$$

The inference is exact: the analysis returned by the parser is guaranteed to be the most probable analysis according to the model.

Learning with a Treebank

- The learning problem for PCFGs is to learn a grammar from a sample of sentences $X = \{x^1, \ldots, x^m\}$.

- If the sentences are annotated with their correct parse trees: the method is to extract a treebank grammar (Charniak, 1996) where the CFG contains all and only the symbols, rules to generate the trees in the training set $y = \{y^1, \ldots, y^m\}$.

Learning with a Treebank

- This is an example of *supervised learning*, since we require the inputs in the training corpus to be labeled with their correct outputs.

- Treebank grammars are guaranteed to be both proper and consistent.
- Learning and decoding is fairly simple and efficient.

Learning with a Treebank

- This is an example of *supervised learning*, since we require the inputs in the training cor utputs.
- Tre be both pro
- Lea nd effi

Reading off the Penn treebank and just simply do what I explained so far, you should be able to get a parser that gets 75.x F1 score.

The state-of-the-art nowadays is higher than 90.

We will later see how we can approach this 90 F1 score.

Learning without a Treebank

- If our training corpus is not a treebank but the CFG is given.
- We need to use Expectation-Maximization (EM) to learn Q = the rule probabilities.

- EM is an approximate method for maximum likelihood estimation,
  - Instead of maximizing the joint likelihood of inputs and outputs, as in a treebank grammar.
  - It attempts to maximize the marginal likelihood of inputs

    (by summing over all outputs for a given input)

Learning without a Treebank

- The basic structure of EM (applied to CFG) is:

  (1) Guess a probability qi for each rule $p_i \in P$.

  (2) Repeat until convergence:

  - E-step: Compute the expected count $f(p_i)$ of each rule $p_i \in P$.

$$f(pi) = \sum_{j=1} \sum_{y \in GEN(x^j)} P(y \mid x^j, P) \cdot COUNT(i, y)$$

  - M-Step: Reestimate the probability qi of each rule pi to maximize the marginal likelihood given expected counts.

$$Qi = f(pi) / \sum_{pj \in P \; lhs(pj) = lhs(pi)} f(pi)$$

Learning without a Treebank

- The tricky part is to compute the expected rule counts in an efficient way, which can be done using the inside-outside algorithm (see recitation next Friday)

- EM training with the inside-outside algorithm was used in early work on PCFG parsing to estimate the probabilistic parameters of hand-crafted context-free grammars from raw text corpora.

- PCFGs induced in this way are guaranteed to be proper and consistent.

Learning without a Treebank

- The tricky part is to compute the expected rule counts in an efficient way, which can be done using the inside-outside algorithm (see recitation next Friday)
- EM as used in early work listic parameters of h w text corpora.
- PCF be proper and consistent.

EM is not the only thing you can do... there are bayesian models, spectral learning, etc.