COSC 6336 Natural Language Processing
Statistical Parsing

Ch. 14 J&M

Content in this set was adapted from Ray Mooney's course

Statistical Parsing

- Statistical parsing uses a probabilistic model of syntax in order to assign probabilities to each parse tree.
- Provides principled approach to resolving syntactic ambiguity.
- Allows supervised learning of parsers from tree-banks provided by human linguists.
- Also allows unsupervised learning of parsers from unannotated text, but the accuracy of such parsers has been limited.

Probabilistic Context Free Grammar (PCFG)

- A PCFG is a CFG where each production has a probability subject to:

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

- String generation is now probabilistic where production probabilities are used to **non-deterministically** select a production for rewriting a given non-terminal.

Simple PCFG for English

Grammar	Prob		
S → NP VP	0.8	+	1.0
S → Aux NP VP	0.1		
S → VP	0.1		
NP → Pronoun	0.2	+	1.0
NP → Proper-Noun	0.2		
NP → Det Nominal	0.6		
Nominal → Noun	0.3	+	1.0
Nominal → Nominal Noun	0.2		
Nominal → Nominal PP	0.5		
VP → Verb	0.2	+	1.0
VP → Verb NP	0.5		
VP → VP PP	0.3		
PP → Prep NP	1.0		

Lexicon

Det → the | a | that | this
0.6 0.2 0.1 0.1

Noun → book | flight | meal | money
0.1 0.5 0.2 0.2

Verb → book | include | prefer
0.5 0.2 0.3

Pronoun → I | he | she | me
0.5 0.1 0.1 0.3

Proper-Noun → Houston | NWA
0.8 0.2

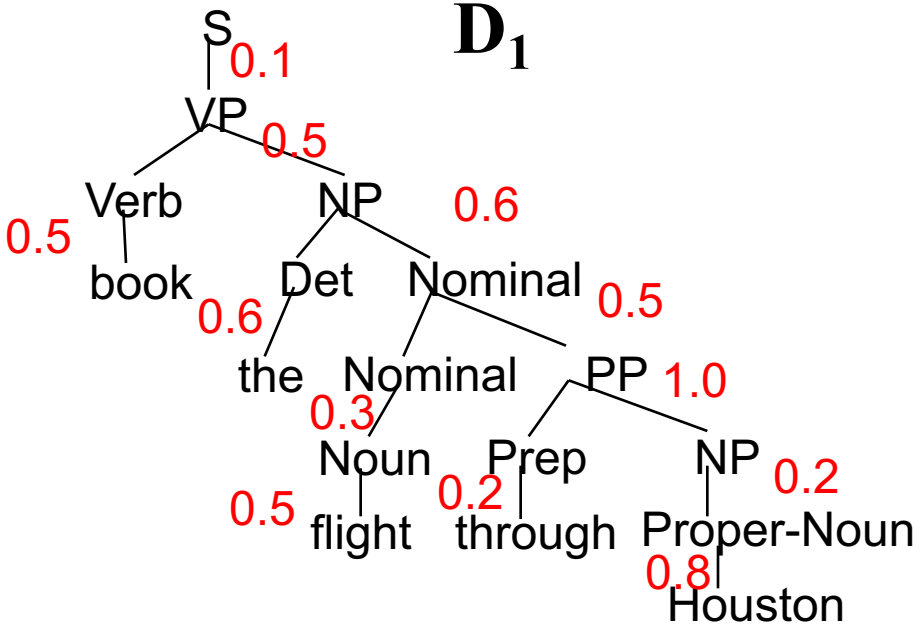
Aux → does
1.0

Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2

Sentence Probability

Probability of derivation is the product of the probabilities of its productions.

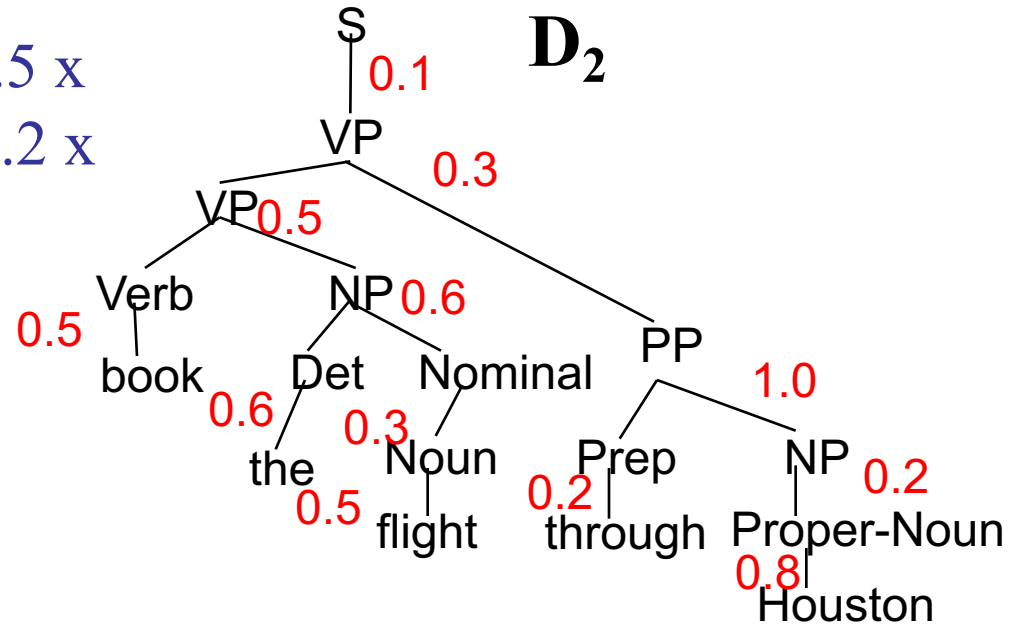
$$P(D_1) = 0.1 \times 0.5 \times 0.5 \times 0.6 \times 0.6 \times 0.5 \times 0.3 \times 1.0 \times 0.2 \times 0.2 \times 0.5 \times 0.8 = 0.0000216$$



Syntactic Disambiguation

- Resolve ambiguity by picking most probable parse tree.

$$P(D_2) = 0.1 \times 0.3 \times 0.5 \times 0.6 \times 0.5 \times 0.6 \times 0.3 \times 1.0 \times 0.5 \times 0.2 \times 0.2 \times 0.8 = 0.00001296$$



Sentence Probability

- Probability of a sentence is the sum of the probabilities of all of its derivations.

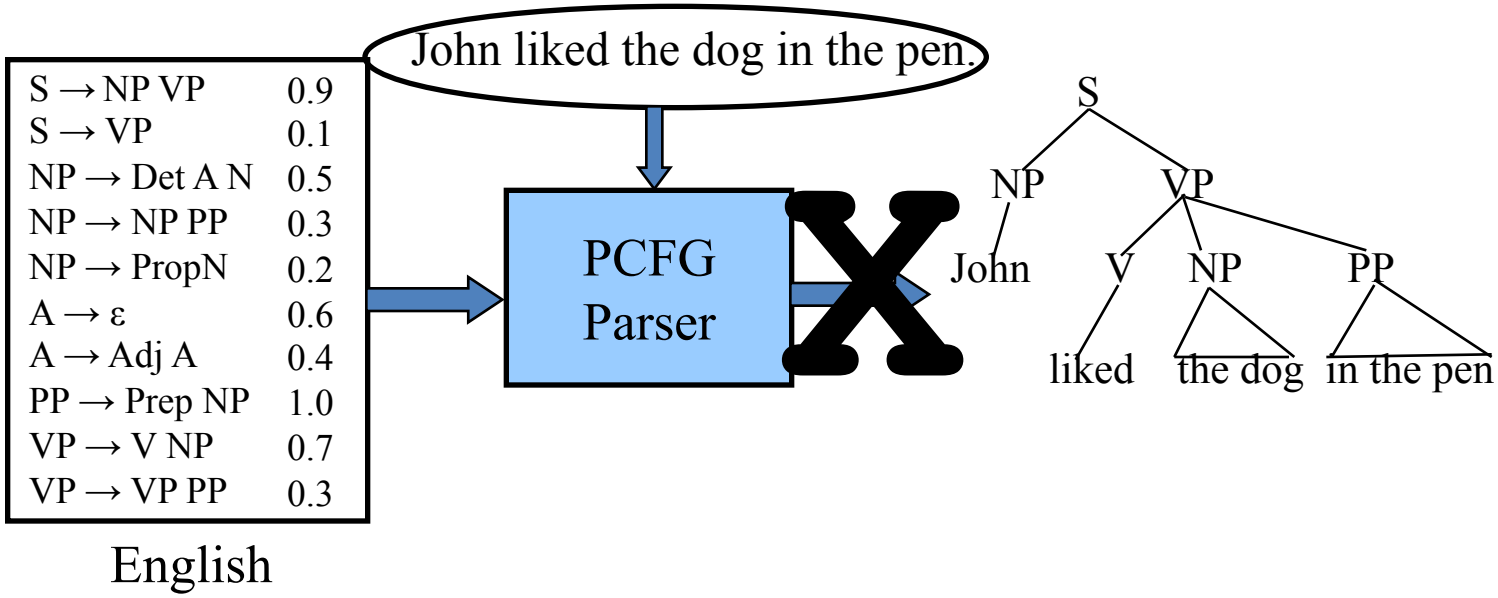
$$P(\text{“book the flight through Houston”}) =$$
$$P(D_1) + P(D_2) = 0.0000216 + 0.00001296$$
$$= 0.00003456$$

Three Useful PCFG Tasks

- Observation likelihood: To classify and order sentences.
- Most likely derivation: To determine the most likely parse tree for a sentence.
- Maximum likelihood training: To train a PCFG to fit empirical training data.

