Natural Language Generation, Non-Metric Methods, Probabilistic Context Free Grammar, Parsing Algorithms, NLP Tools

Sameer Maskey

Week 4, Sept 26, 2012

*animation slides on parsing obtained from Prof Raymond Mooney

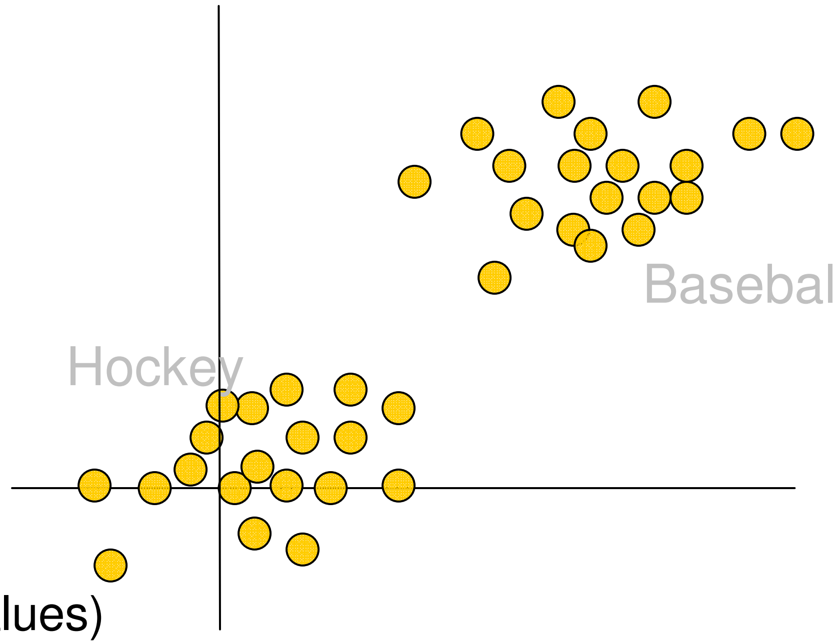
Topics for Today

- Non-metric Methods
- Probabilistic Context Free Grammar
- Parsing Algorithms
  - CKY Parsing
- Writing your Grammar and Parser
- Weighted Finite State Transducers
- Using WFST in NLP and Speech processing tasks

Announcement

- Proposal Due tonight (11:59pm) – Graded
  - 5% of the project grade
  - Email me the proposal with the title
    - "Project Proposal : Statistical NLP for the Web"
  - Use the following format if appropriate
    1. Abstract/Summary
    2. Introduction and Related Work
    3. Data
    4. NLP/ML Algorithms
    5. System Description (end-to-end)
    5. Conclusion
- Homework 1 due October 4th (11:59pm) Thursday
  - Start early

K-Means in Words



- Parameters to estimate for K classes
- Let us assume we can model this data
- with mixture of two Gaussians
- Start with 2 Gaussians (initialize mu values)
- Compute distance of each point to the mu of 2 Gaussians and assign it to the closest Gaussian (class label (Ck))
- Use the assigned points to recompute mu for 2 Gaussians

- Minimize J with respect to r_{nk}
  - Keep μ_k fixed

Step 1

- Optimize for each n separately by choosing r_{nk} for k that gives minimum $||x_n - r_{nk}||^2$

$$r_{nk} = 1 \text{ if } k = argmin_j ||x_n - \mu_j||^2$$
$$= 0 \text{ otherwise}$$

- Assign each data point to the cluster that is the closest
- Hard decision to cluster assignment

- Minimize J with respect to μ_k
  - Keep r_{nk} fixed

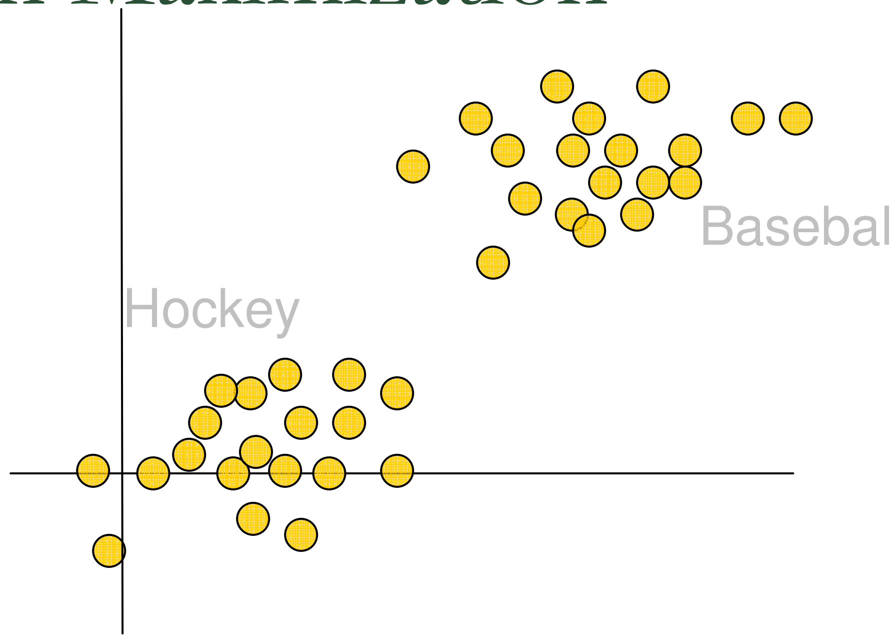
Step 2

- J is quadratic in μ_k. Minimize by setting derivative w.rt. μ_k to zero

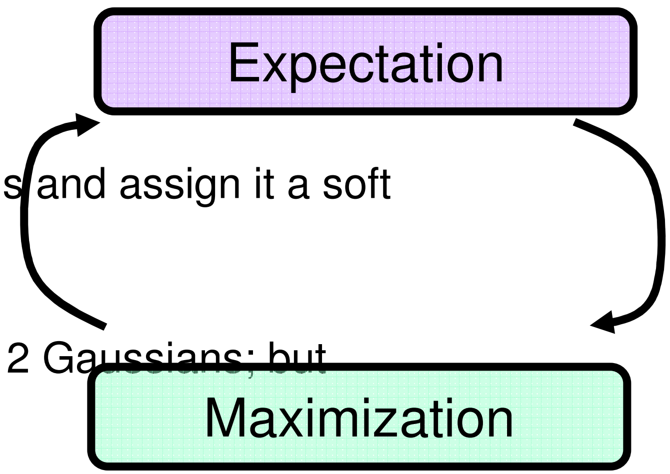
$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

- Take all the points assigned to cluster K and re-estimate the mean for cluster K

Explaining Expectation Maximization



- EM is like fuzzy K-means
- Parameters to estimate for K classes
- Let us assume we can model this data with mixture of two Gaussians (K=2)



- Start with 2 Gaussians (initialize mu and sigma values)
- Compute distance of each point to the mu of 2 Gaussians and assign it a soft class label (Ck)
- Use the assigned points to recompute mu and sigma for 2 Gaussians; but weight the updates with soft labels

Estimating Parameters

$$\gamma(z_{nk}) = E(z_{nk}|x_n) = p(z_k = 1|x_n)$$

- E-Step

Estimating Parameters

- M-step

$$\mu'_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

$$\sum'_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu'_k)(x_n - \mu'_k)^T$$

$$\pi'_k = \frac{N_k}{N} \quad \gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \sum_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \sum_j)}$$

$$\text{where } N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

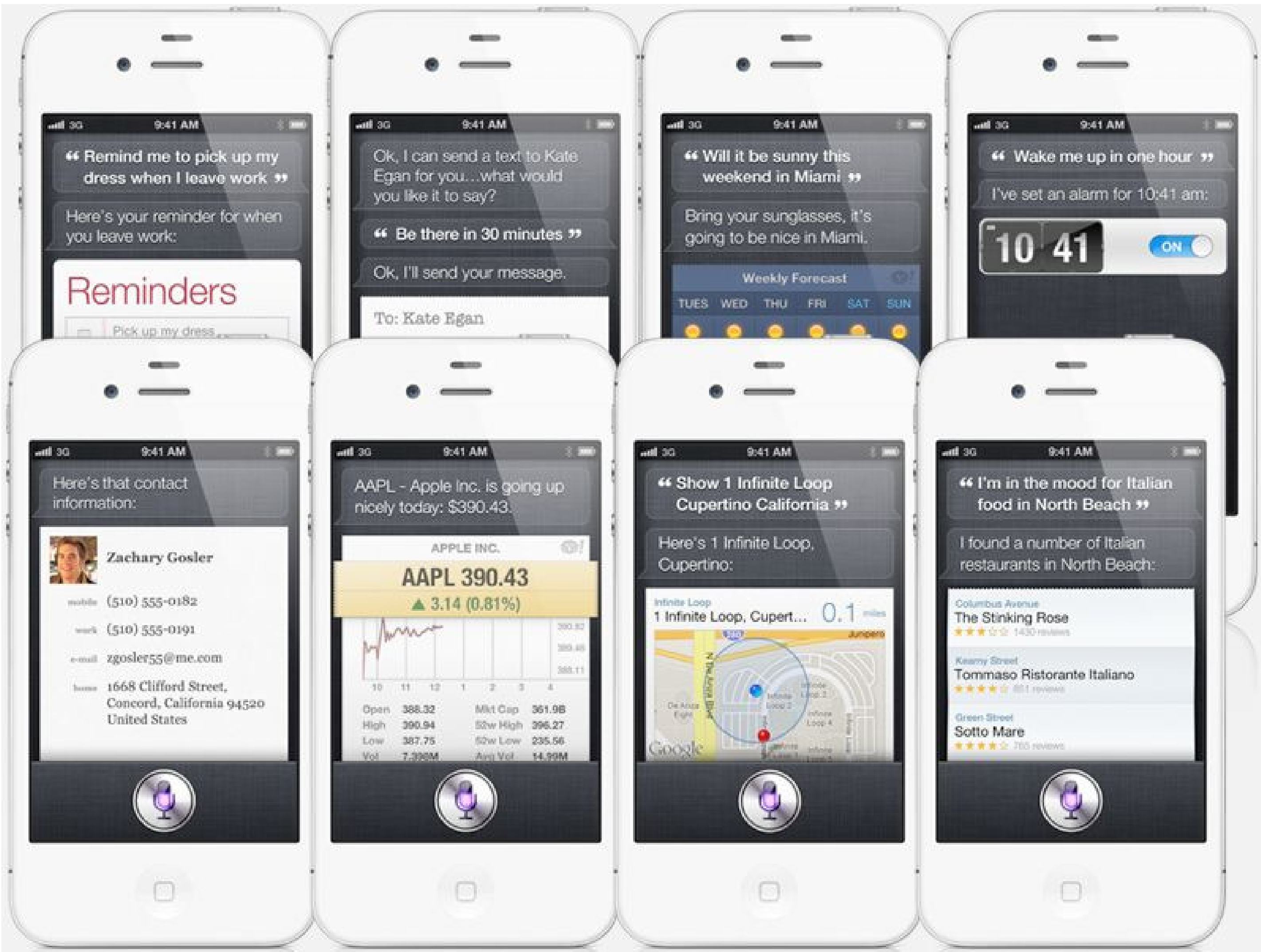
- Iterate until convergence of log likelihood

$$log\ p(X|\pi, \mu, \sum) = \sum_{n=1}^{N} log\ (\sum_{k=1}^{k} \mathcal{N}(x|\mu_k, \sum_k))$$

Hierarchical Clustering Algorithm

- Step 1
  - Assign each data point to its own cluster
- Step 2
  - Compute similarity between clusters
- Step 3
  - Merge two most similar cluster to form one less cluster

Human-Machine Dialog



Human-Machine Dialog

Machine may need to generate text to communicate with Humans

Natural Language Generation

- For machines to communicate with humans they need to know how to generate **valid** meaningful text
- Validity
  - Morphologically
  - Syntactically
  - Semantically
- How about discourse?

Natural Language Generation (NLG)

- Text generation used in various NLP tasks
  - Summarization
  - Machine translation
  - Question Answering
  - Dialog System
- Based on data and tasks, generation methods vary widely
  - Text to Text Generation
  - Database to Text Generation
  - Speech to Text Generation
  - Concept to Text Generation
- Text Generators? :
  - http://www.elsewhere.org/pomo/
  - http://pdos.csail.mit.edu/scigen/

NLG

- McDonald (1987)
  - Natural language generation is the process of deliberately constructing a natural language text in order to meet specified communicative goals.
- Dale (1997):
  - Natural language generation is the subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable texts in... human languages from some underlying non-linguistic representation of information.

Dialog System

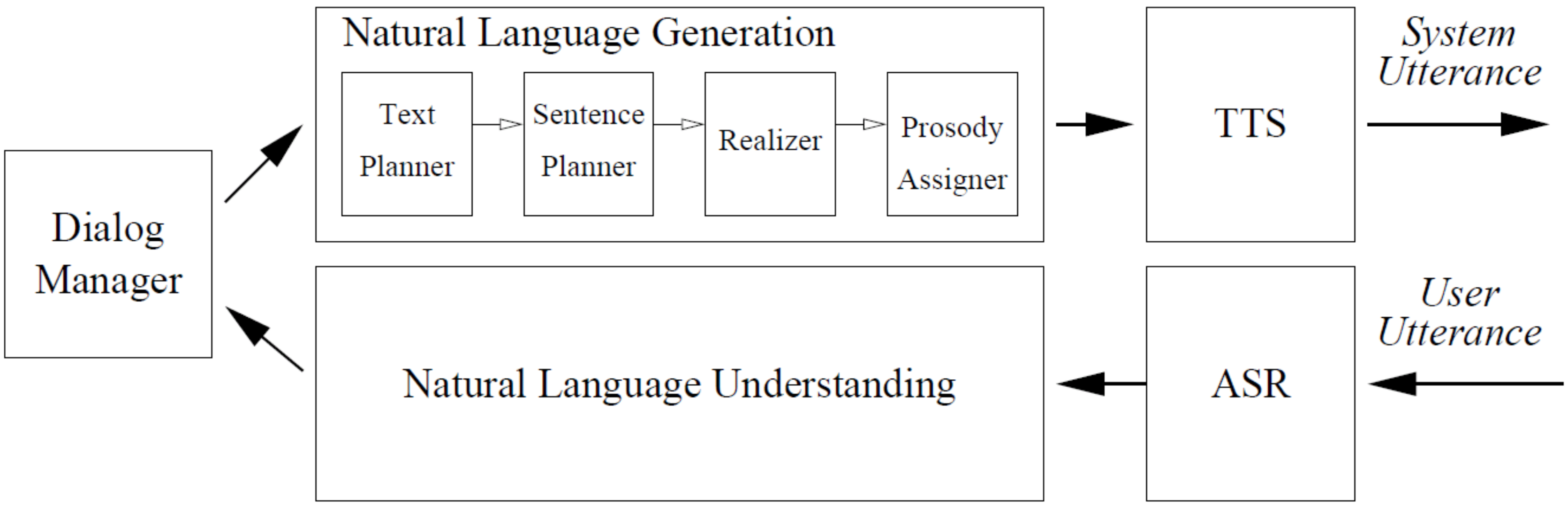
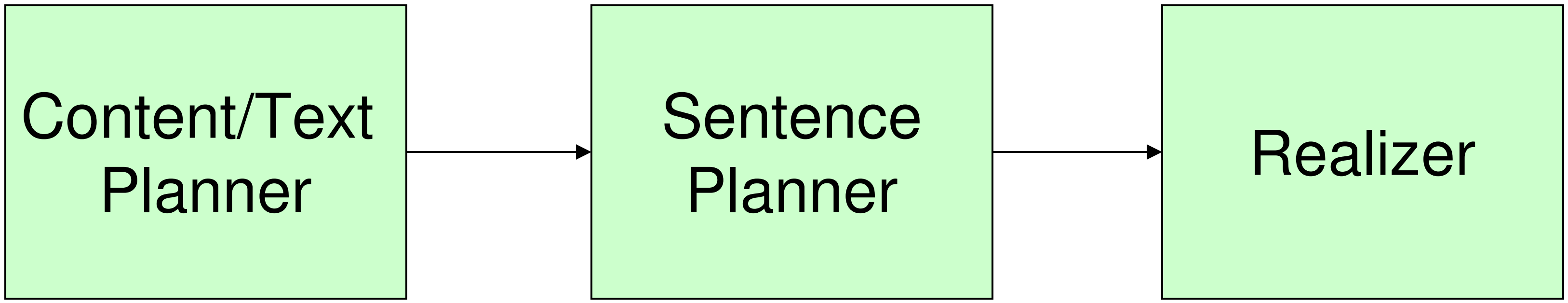
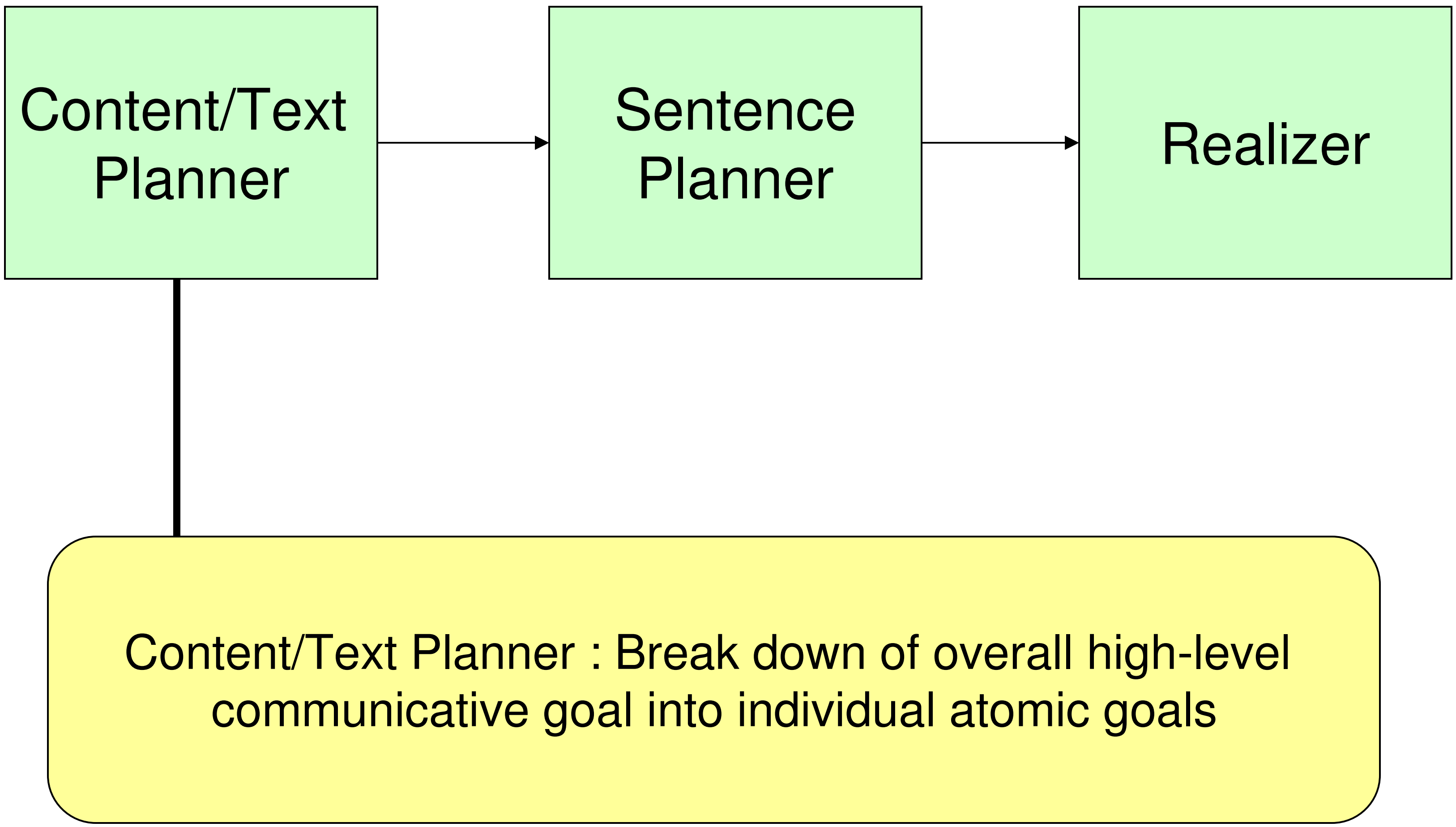