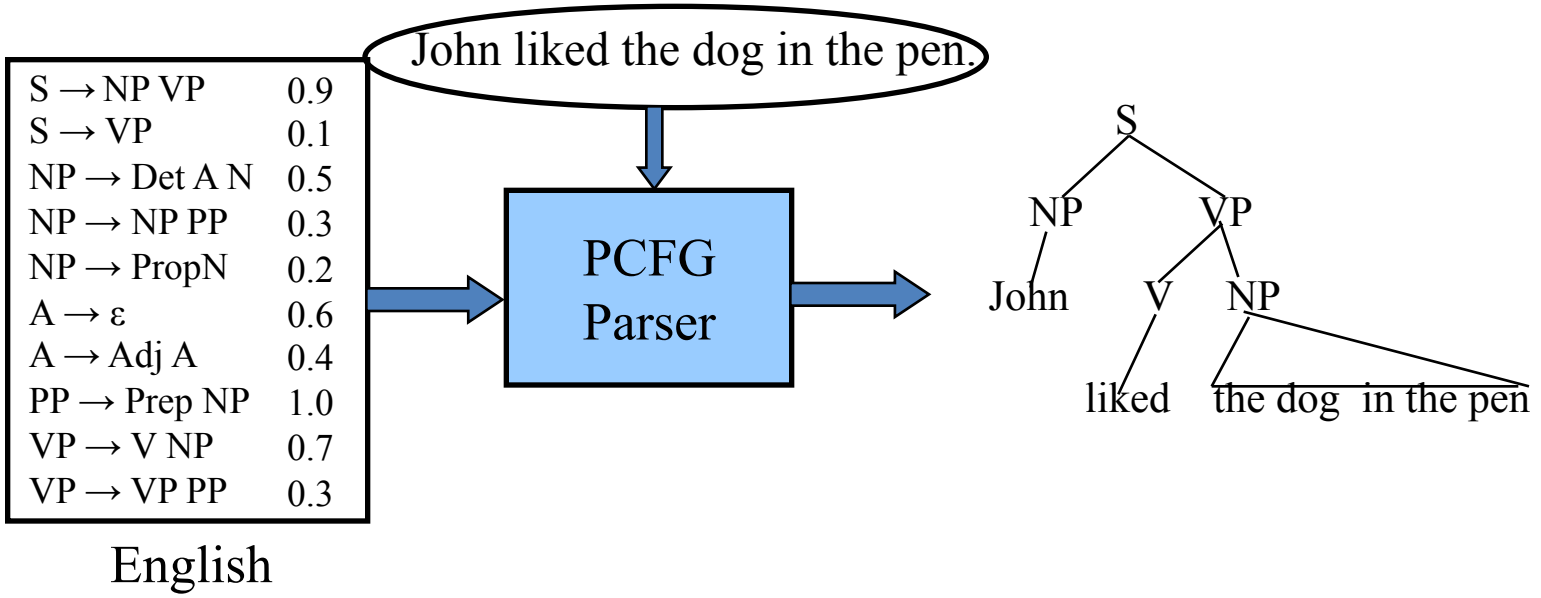
PCFG: Most Likely Derivation

- There is an analog to the Viterbi algorithm to efficiently determine the most probable derivation (parse tree) for a sentence.



PCFG: Most Likely Derivation

- There is an analog to the Viterbi algorithm to efficiently determine the most probable derivation (parse tree) for a sentence.



Probabilistic CKY

- CKY can be modified for PCFG parsing by including in each cell a probability for each non-terminal.
- Cell[*i*,*j*] must retain the *most probable* derivation of each constituent (non-terminal) covering words *i* +1 through *j* together with its associated probability.
- When transforming the grammar to CNF, must set production probabilities to preserve the probability of derivations.

Probabilistic Grammar Conversion

Original Grammar

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	0.1
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	0.2
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Chomsky Normal Form

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book | include | prefer 0.01 0.004 0.006	
S → Verb NP	0.05
S → VP PP	0.03
NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6
Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Probabilistic Grammar Conversion

The lexical entries

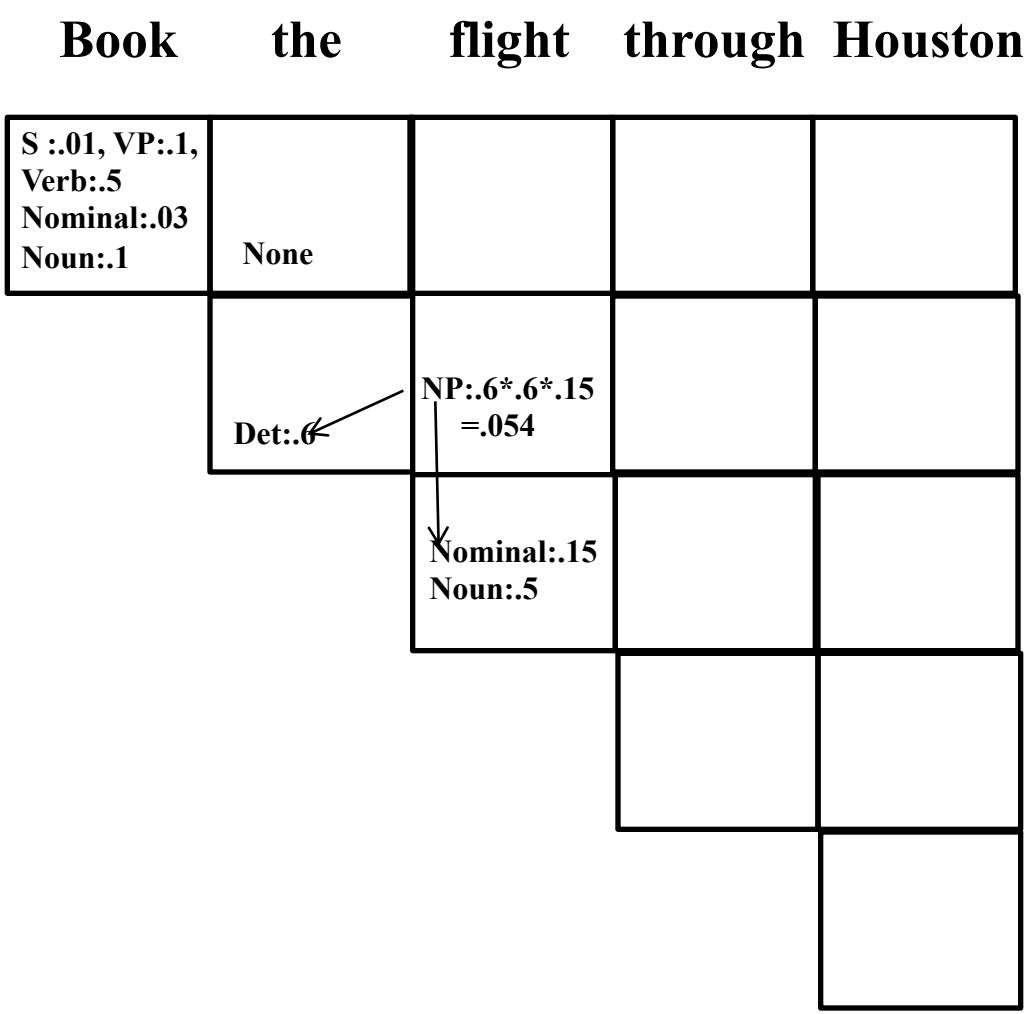
Noun → book | flight | meal | money
0.1 0.5 0.3 0.1

Proper-Noun → Houston | NWA
0.8 0.2

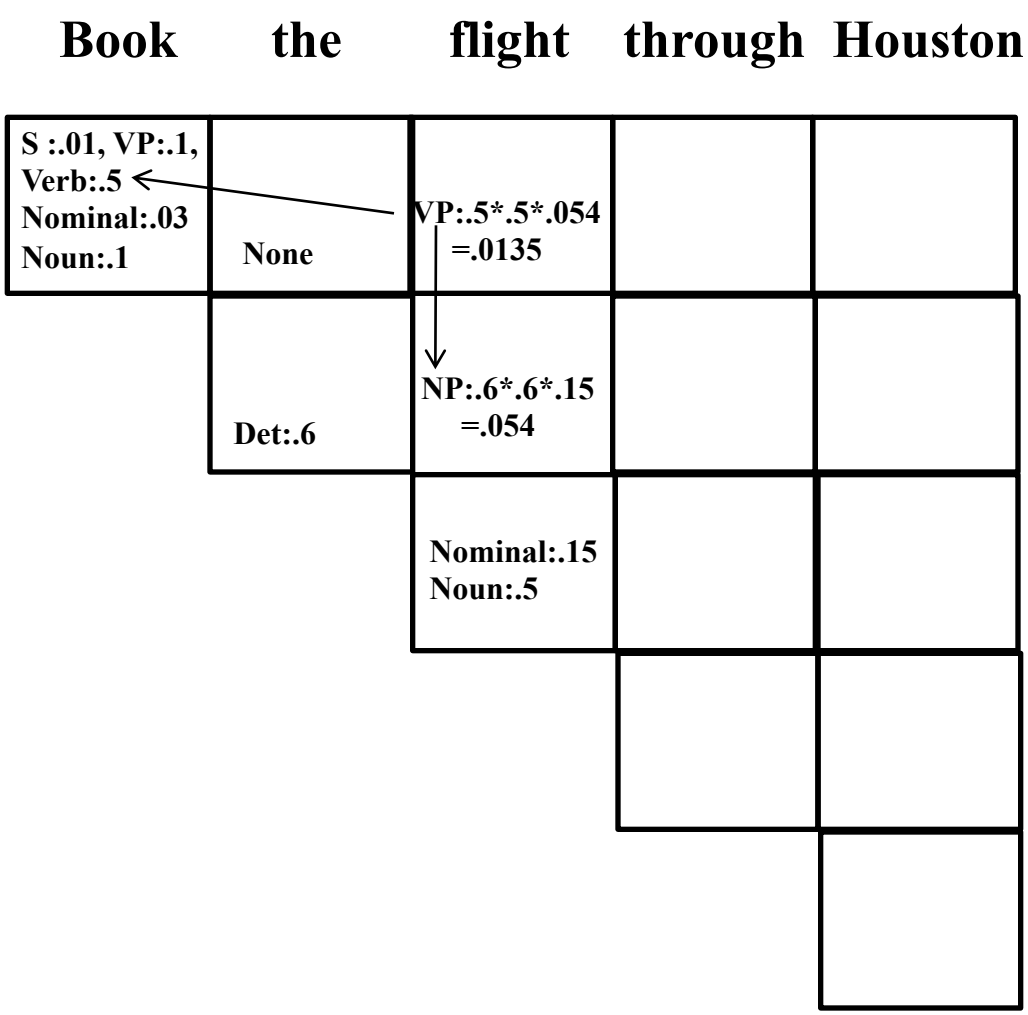
Verb → book | include | prefer
0.5 0.2 0.3