Figure Source :Natural Language Generation in Dialog System [Rambow, et. al]

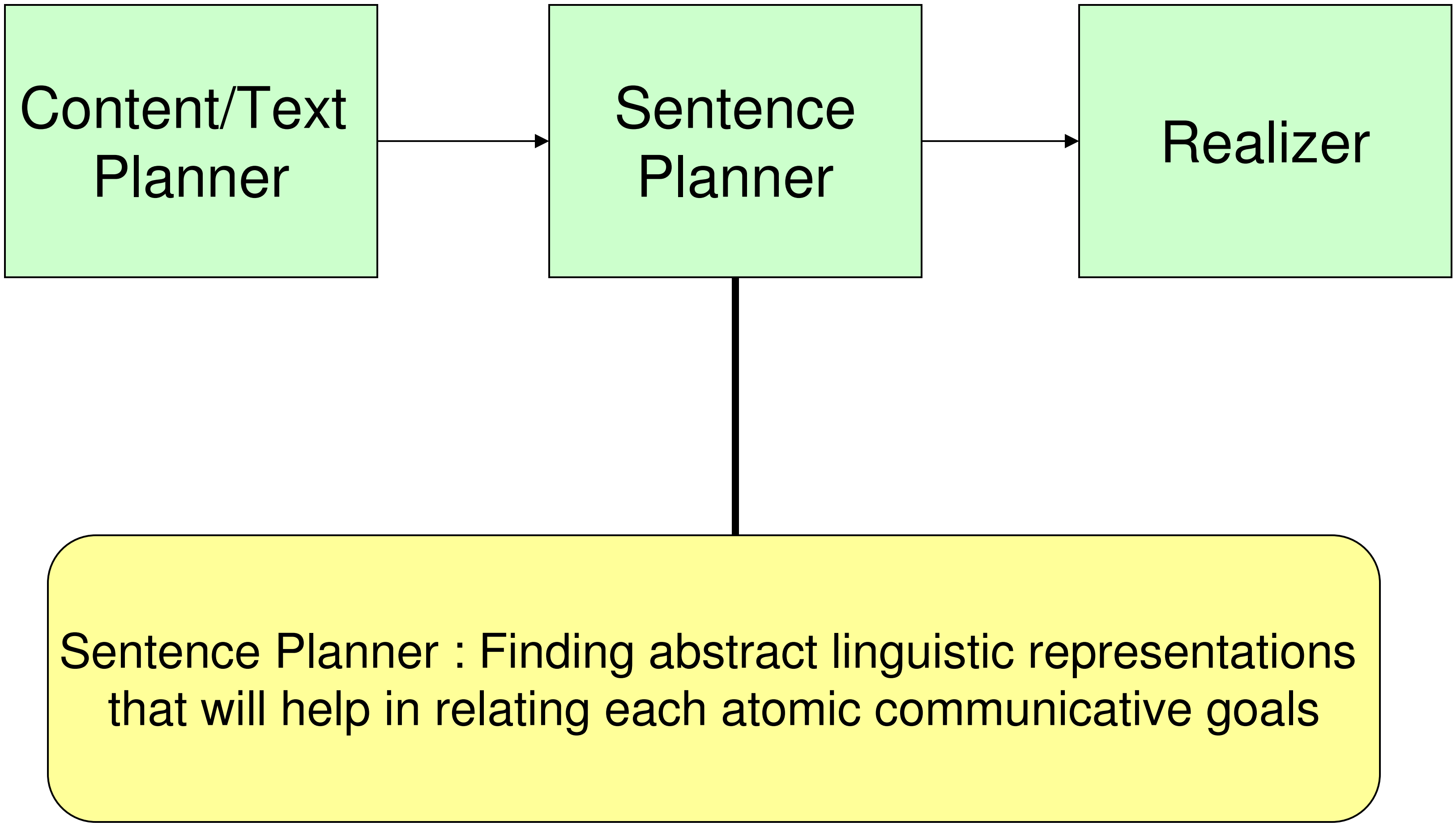
NLG Components



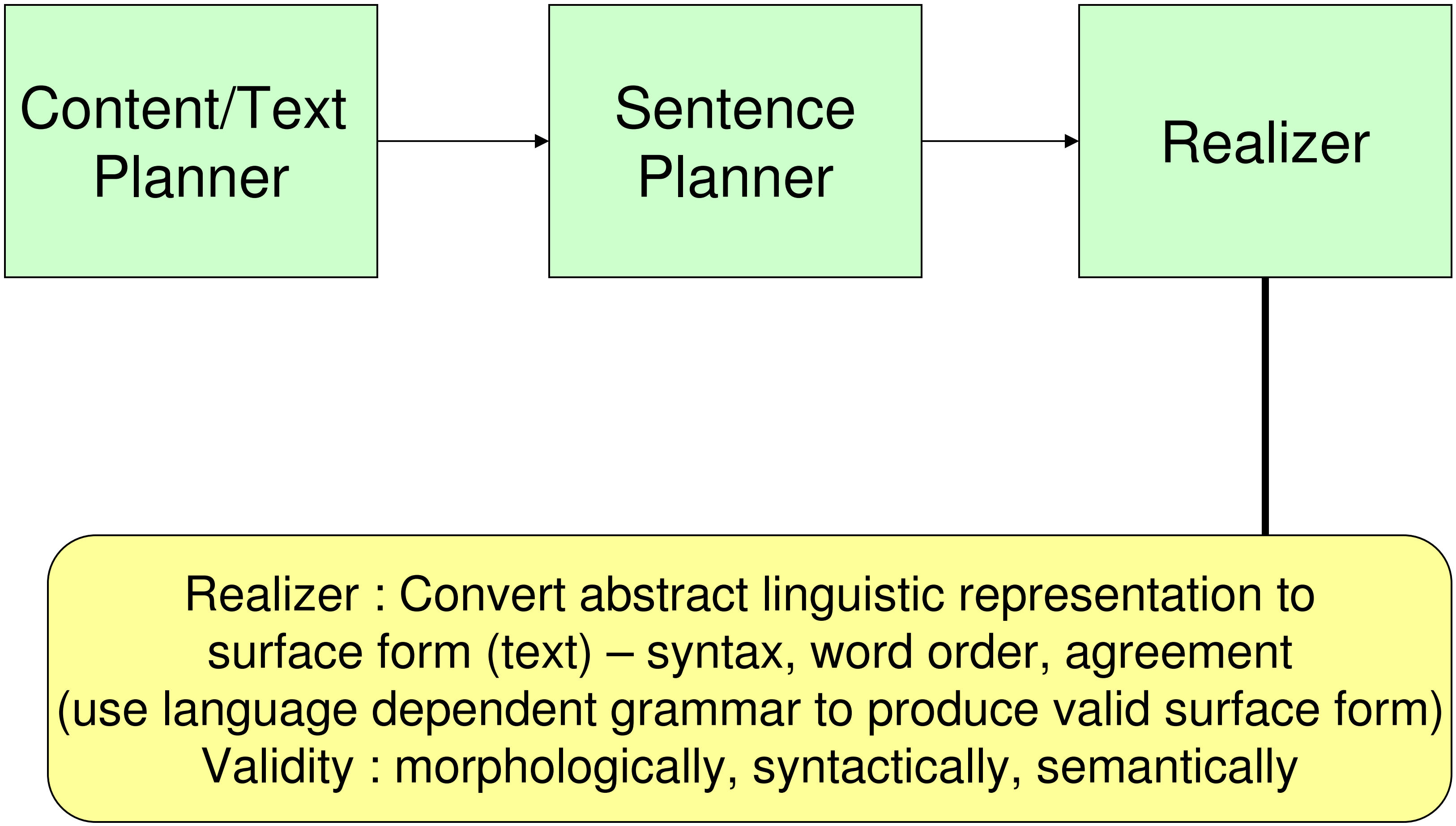
NLG Components



NLG Components

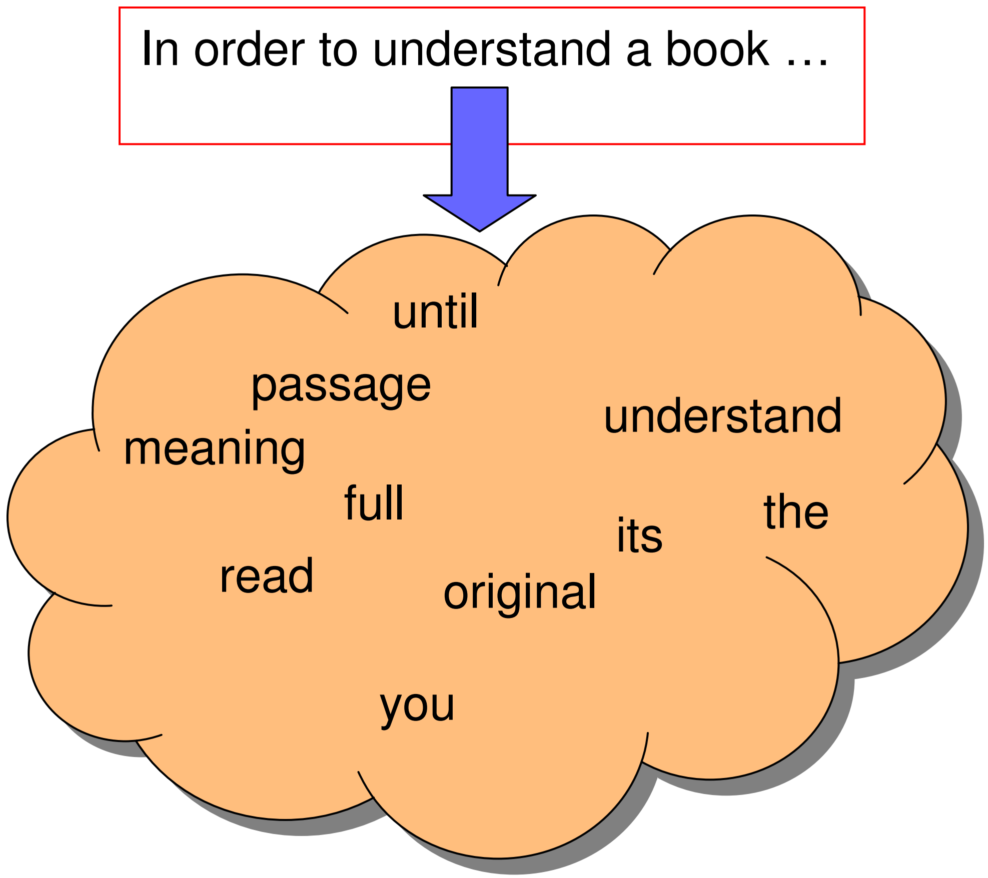


NLG Components

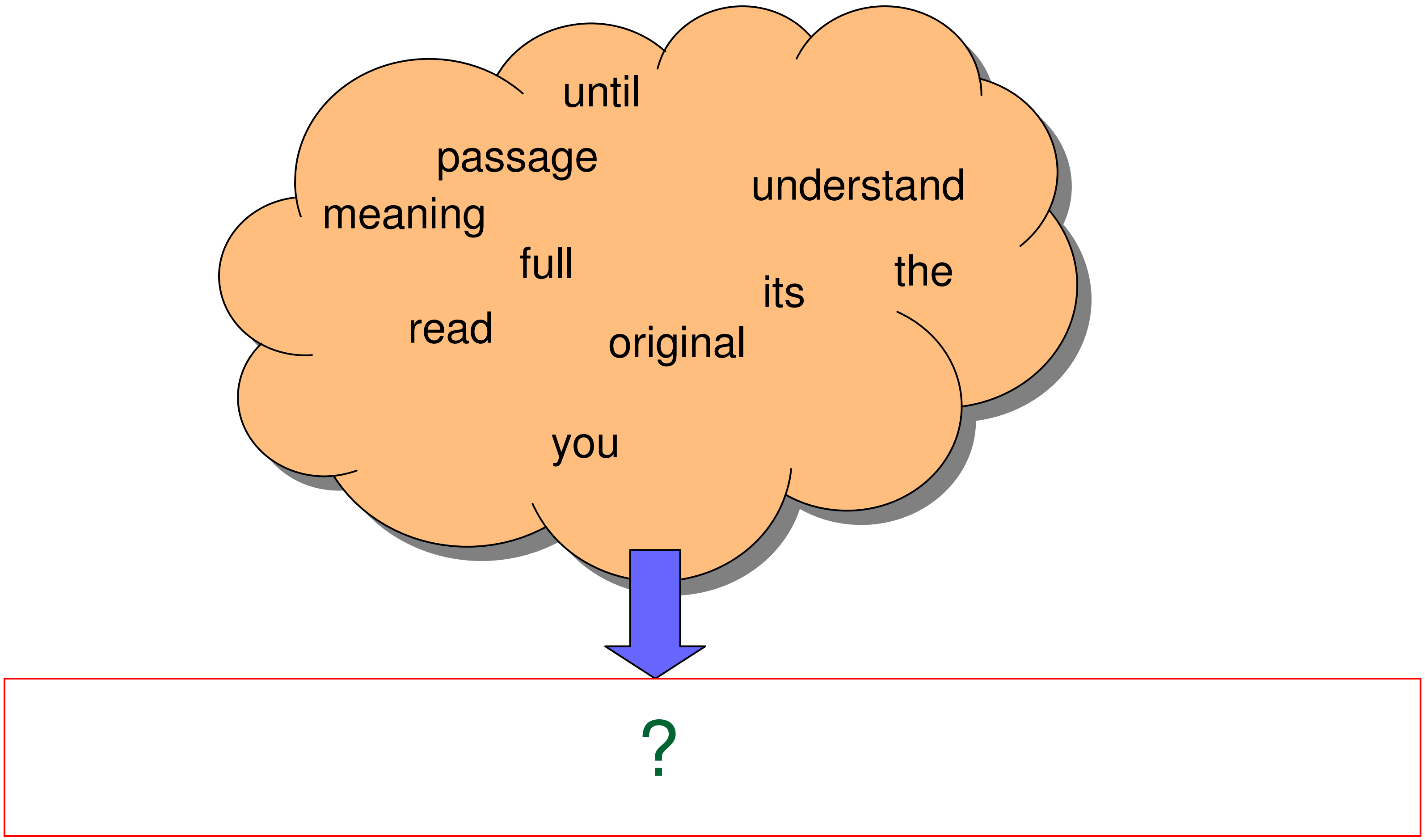


Text to Words : Bag of Words Model

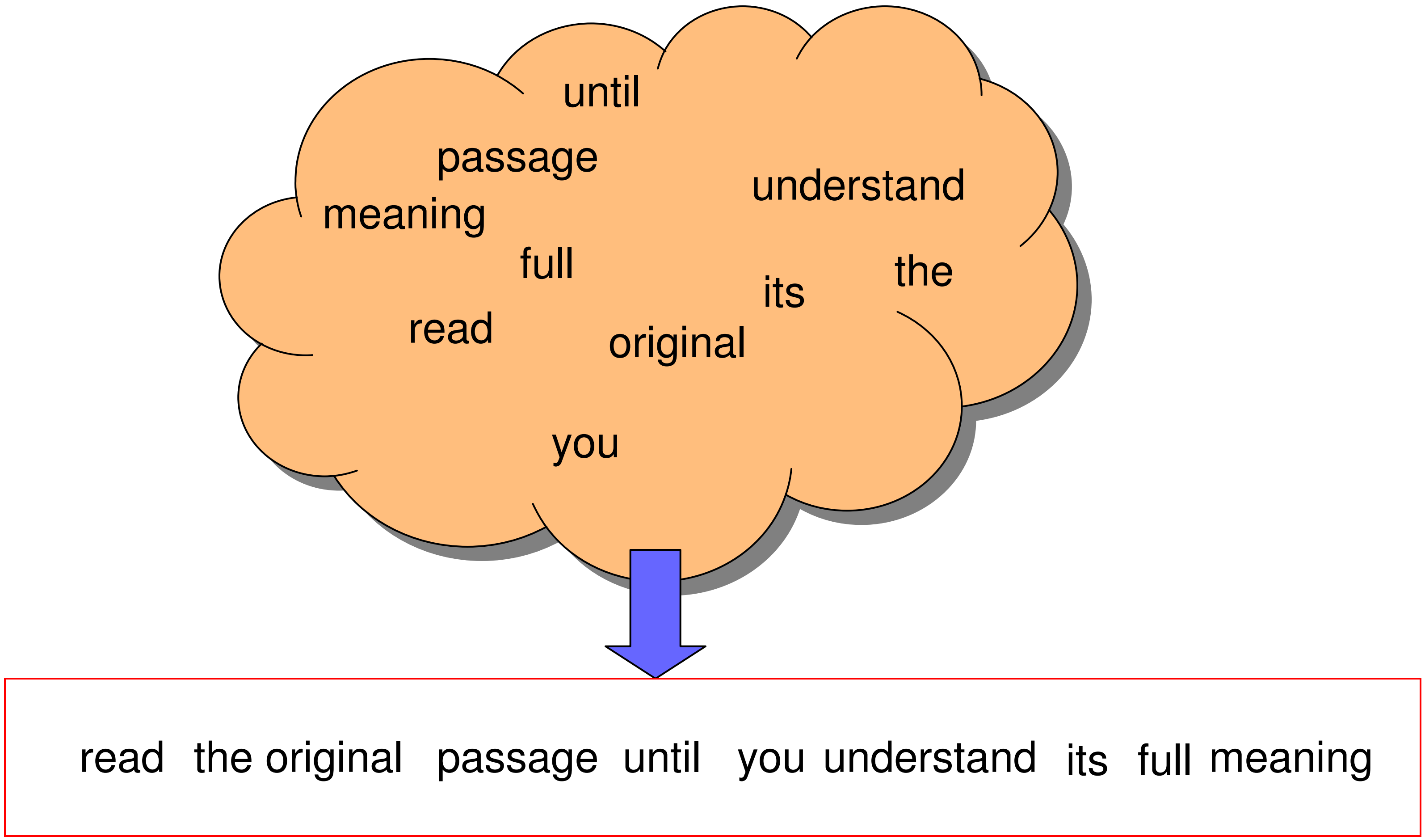
- We produced multinomial vectors from a given piece of text



Words to Text



Human Sentence Generation Performance



Human Performance

- Humans backtrack to disambiguate
- Many points of disambiguation
- Frequency matters
- Generated sentence by human
  - Syntactically sound
  - Semantically sound
- Do we start think of **semantics** first or **syntax** first?

read the original passage until you understand its full meaning

Syntax and Semantics

read the original passage until you understand its full meaning

Syntax

-study of combinatorics of basic units of language

-how to combine words together?

Semantics

-study of meaning

-what do grouped words mean together?

```
(S (VP read
        (NP the original passage)
        (SBAR until
              (S (NP you)
                  (VP understand
                   (NP its full meaning))))))
```

Meaning of grouped words
"read the original passage"
vs
"read the passage"

Putting Words Together

- Combinatorial problem created when we try to put words together is huge
- Try producing all possible word combination of our previous sentence of length 10
  - Total combinations : 10^10 = 1 billion sentences
  - Sent1 : “read the the the … passage” is unlikely
  - Sent2 : “read the passage …” is more likely
- How can we come up with scores that are higher for Sent1 than Sent2
  - Don’t allow to group words like “the the the”
    - Make such construction invalid
  - Invalidity as defined by a set of rules that govern the language
  - Such rules define the grammar
  - For mathematical modeling easier to use “Context Free Grammar”

Non-Metric Methods

- Can we use previously learned ML algorithms for NLG
  - Yes
- Why is it difficult?
  - Combination problem
  - Notion of metric or distance is limited
    - What is the mean of distribution of all possible sentence combination of length 10?
    - Distance between
      - “What is Apple?” - “What is Vodafone?”
      - “What is Apple?” – “What is Orange?”
      - “What is Apple?” – “What is a fruit?”
      - “What is Apple?” – “What is a rock?”
      - “What is Apple?” – “What is the?” (?)
- No clear notion of similarity
- From vector of real numbers to list of attributes

Non-Metric Methods

- Decision Trees
- Rule Based Methods
- Grammar based Methods
- Finite State Transducers

Grammar Based Methods

- **Regular Grammar**
  - Can be represented by Finite State Automata
- **Context Free Grammar**
  - Allows only 1 symbol on LHS
  - Can apply the rule without caring about what is the context (left and right symbols)
  - Well suited to describe recursive syntax
- **Context Sensitive Grammar**
  - Allows more than 1 symbol on LHS
  - aZb → aKb can only be applied to non-terminal Z only in the context of a and b
- **Unrestricted Grammar**
  - E.g. natural language

Context Free Grammars (CFG)

- N a set of ***non-terminal symbols*** (or ***variables***)
- Σ a set of ***terminal symbols*** (disjoint from N)
- R a set of ***productions*** or ***rules*** of the form $A \rightarrow \beta$, where A is a non-terminal and β is a string of symbols from $(\Sigma \cup N)^*$
- S, a designated non-terminal called the ***start symbol***

*animation starting this one is provided by Prof. Raymond Mooney

Simple CFG for ATIS English

Grammar

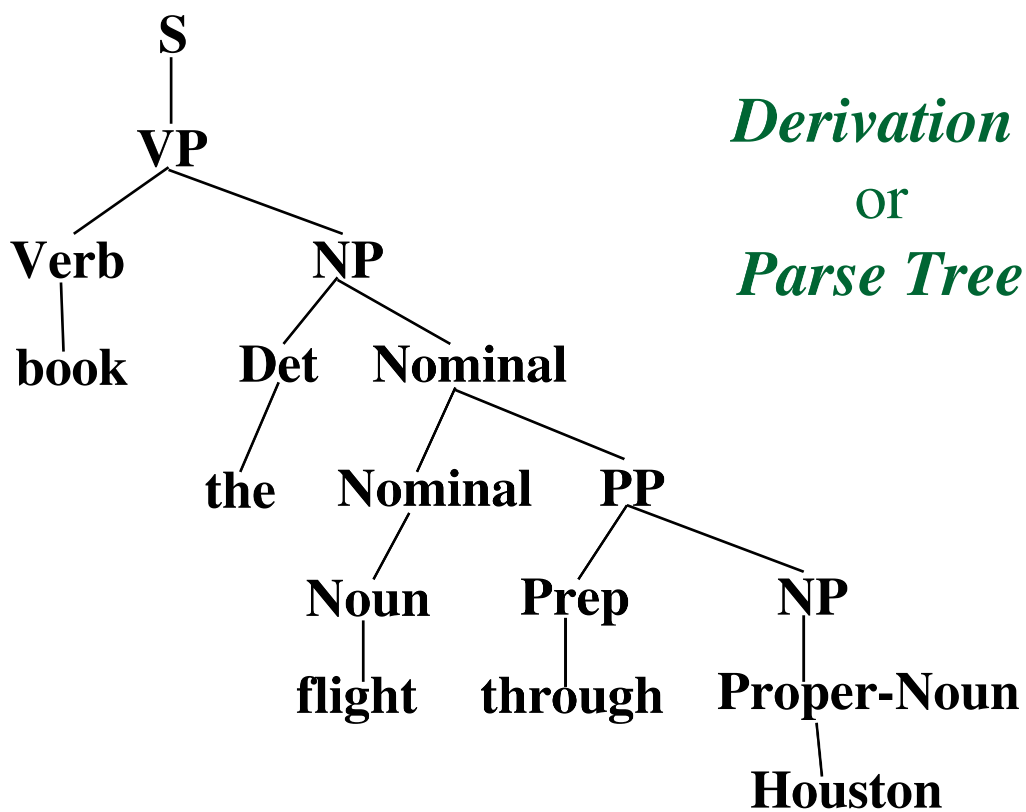
S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

Lexicon

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

Sentence Generation

- Sentences are generated by recursively rewriting the start symbol using the productions until only terminals symbols remain.



Derivation
or
Parse Tree

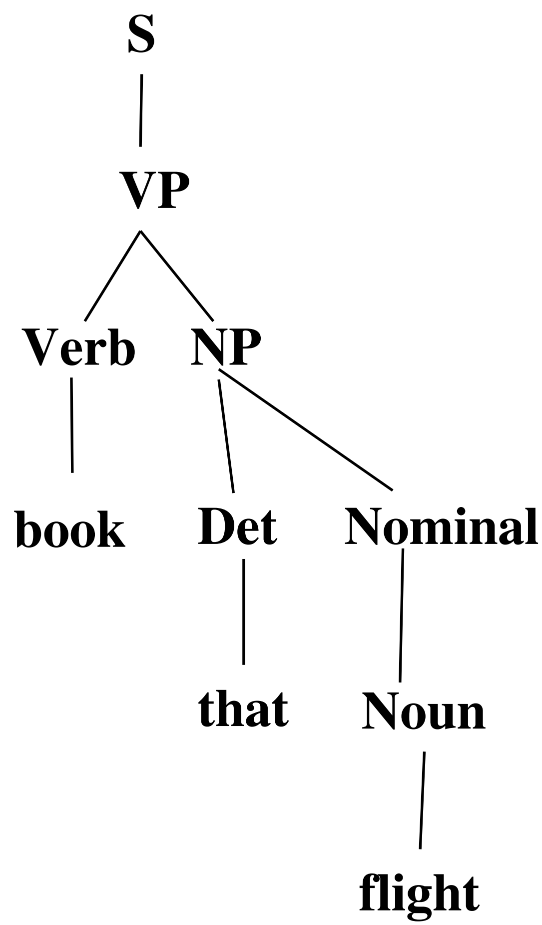
Parsing

- Given a string of terminals and a CFG, determine if the string can be generated by the CFG.
  - Also return a parse tree for the string
  - Also return all possible parse trees for the string

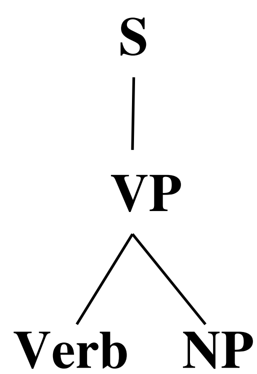
- Must search space of derivations for one that derives the given string.
  - **Top-Down Parsing**: Start searching space of derivations for the start symbol.
  - **Bottom-up Parsing**: Start search space of reverse deivations from the terminal symbols in the string.