Det → the | a
0.6 0.4

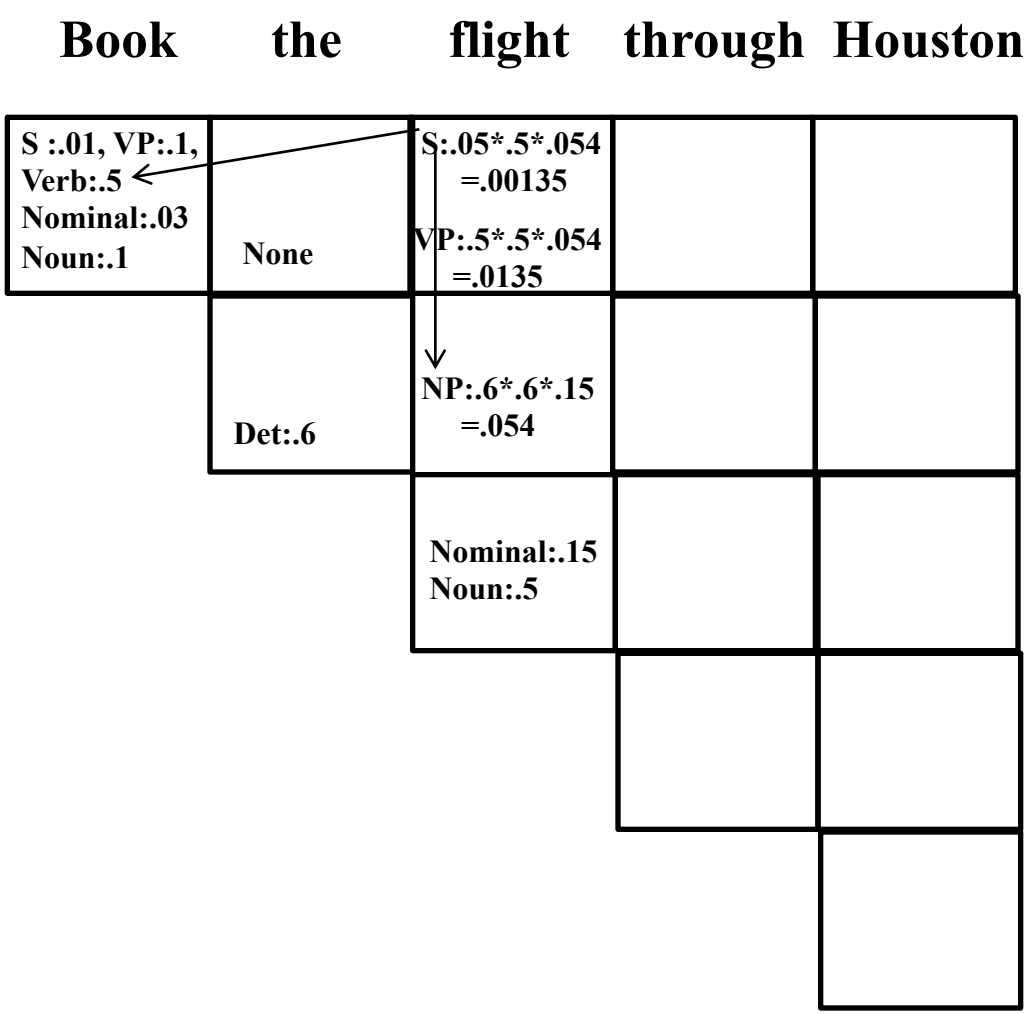
Probabilistic CKY Parser



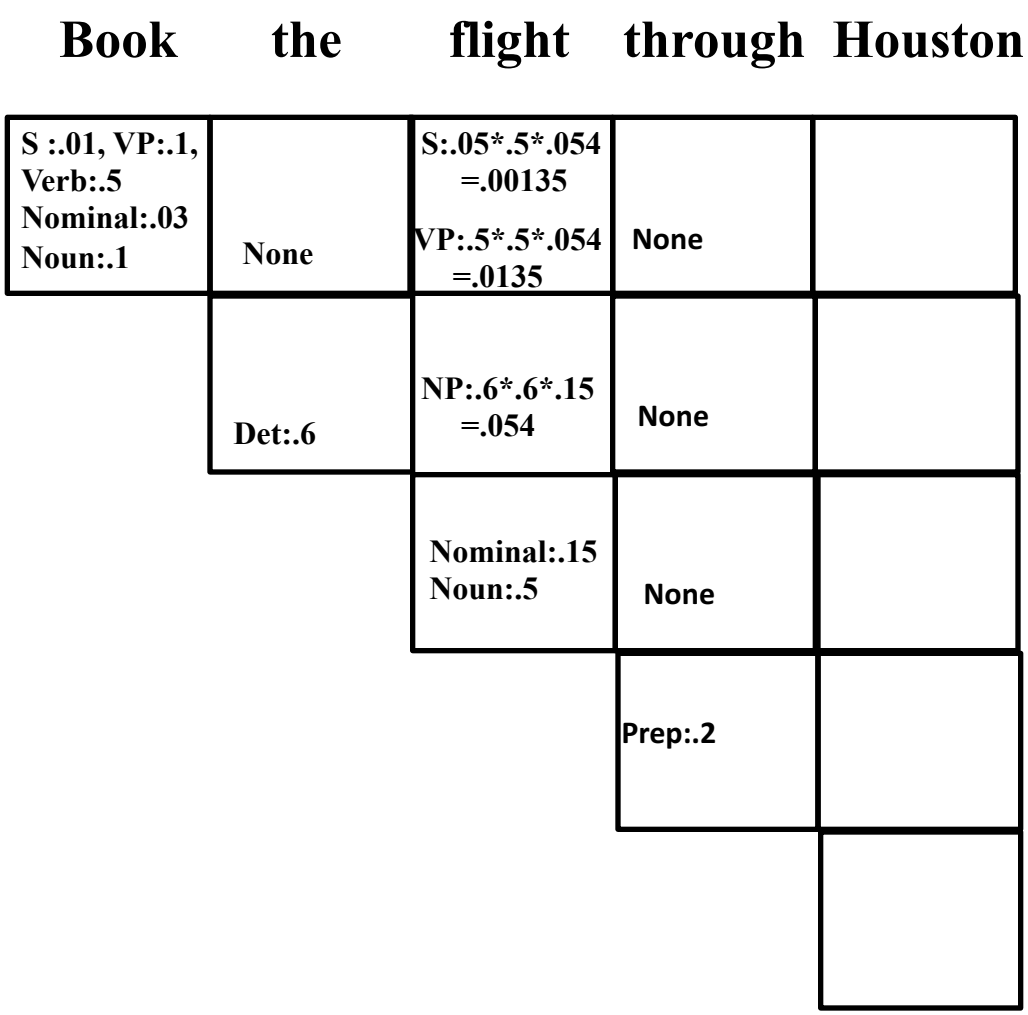
Probabilistic CKY Parser



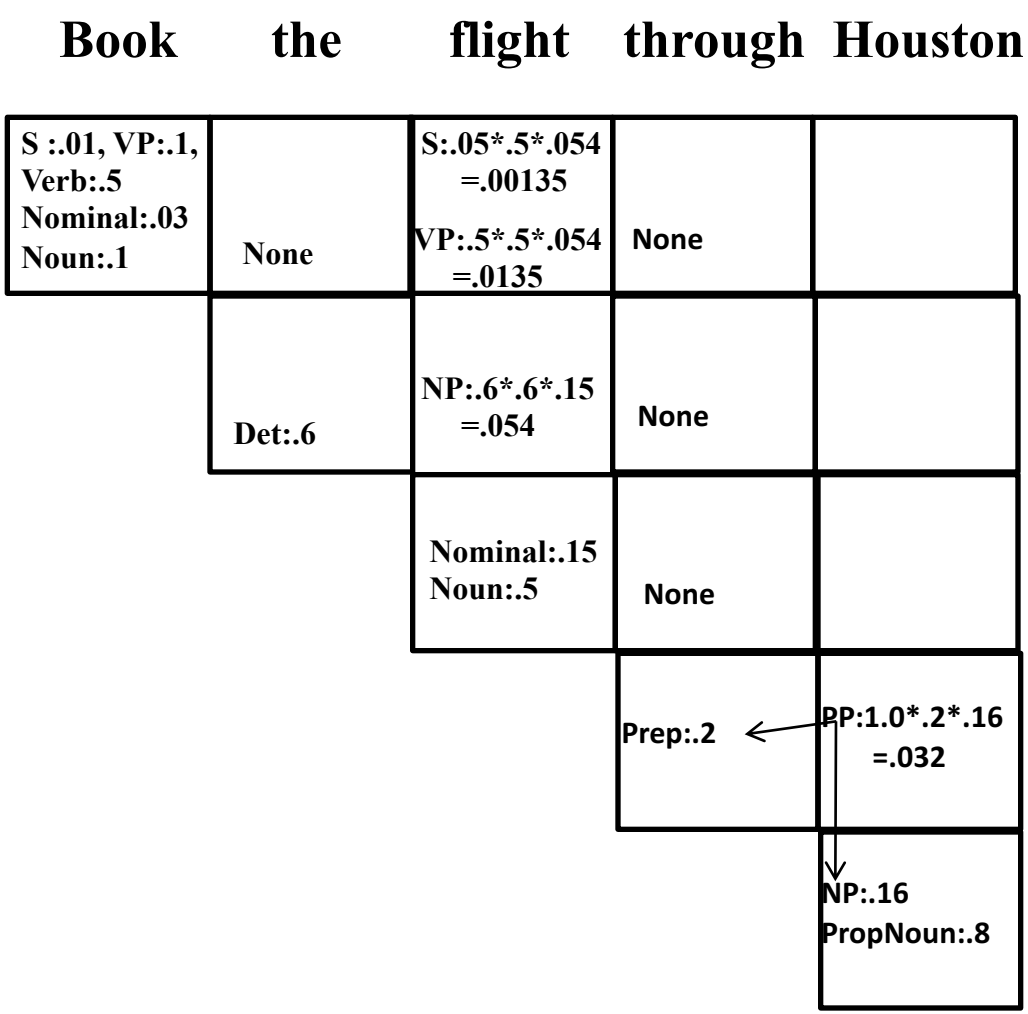
Probabilistic CKY Parser



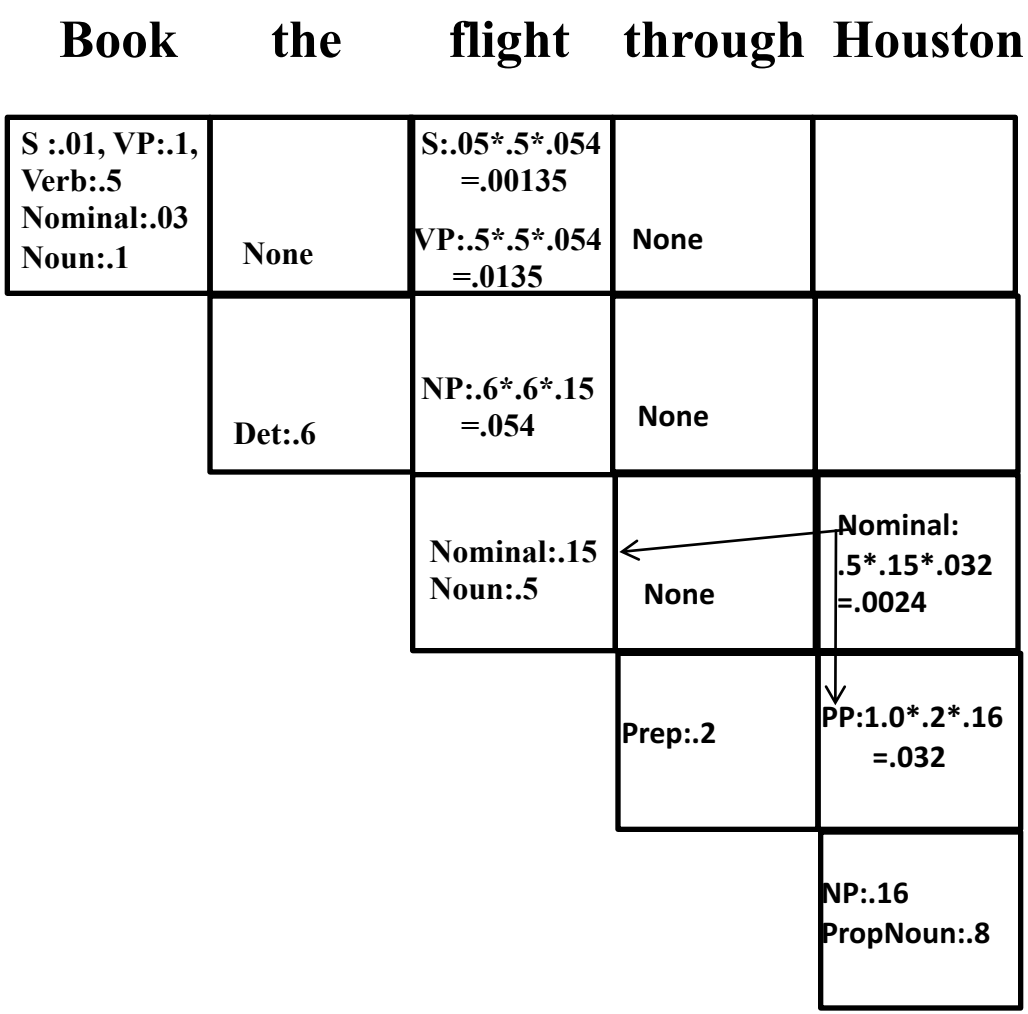
Probabilistic CKY Parser



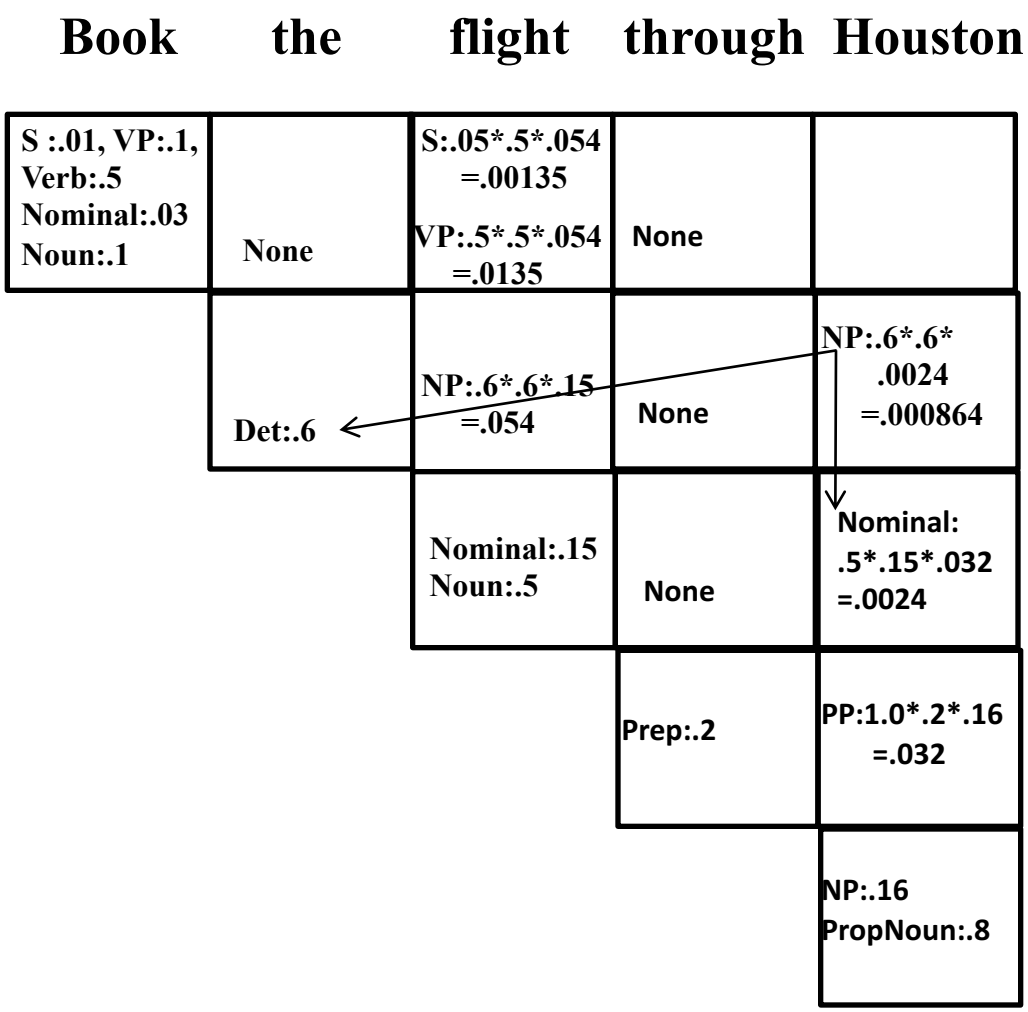
Probabilistic CKY Parser



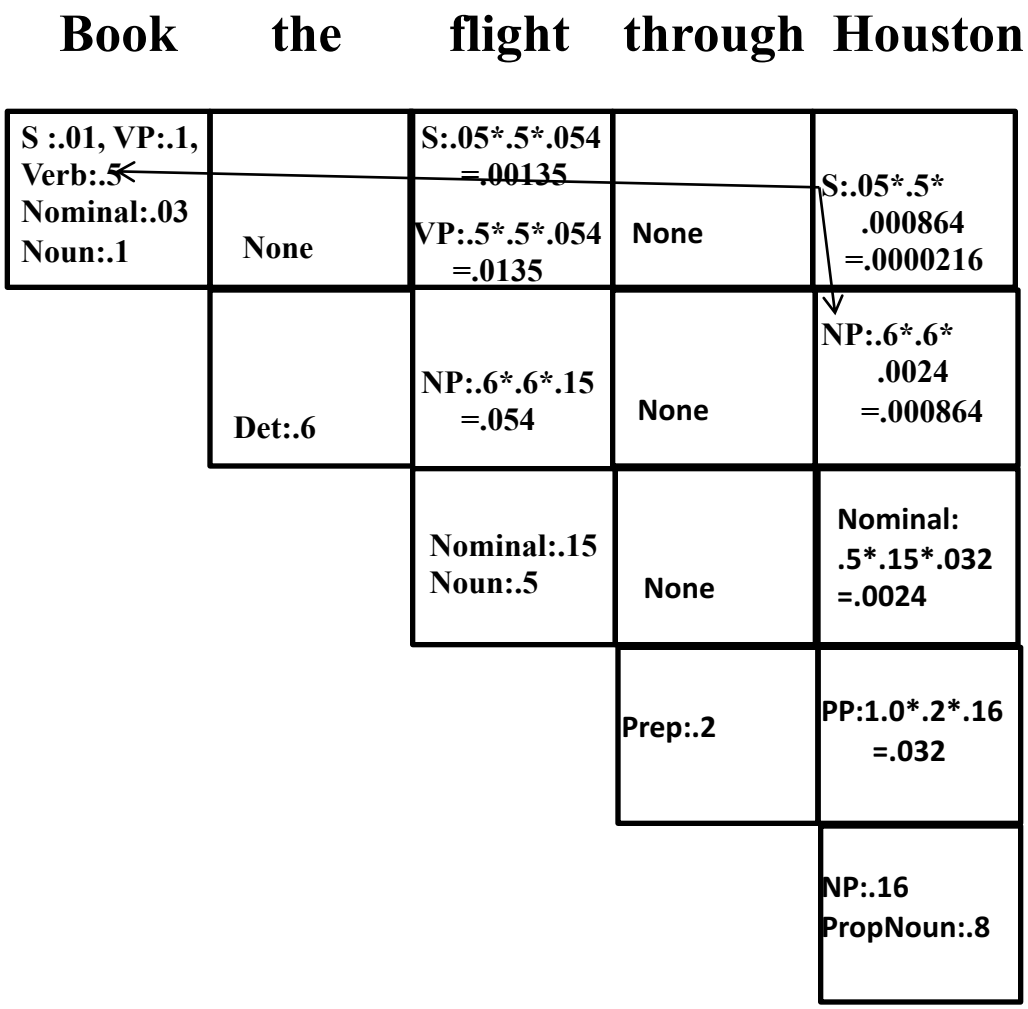
Probabilistic CKY Parser



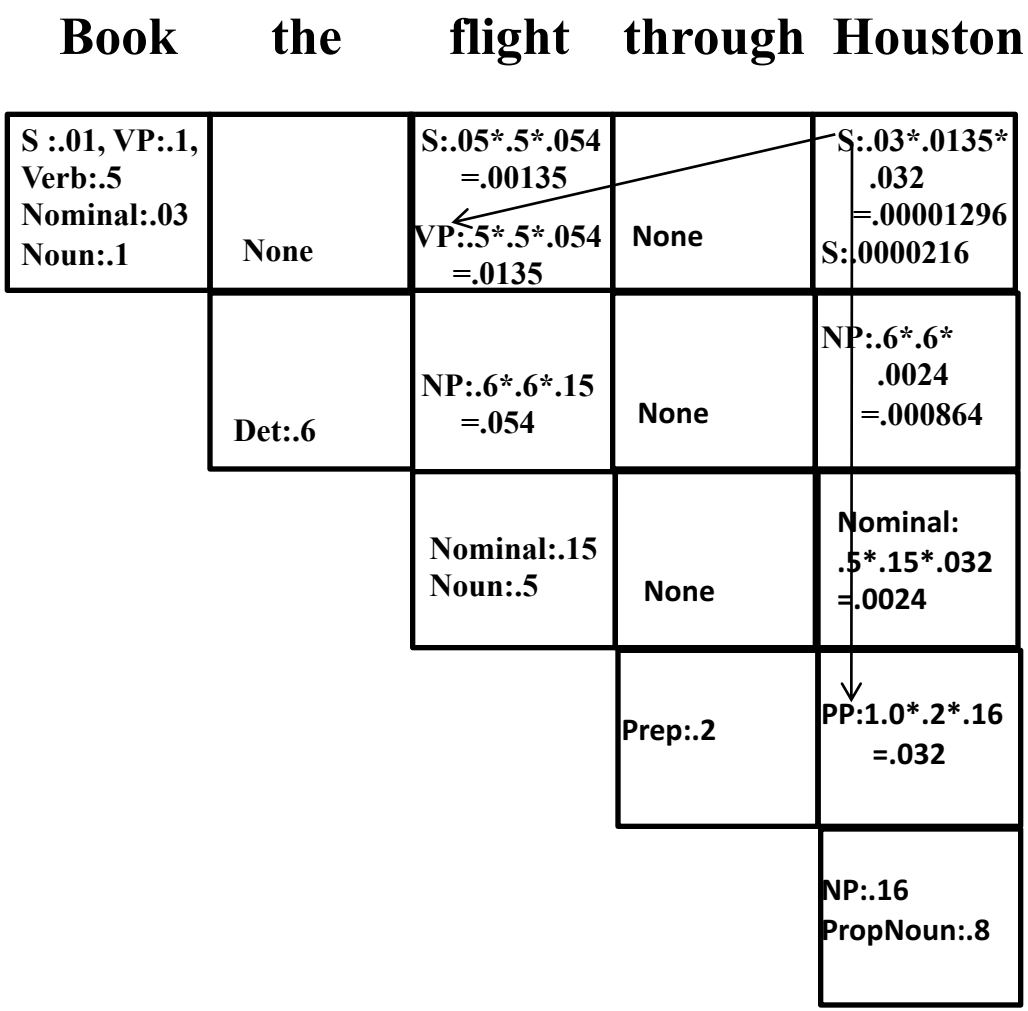
Probabilistic CKY Parser



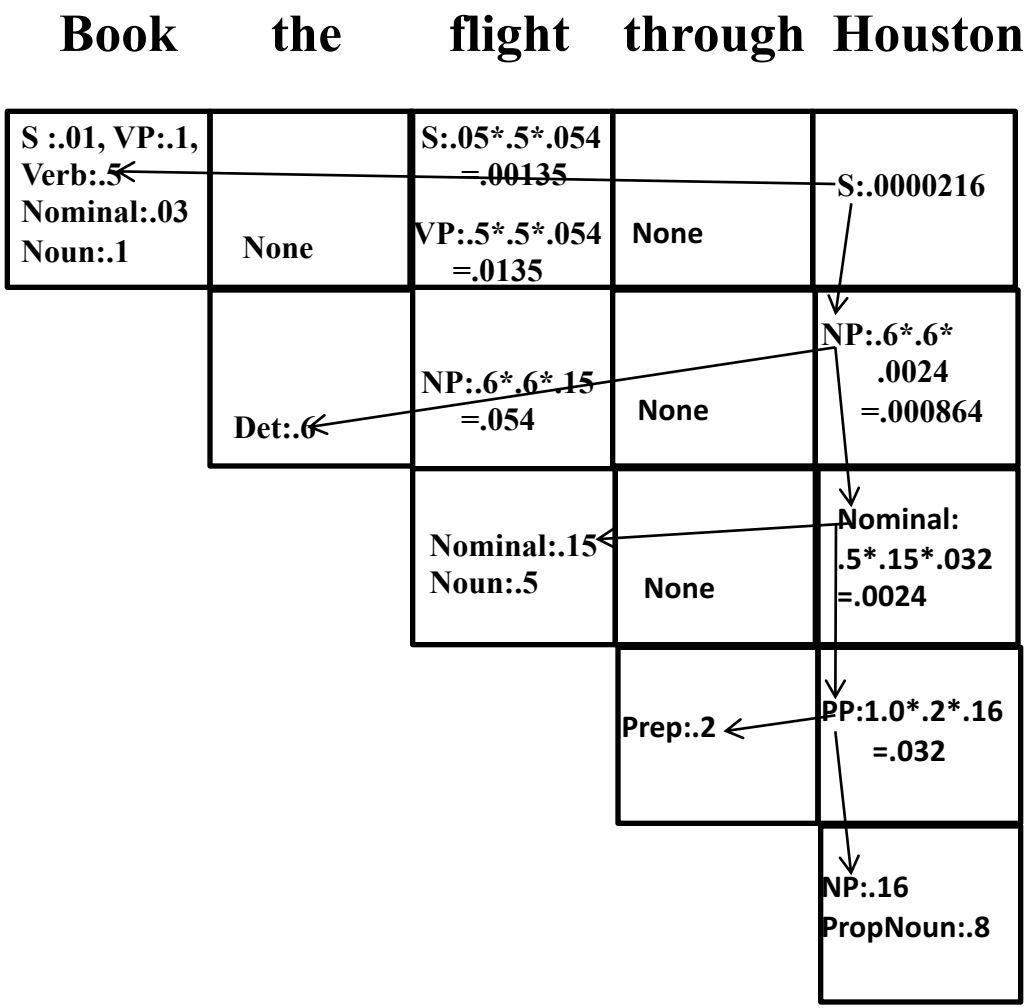