Parsing Example

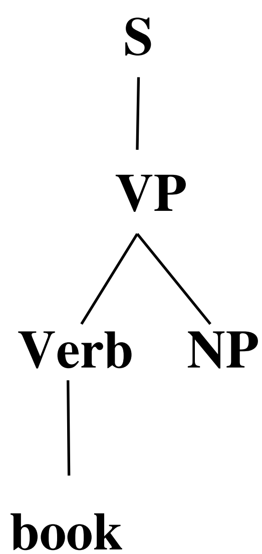
book that flight



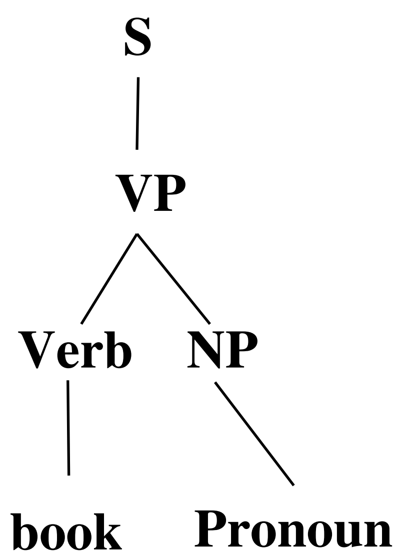
Top Down Parsing



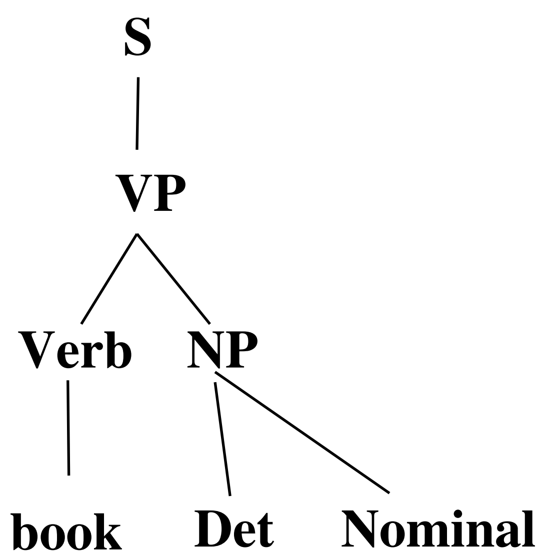
Top Down Parsing



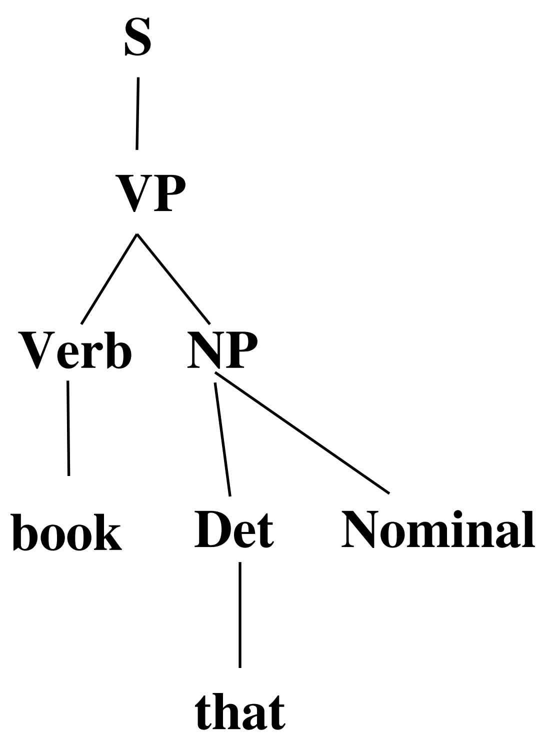
Top Down Parsing



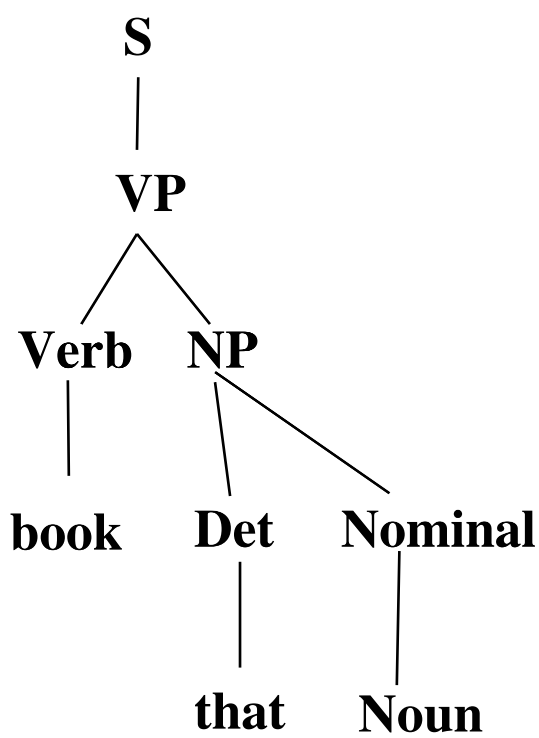
Top Down Parsing



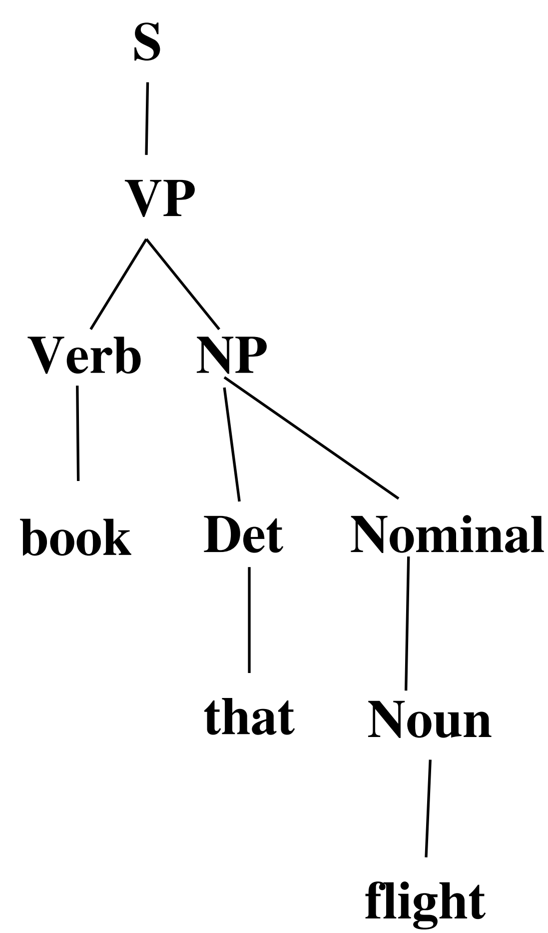
Top Down Parsing



Top Down Parsing



Top Down Parsing



Simple CFG for ATIS English

Grammar

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

Lexicon

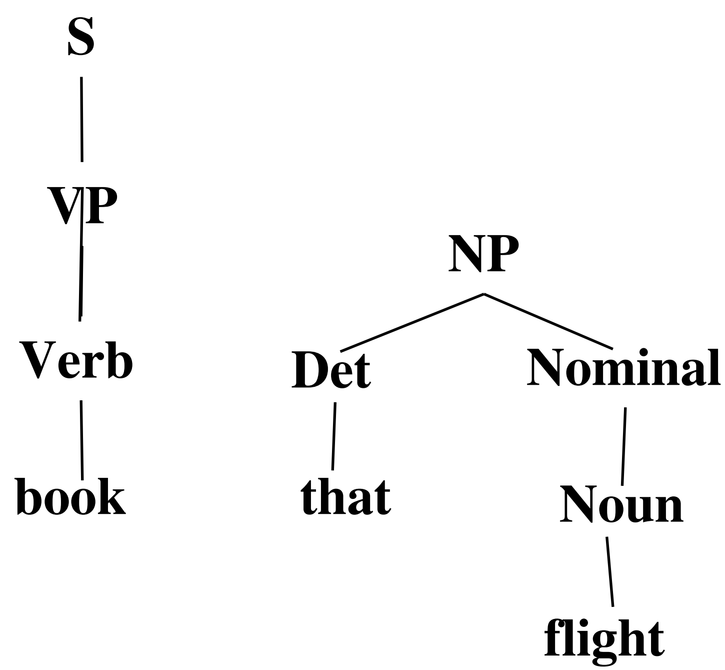
Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