Probabilistic CKY Parser



Probabilistic CKY Parser



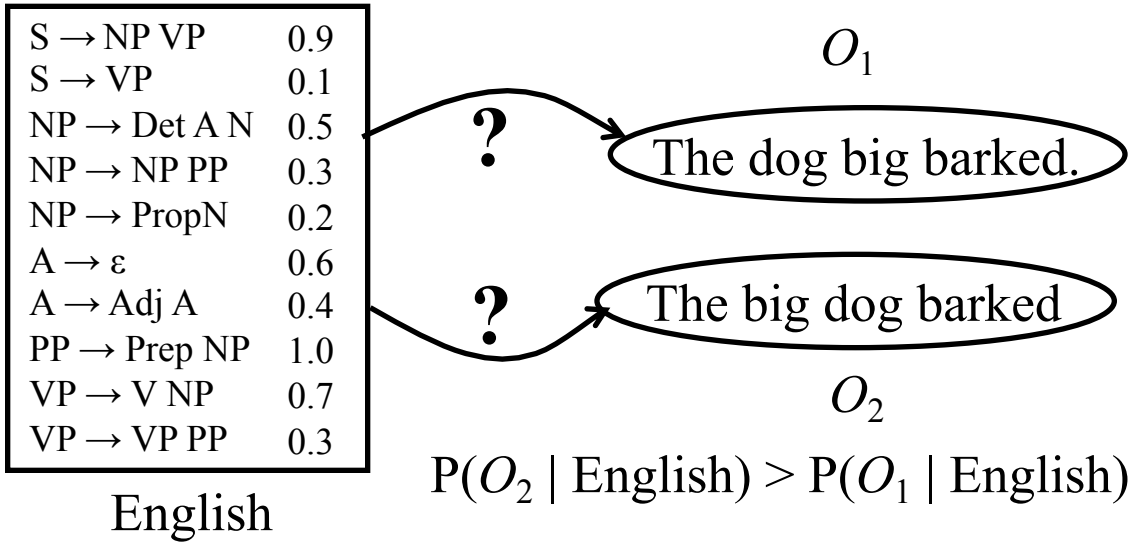
Probabilistic CKY Parser



Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.

PCFG: Observation Likelihood

- There is an analog to Forward algorithm for HMMs called the Inside algorithm for efficiently determining how likely a string is to be produced by a PCFG.
- Can use a PCFG as a language model to choose between alternative sentences for speech recognition or machine translation.

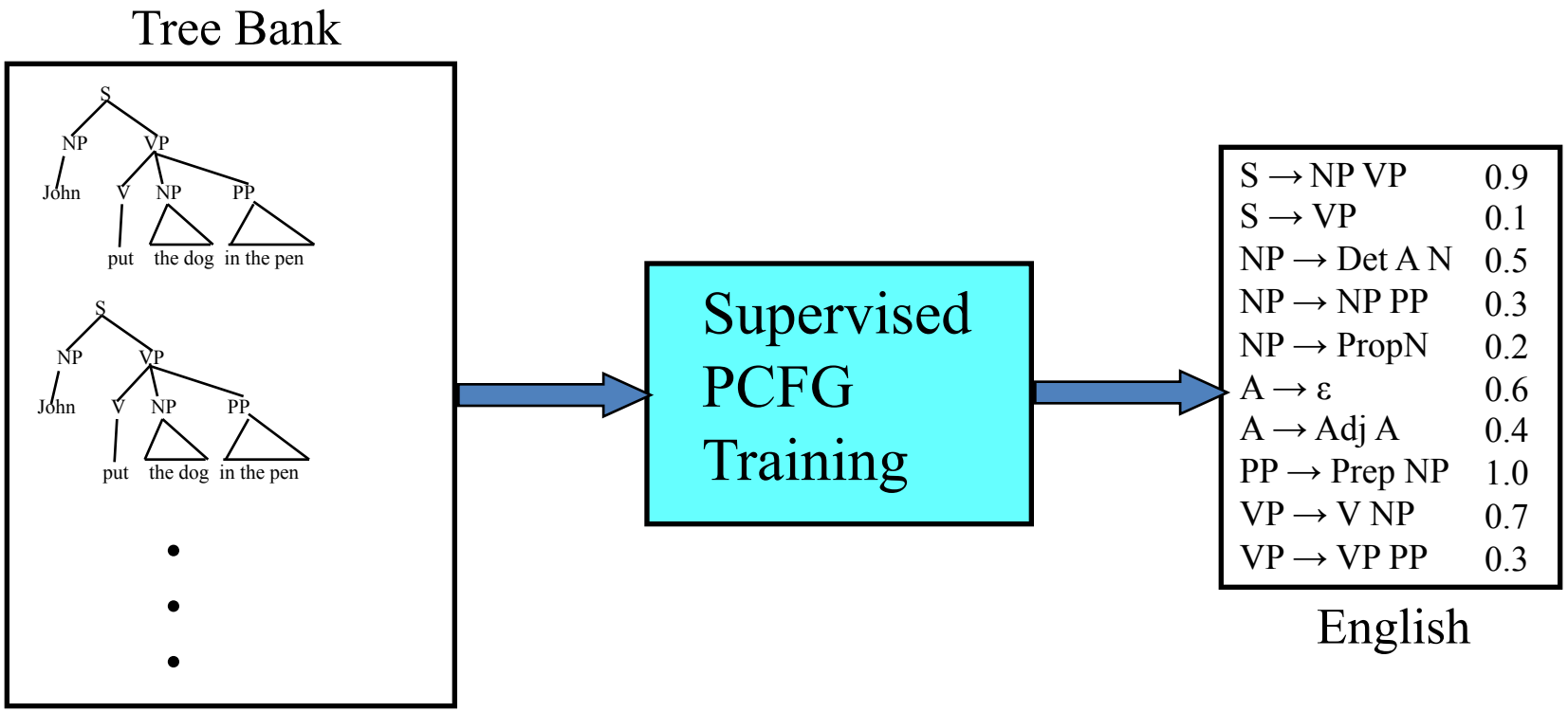


Inside Algorithm

- Use CKY probabilistic parsing algorithm but combine probabilities of multiple derivations of any constituent using **addition** instead of **max**.

PCFG: Supervised Training

- If parse trees are provided for training sentences, a grammar and its parameters can all be estimated directly from counts accumulated from the tree-bank (with appropriate smoothing).



Estimating Production Probabilities

- Set of production rules can be taken directly from the set of rewrites in the treebank.
- Parameters can be directly estimated from frequency counts in the treebank.

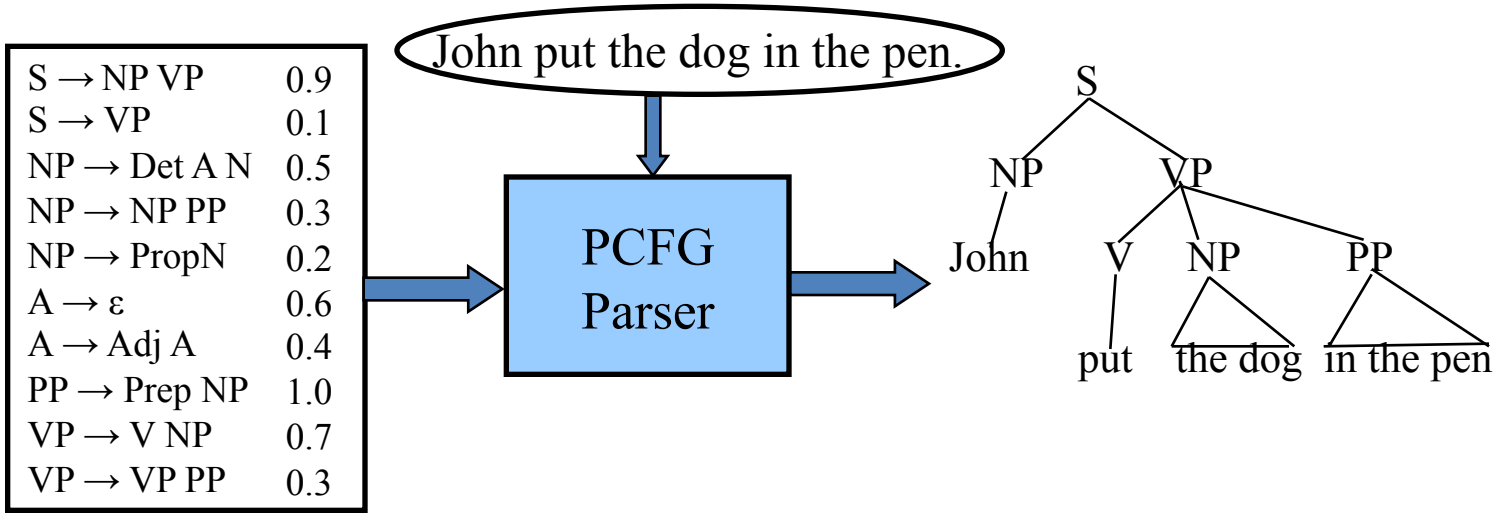
$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{count}(\alpha \rightarrow \gamma)} = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

Vanilla PCFG Limitations

- Lack ability to model relationships across the parse tree.
- Only general structural disambiguation is possible (e.g. prefer to attach PPs to Nominals).
- Consequently, vanilla PCFGs cannot resolve syntactic ambiguities that require semantics to resolve
- In order to work well, PCFGs must be lexicalized (e.g. VP-ate).

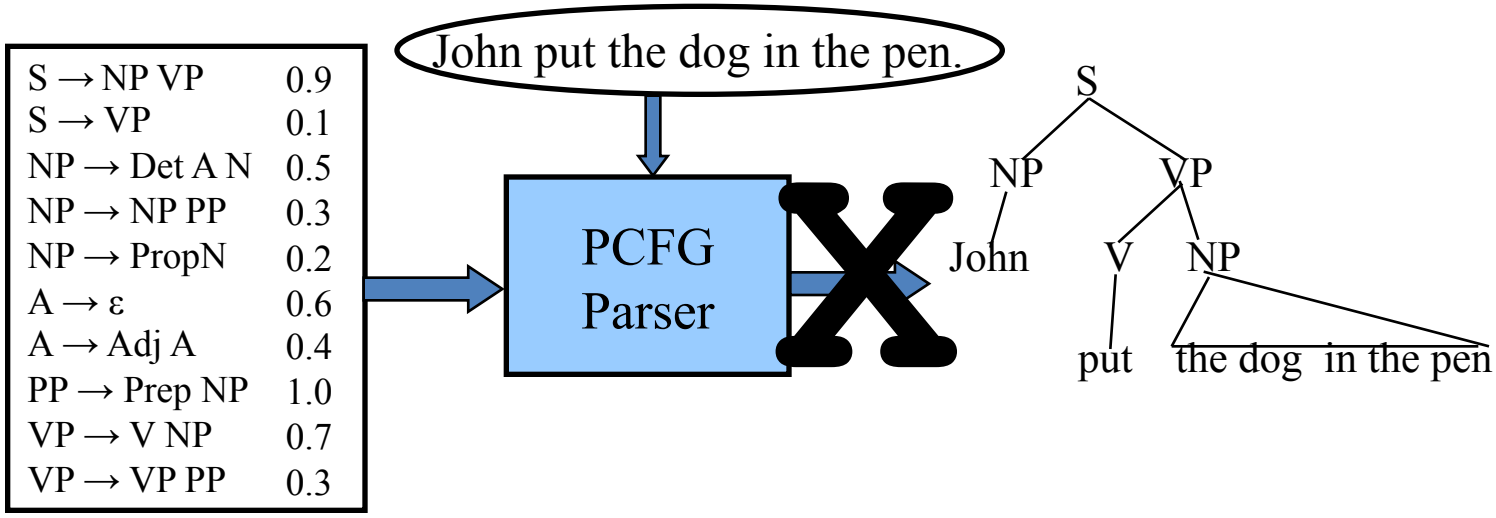
Example of Importance of Lexicalization

- A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG.
- But the desired preference can depend on specific words.



Example of Importance of Lexicalization

- A general preference for attaching PPs to NPs rather than VPs can be learned by a vanilla PCFG.
- But the desired preference can depend on specific words.

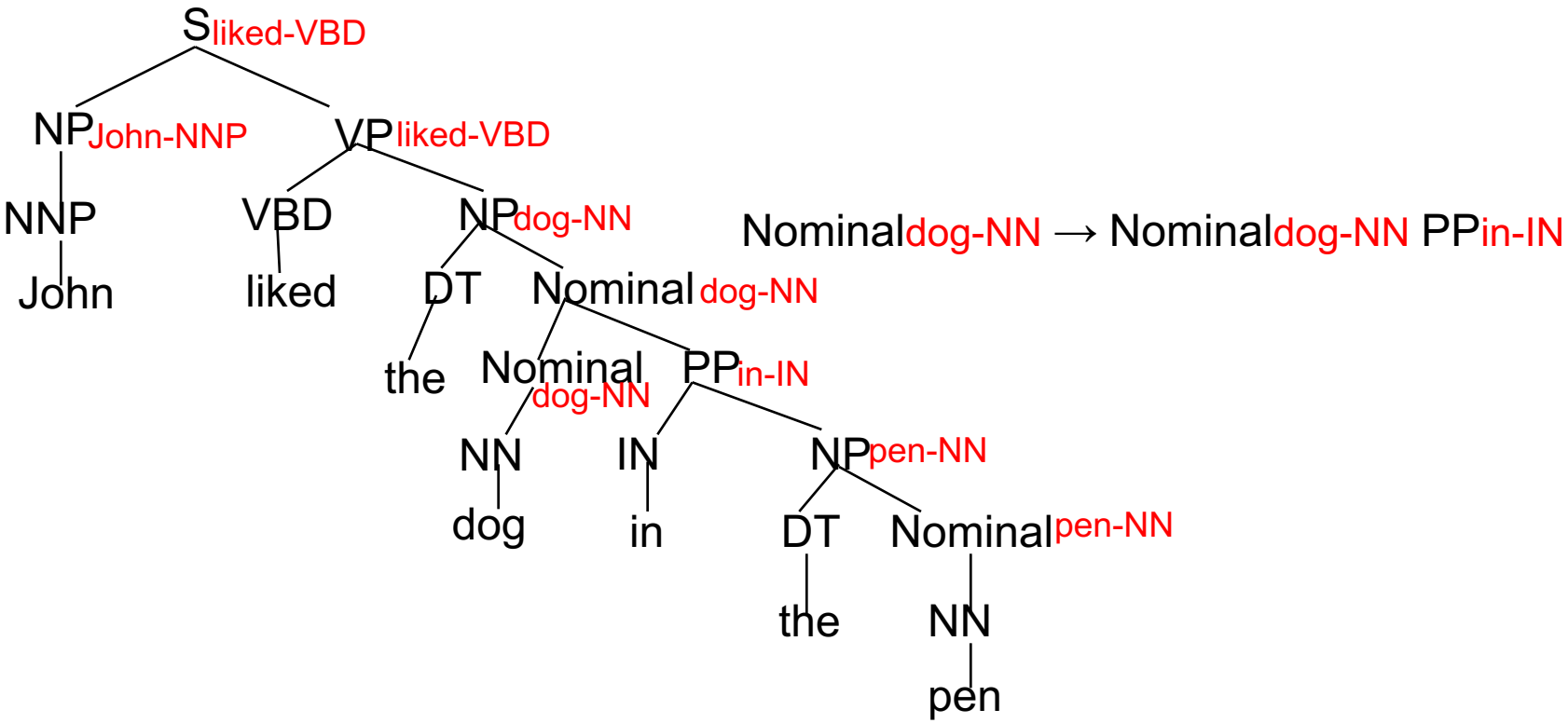


Head Words

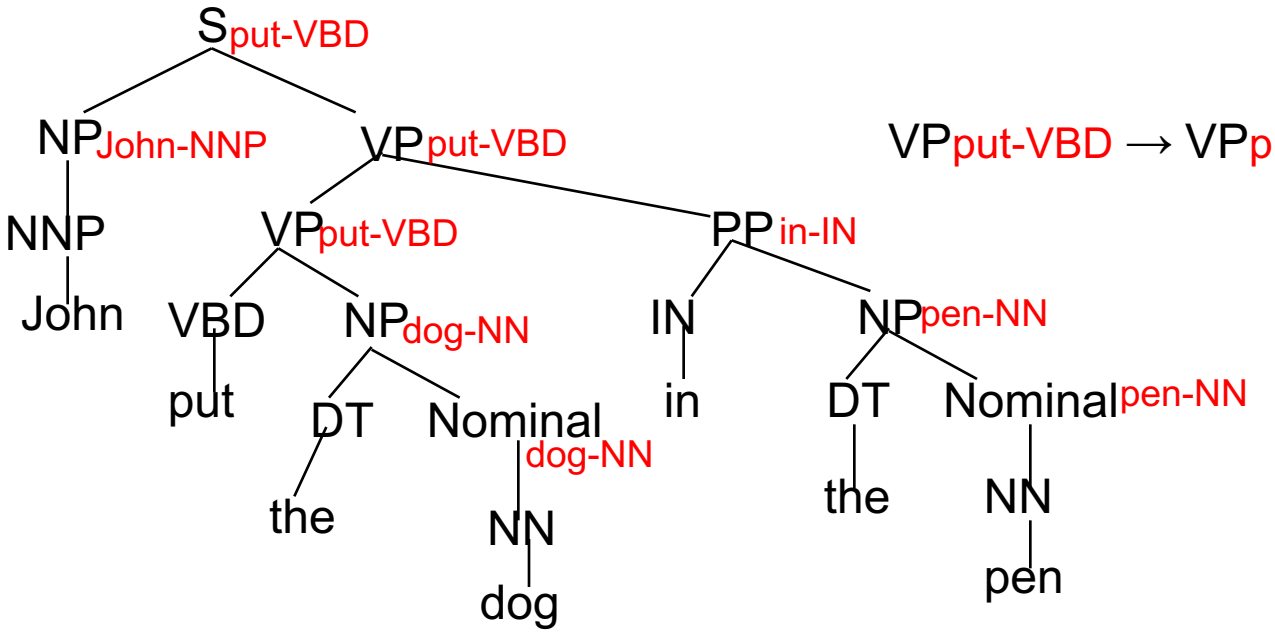
- Syntactic phrases usually have a word in them that is most "central" to the phrase.
- Linguists have defined the concept of a lexical **head** of a phrase.
- Simple rules can identify the head of any phrase by percolating head words up the parse tree.
  - Head of a VP is the main verb
  - Head of an NP is the main noun
  - Head of a PP is the preposition
  - Head of a sentence is the head of its VP

Lexicalized Productions

- Specialized productions can be generated by including the head word and its POS of each non-terminal as part of that non-terminal's symbol.