Bottom Up Parsing

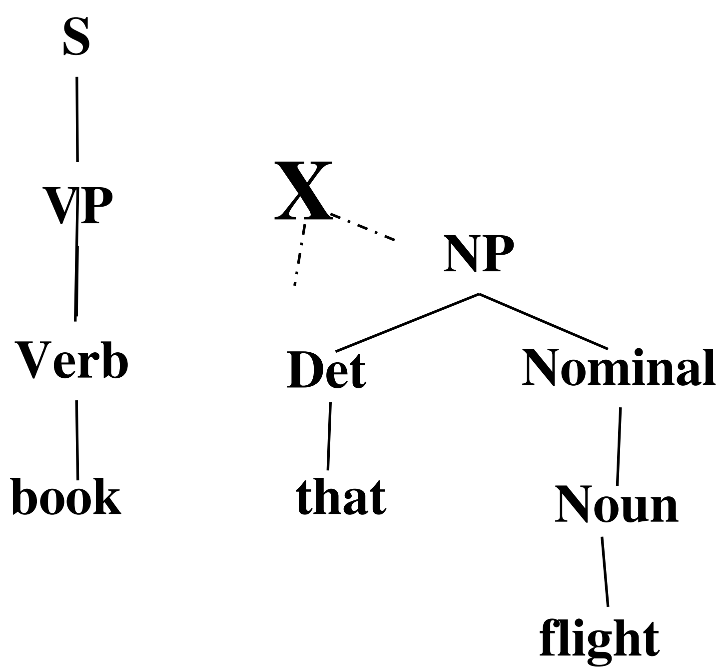
book **that** **flight**

43

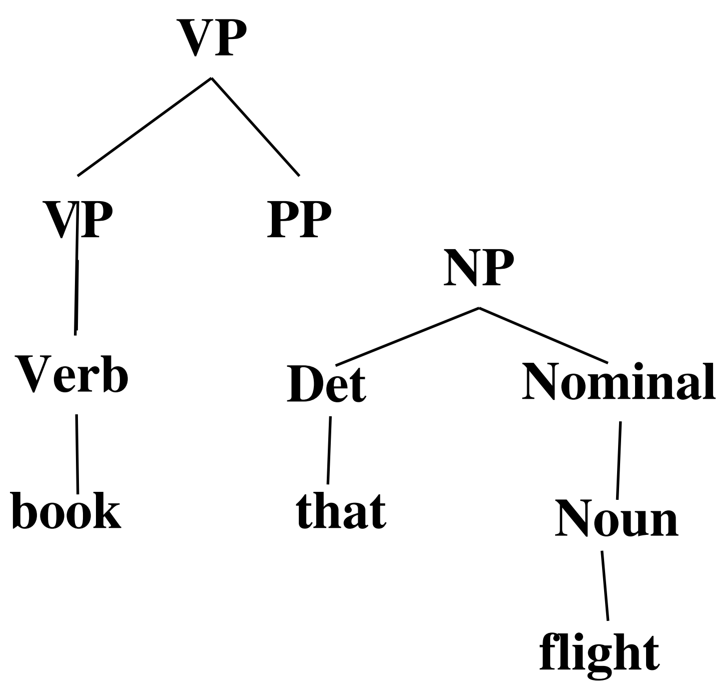
Bottom Up Parsing



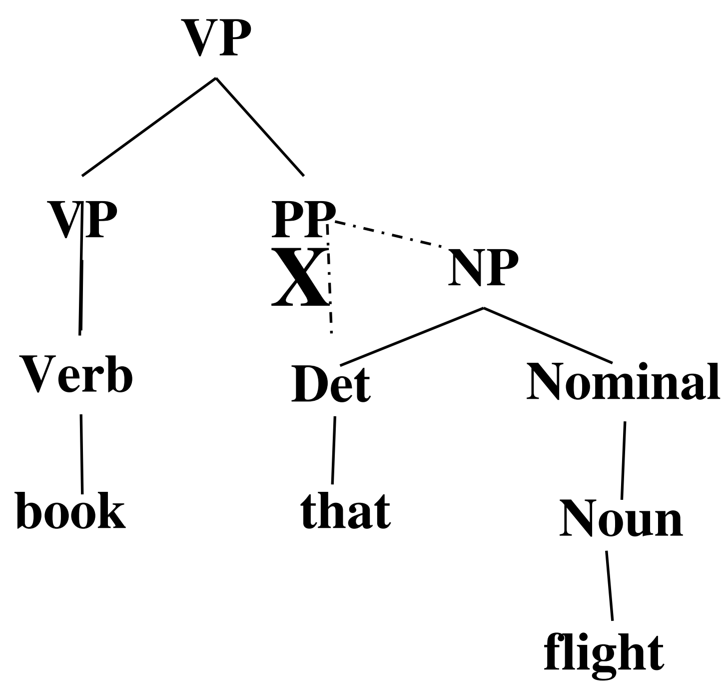
Bottom Up Parsing



Bottom Up Parsing

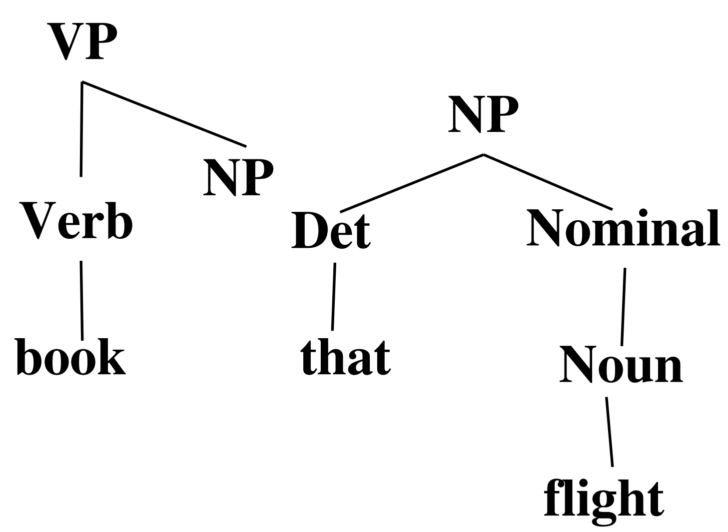


Bottom Up Parsing

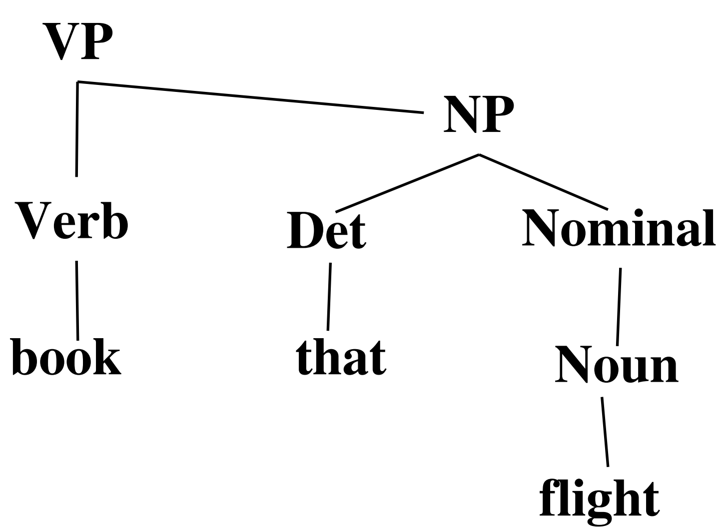


47

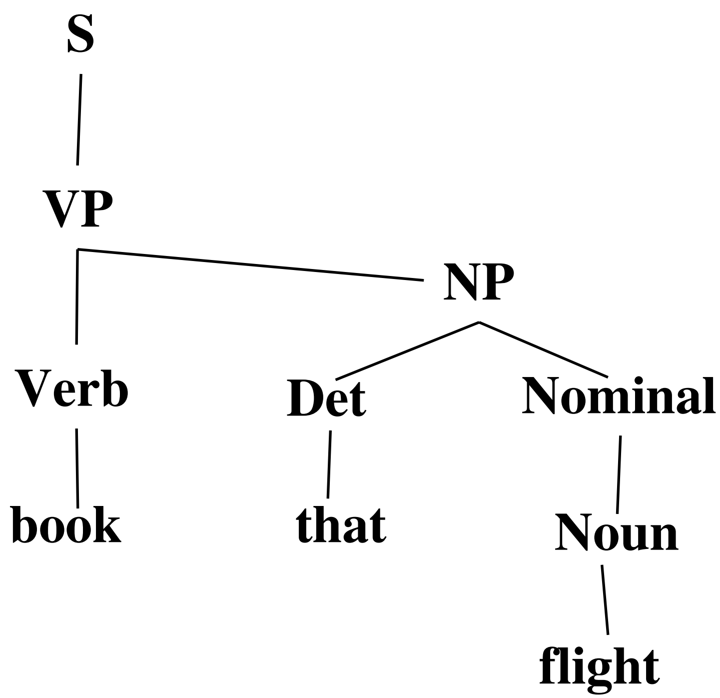
Bottom Up Parsing



Bottom Up Parsing



Bottom Up Parsing



Top Down vs. Bottom Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.
- Relative amounts of wasted search depend on how much the grammar branches in each direction.

51

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.
- Caching critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.
- Dynamic programming algorithms based on both top-down and bottom-up search can achieve $O(n^3)$ recognition time where *n* is the length of the input string.

52

Dynamic Programming Parsing Methods

- **CKY** (Cocke-Kasami-Younger) algorithm based on bottom-up parsing and requires first normalizing the grammar.
- **Earley parser** is based on top-down parsing and does not require normalizing grammar but is more complex.
- More generally, **chart parsers** retain completed phrases in a chart and can combine top-down and bottom-up search.

CKY

- First grammar must be converted to **Chomsky normal form (CNF)** in which productions must have either exactly 2 non-terminal symbols on the RHS or 1 terminal symbol (lexicon rules).
- Parse bottom-up storing phrases formed from all substrings in a triangular table (chart).

ATIS English Grammar Conversion

<table>
<tr><th>Original Grammar</th><th>Chomsky Normal Form</th></tr>
<tr><td>S → NP VP</td><td>S → NP VP</td></tr>
<tr><td>S → Aux NP VP</td><td>S → X1 VP</td></tr>
<tr><td></td><td>X1 → Aux NP</td></tr>
<tr><td>S → VP</td><td>S → book | include | prefer</td></tr>
<tr><td></td><td>S → Verb NP</td></tr>
<tr><td></td><td>S → VP PP</td></tr>
<tr><td>NP → Pronoun</td><td>NP → I | he | she | me</td></tr>
<tr><td>NP → Proper-Noun</td><td>NP → Houston | NWA</td></tr>
<tr><td>NP → Det Nominal</td><td>NP → Det Nominal</td></tr>
<tr><td>Nominal → Noun</td><td>Nominal → book | flight | meal | money</td></tr>
<tr><td>Nominal → Nominal Noun</td><td>Nominal → Nominal Noun</td></tr>
<tr><td>Nominal → Nominal PP</td><td>Nominal → Nominal PP</td></tr>
<tr><td>VP → Verb</td><td>VP → book | include | prefer</td></tr>
<tr><td>VP → Verb NP</td><td>VP → Verb NP</td></tr>
<tr><td>VP → VP PP</td><td>VP → VP PP</td></tr>
<tr><td>PP → Prep NP</td><td>PP → Prep NP</td></tr>
</table>

CKY Parser

	Book j= 1	the 2	flight 3	through 4	Houston 5
i= 0					
1					
2					
3					
4					

Cell[*i*,*j*] contains all constituents (non-terminals) covering words *i* +1 through *j*

56

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None			
	Det	NP		
		Nominal, Noun		

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	VP		
	Det	NP		
		Nominal, Noun		

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP		
	Det	NP		
		Nominal, Noun		

59

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP		
	Det	NP		
		Nominal, Noun		

60

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	
	Det	NP	None	
		Nominal, Noun	None	
			Prep	

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	
	Det	NP	None	
		Nominal, Noun	None	
			Prep	PP
				NP ProperNoun

62

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	
	Det	NP	None	
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

63

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

64

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	VP
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

65

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	S VP
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