Lexicalized Productions



$VP_{\text{put-VBD}} \rightarrow VP_{\text{put-VBD}}\ PP_{\text{in-IN}}$

Parameterizing Lexicalized Productions

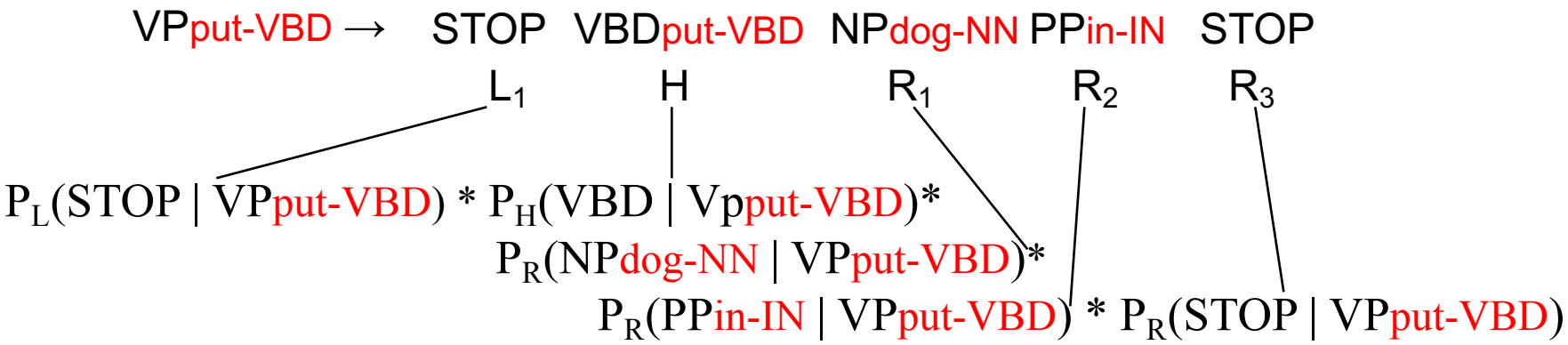
- Accurately estimating parameters on such a large number of very specialized productions could require enormous amounts of treebank data.
- Need some way of estimating parameters for lexicalized productions that makes reasonable independence assumptions so that accurate probabilities for very specific rules can be learned.

Collins' Parser

- Collins' (1999) parser assumes a simple generative model of lexicalized productions.
- Models productions based on context to the left and the right of the head daughter.
  - LHS → $L_n L_{n-1} \ldots L_1 H\ R_1 \ldots R_{m-1} R_m$
- First generate the head (H) and then repeatedly generate left (L_i) and right (R_i) context symbols until the symbol STOP is generated.

Sample Production Generation

VPput-VBD → VBDput-VBD NPdog-NN PPin-IN



Estimating Production Generation Parameters

- Estimate P_H, P_L, and P_R parameters from treebank data.

$$P_R(\text{PP}_{\text{in-IN}} \mid \text{VP}_{\text{put-VBD}}) = \frac{\text{Count}(\text{PP}_{\text{in-IN}} \text{ right of head in a VP}_{\text{put-VBD}} \text{ production})}{\text{Count}(\text{symbol right of head in a VP}_{\text{put-VBD}})}$$

$$P_R(\text{NP}_{\text{dog-NN}} \mid \text{VP}_{\text{put-VBD}}) = \frac{\text{Count}(\text{NP}_{\text{dog-NN}} \text{ right of head in a VP}_{\text{put-VBD}} \text{ production})}{\text{Count}(\text{symbol right of head in a VP}_{\text{put-VBD}})}$$

- Smooth estimates by linearly interpolating with simpler models conditioned on just POS tag or no lexical info.

$$\begin{aligned} \text{smP}_R(\text{PP}_{\text{in-IN}} \mid \text{VP}_{\text{put-VBD}}) = {} & \lambda_1 P_R(\text{PP}_{\text{in-IN}} \mid \text{VP}_{\text{put-VBD}}) \\ & + (1 - \lambda_1)\,(\lambda_2 P_R(\text{PP}_{\text{in-IN}} \mid \text{VP}_{\text{VBD}}) + \\ & (1 - \lambda_2) P_R(\text{PP}_{\text{in-IN}} \mid \text{VP})) \end{aligned}$$

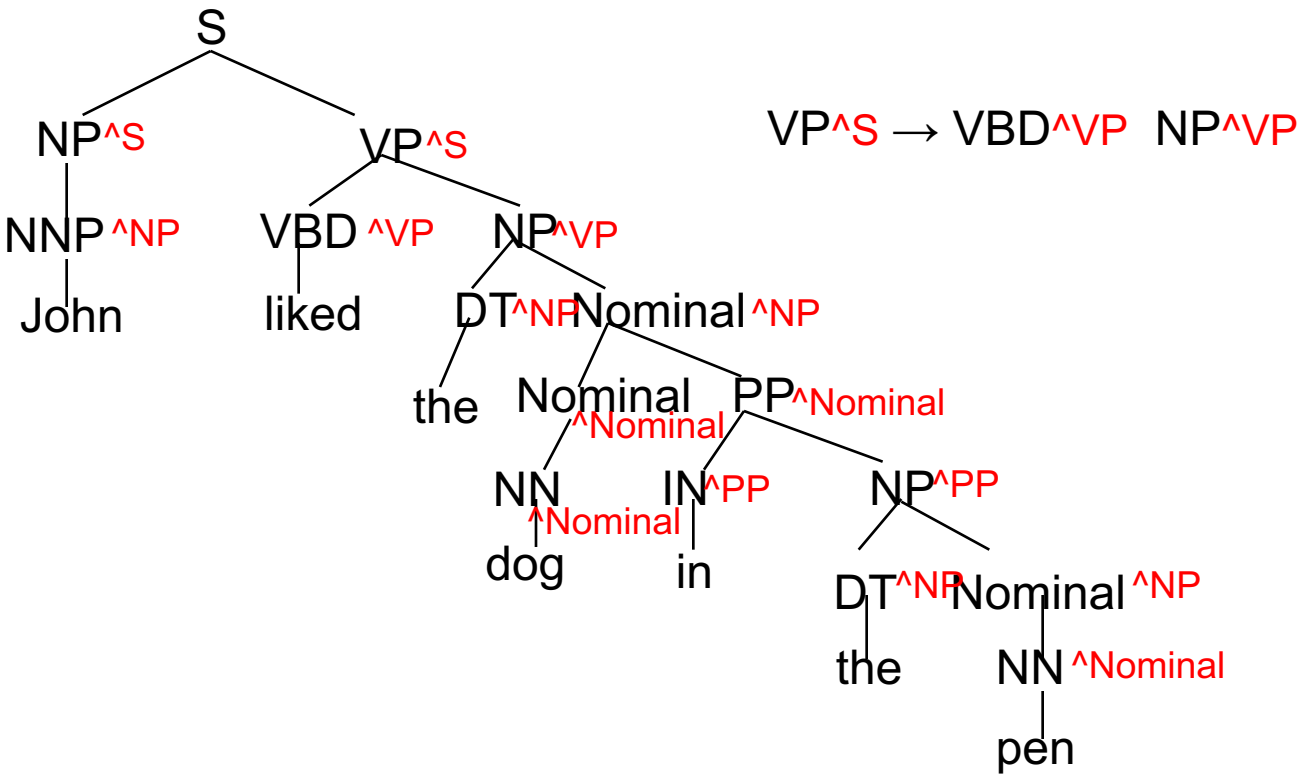
Missed Context Dependence

- Another problem with CFGs is that the production chosen to expand a non-terminal is independent of its context.
- However, this independence is frequently violated for normal grammars.
  - NPs that are subjects are more likely to be pronouns than NPs that are objects.

Splitting Non-Terminals

- To provide more contextual information, non-terminals can be split into multiple new non-terminals based on their parent in the parse tree using parent annotation.
  - A subject NP becomes NP^S since its parent node is an S.
  - An object NP becomes NP^VP since its parent node is a VP

Parent Annotation Example



Split and Merge

- Non-terminal splitting greatly increases the size of the grammar and the number of parameters that need to be learned from limited training data.
- Best approach is to only split non-terminals when it improves the accuracy of the grammar.
- May also help to merge some non-terminals to remove some un-helpful distinctions and learn more accurate parameters for the merged productions.
- Method: Heuristically search for a combination of splits and merges that produces a grammar that maximizes the likelihood of the training treebank.

Treebanks

- English Penn Treebank: Standard corpus for testing syntactic parsing consists of 1.2 M words of text from the Wall Street Journal (WSJ).
- Typical to train on about 40,000 parsed sentences and test on an additional standard disjoint test set of 2,416 sentences.
- Chinese Penn Treebank: 100K words from the Xinhua news service.
- Other corpora existing in many languages, see the Wikipedia article “Treebank”

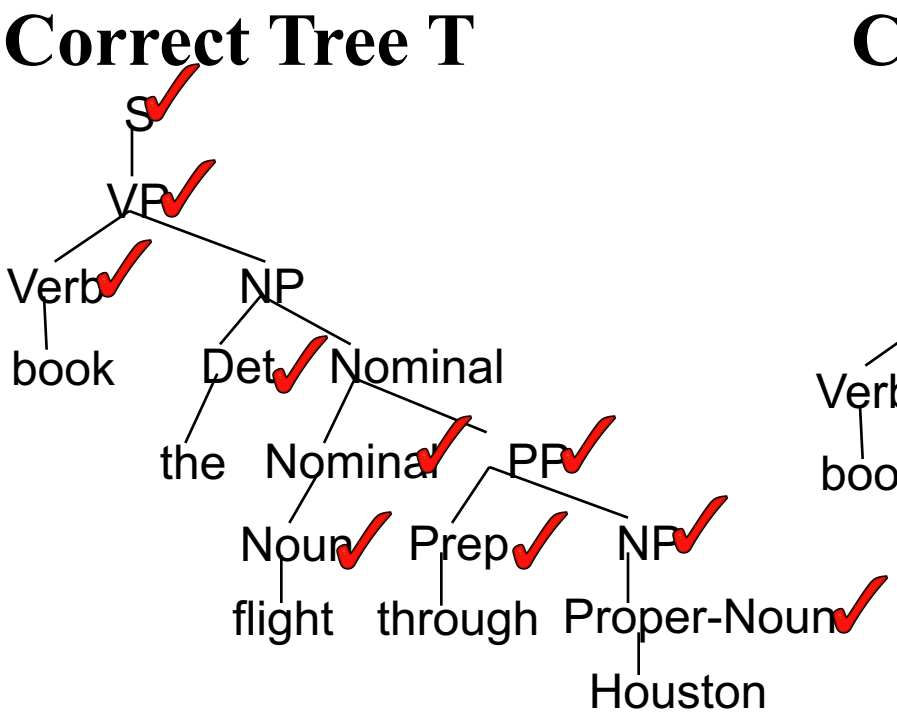
First WSJ Sentence

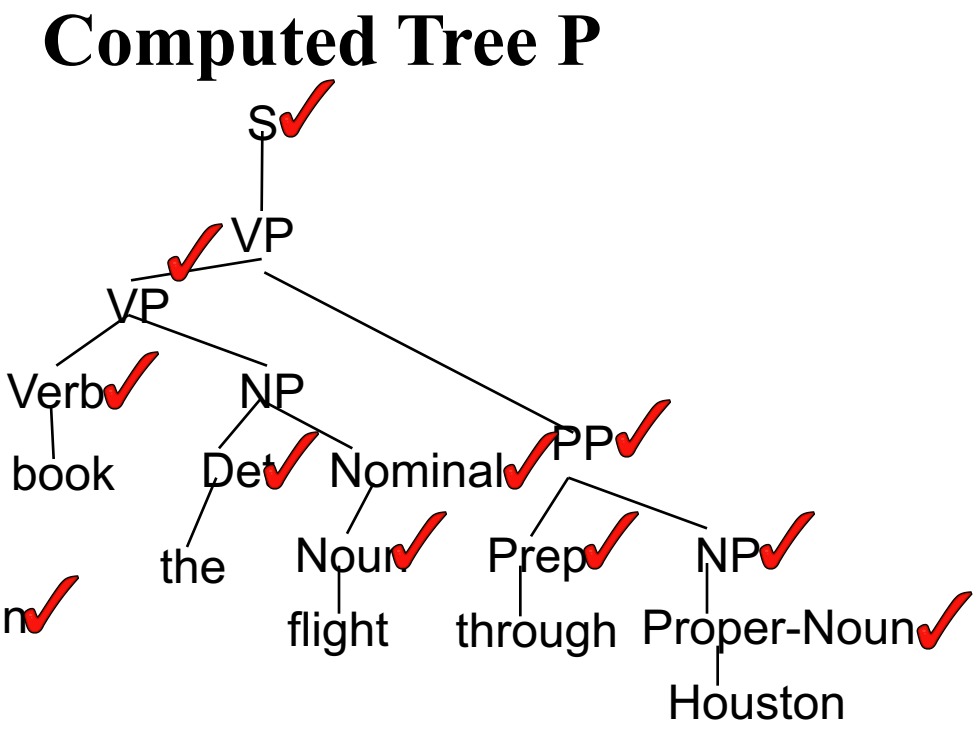
```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

Parsing Evaluation Metrics

- PARSEVAL metrics measure the fraction of the constituents that match between the computed and human parse trees. If *P* is the system's parse tree and *T* is the human parse tree (the "gold standard"):
  - ***Recall*** = (# correct constituents in *P*) / (# constituents in *T*)
  - ***Precision*** = (# correct constituents in *P*) / (# constituents in *P*)
- ***Labeled Precision*** and ***labeled recall*** require getting the non-terminal label on the constituent node correct to count as correct.
- F_1 is the harmonic mean of precision and recall.

Computing Evaluation Metrics





Constituents: 12

Constituents: 12

Correct Constituents: 10

Recall = 10/12= 83.3% Precision = 10/12=83.3% F_1 = 83.3%

Treebank Results

- Results of current state-of-the-art systems on the English Penn WSJ treebank are slightly greater than 90% labeled precision and recall.

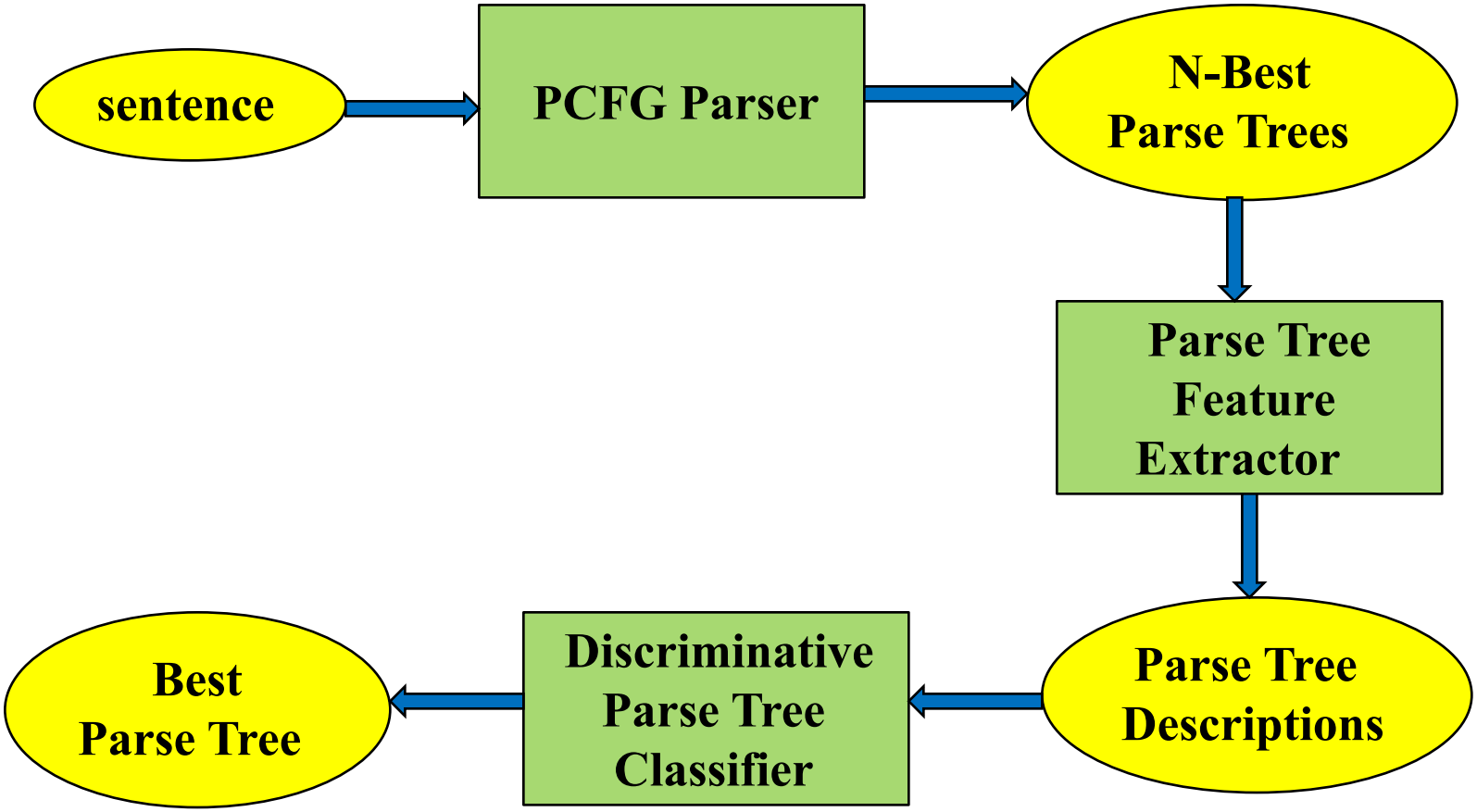
Discriminative Parse Reranking

- Motivation: Even when the top-ranked parse not correct, frequently the correct parse is one of those ranked highly by a statistical parser.
- Use a discriminative classifier that is trained to select the best parse from the N-best parses produced by the original parser.
- Reranker can exploit global features of the entire parse whereas a PCFG is restricted to making decisions based on local info.

2-Stage Reranking Approach

- Adapt the PCFG parser to produce an ***N-best list*** of the most probable parses in addition to the most-likely one.
- Extract from each of these parses, a set of global features that help determine if it is a good parse tree.
- Train a discriminative classifier (e.g. logistic regression) using the best parse in each N-best list as positive and others as negative.

Parse Reranking



Sample Parse Tree Features

- Probability of the parse from the PCFG.
- The number of parallel conjuncts.
  - "the bird in the tree and the squirrel on the ground"
  - "the bird and the squirrel in the tree"
- The degree to which the parse tree is right branching.
  - English parses tend to be right branching (cf. parse of "Book the flight through Houston")
- Frequency of various tree fragments, i.e. specific combinations of 2 or 3 rules.

Evaluation of Reranking

- Reranking is limited by ***oracle accuracy***, i.e. the accuracy that results when an omniscient oracle picks the best parse from the N-best list.
- Typical current oracle accuracy is around F_1=97%
- Reranking can generally improve test accuracy of current PCFG models a percentage point or two.

Other Discriminative Parsing

- There are also parsing models that move from generative PCFGs to a fully discriminative model, e.g. ***max margin parsing*** (Taskar *et al.*, 2004).
- There is also a recent model that efficiently reranks all of the parses in the complete (compactly-encoded) parse forest, avoiding the need to generate an N-best list (***forest reranking***, Huang, 2008).

Human Parsing

- Computational parsers can be used to predict human reading time as measured by tracking the time taken to read each word in a sentence.
- Psycholinguistic studies show that words that are more probable given the preceding lexical and syntactic context are read faster.
  - John put the dog in the pen with a lock.
  - John put the dog in the pen with a bone in the car.
  - John liked the dog in the pen with a bone.
- Modeling these effects requires an ***incremental*** statistical parser that incorporates one word at a time into a continuously growing parse tree.

The

The horse

The horse raced

The horse raced past

The horse raced past the

The horse raced past the barn

The horse raced past the barn fell

Garden Path Sentences

- People are confused by sentences that seem to have a particular syntactic structure but then suddenly violate this structure, so the listener is "lead down the garden path".

  The horse raced past the barn fell

  - vs. The horse raced past the barn broke his leg.

  – The complex houses married students.
  – The old man the sea.
  – While Peter dressed the baby spit up on the bed.
- Incremental computational parsers can try to predict and explain the problems encountered parsing such sentences.

Center Embedding

- Nested expressions are hard for humans to process beyond 1 or 2 levels of nesting.
  - The rat the cat chased died.
  - The rat the cat the dog bit chased died.
  - The rat the cat the dog the boy owned bit chased died.
- Requires remembering and popping incomplete constituents from a stack and strains human short-term memory.
- Equivalent “tail embedded” (tail recursive) versions are easier to understand since no stack is required.
  - The boy owned a dog that bit a cat that chased a rat that died.

Statistical Parsing Conclusions

- Statistical models such as PCFGs allow for probabilistic resolution of ambiguities.
- PCFGs can be easily learned from treebanks.
- Lexicalization and non-terminal splitting are required to effectively resolve many ambiguities.
- Current statistical parsers are quite accurate but not yet at the level of human-expert agreement.

Other Applications of PCFGs

- Authorship attribution (Raghavan *et al.*, 2010)
- Modeling of language acquisition (Waterfall *et al.*, 2010)