66

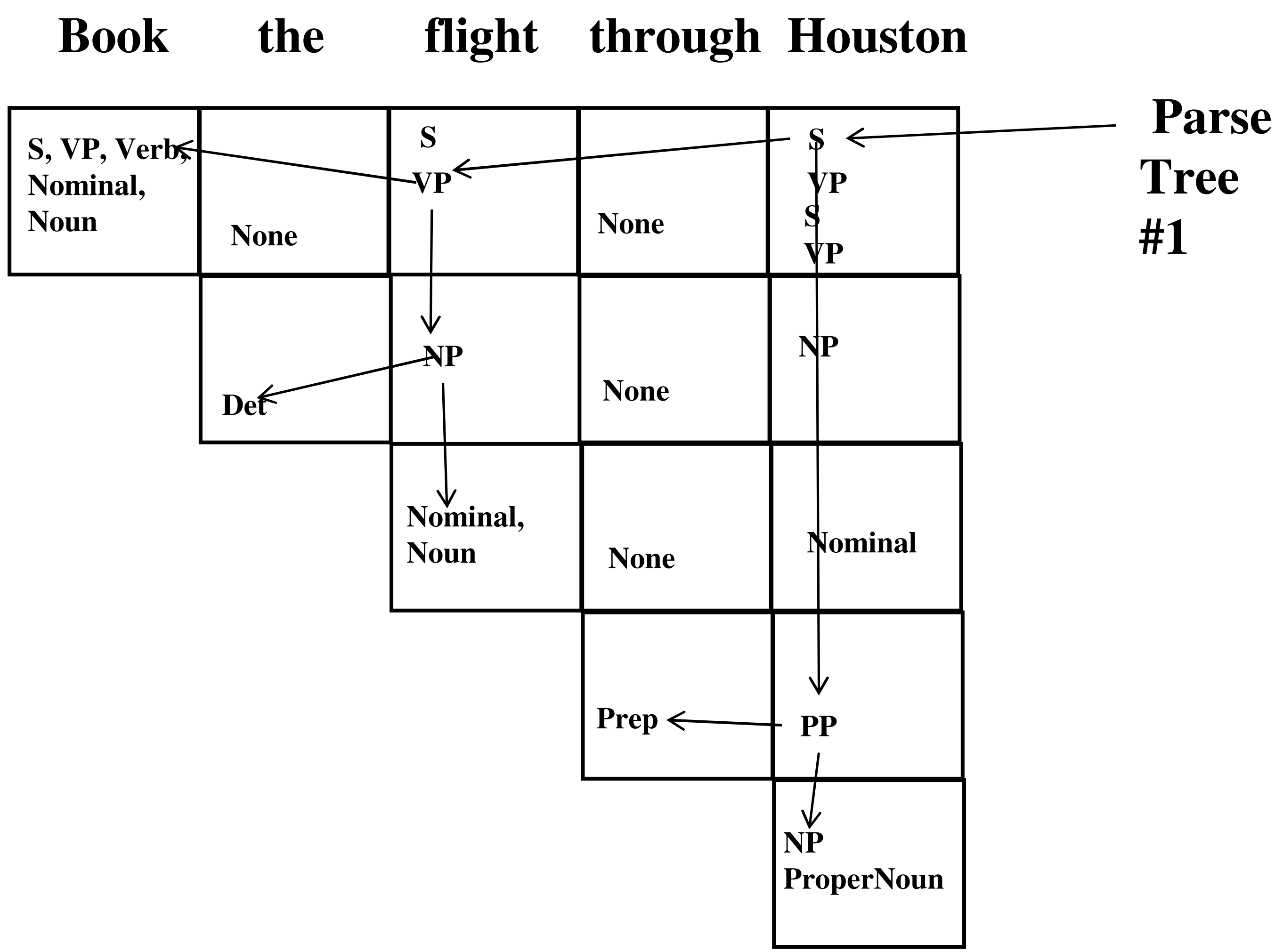
CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	VP S VP
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	S VP S VP
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

CKY Parser


Book
the
flight
through
Houston
S, VP, Verb, Nominal, Noun
None
S
VP
None
S
VP
S
VP
Parse Tree #1
Det
NP
None
NP
Nominal, Noun
None
Nominal
Prep
PP
NP
ProperNoun

CKY Parser

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun	None	S VP	None	S VP S VP
	Det	NP	None	NP
		Nominal, Noun	None	Nominal
			Prep	PP
				NP ProperNoun

Parse Tree #2

70

Complexity of CKY (recognition)

- There are $(n(n+1)/2) = O(n^2)$ cells
- Filling each cell requires looking at every possible split point between the two non-terminals needed to introduce a new phrase.
- There are $O(n)$ possible split points.
- Total time complexity is $O(n^3)$

71

Probabilistic Context Free Grammar (PCFG)

- A PCFG is a probabilistic version of a CFG where each production has a probability.
- Probabilities of all productions rewriting a given non-terminal must add to 1, defining a distribution for each non-terminal.
- String generation is now probabilistic where production probabilities are used to non-deterministically select a production for rewriting a given non-terminal.

72

Simple PCFG for ATIS English

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

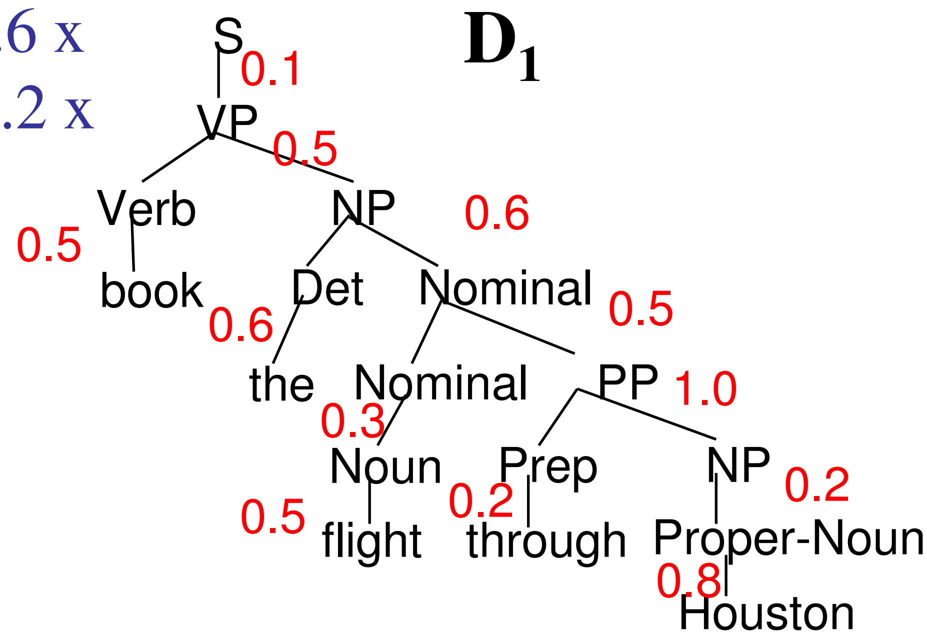
Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2

Sentence Probability

- Assume productions for each node are chosen independently.
- Probability of derivation is the product of the probabilities of its productions.

$$
\begin{aligned}
P(D_1) = {} & 0.1 \times 0.5 \times 0.5 \times 0.6 \times 0.6 \times \\
& 0.5 \times 0.3 \times 1.0 \times 0.2 \times 0.2 \times \\
& 0.5 \times 0.8 \\
= {} & 0.0000216
\end{aligned}
$$

D_1

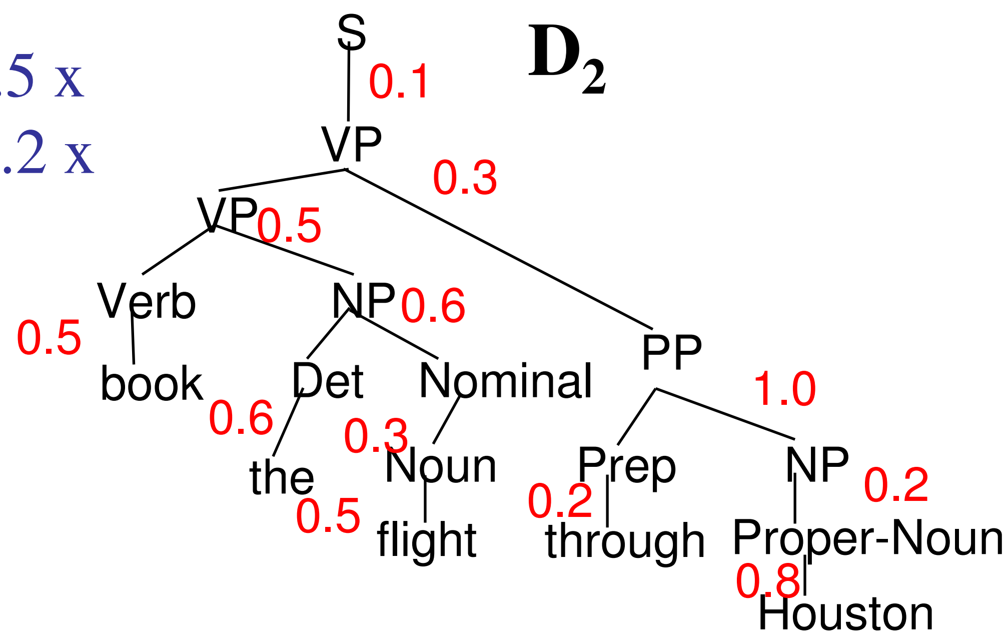


74

Syntactic Disambiguation

- Resolve ambiguity by picking most probable parse tree.

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



75

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.

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

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None			
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

80

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	VP:.5*.5*.054 =.0135		
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135		
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

82

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	

83

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

85

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.05*.5* .000864 =.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.03*.0135*.032 =.00001296 S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6*.0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

88

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

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

89

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.

O_1

The dog big barked.

The big dog barked

O_2

90

Inside Algorithm

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

Probabilistic CKY Parser for Inside Computation

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:..00001296 S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

92

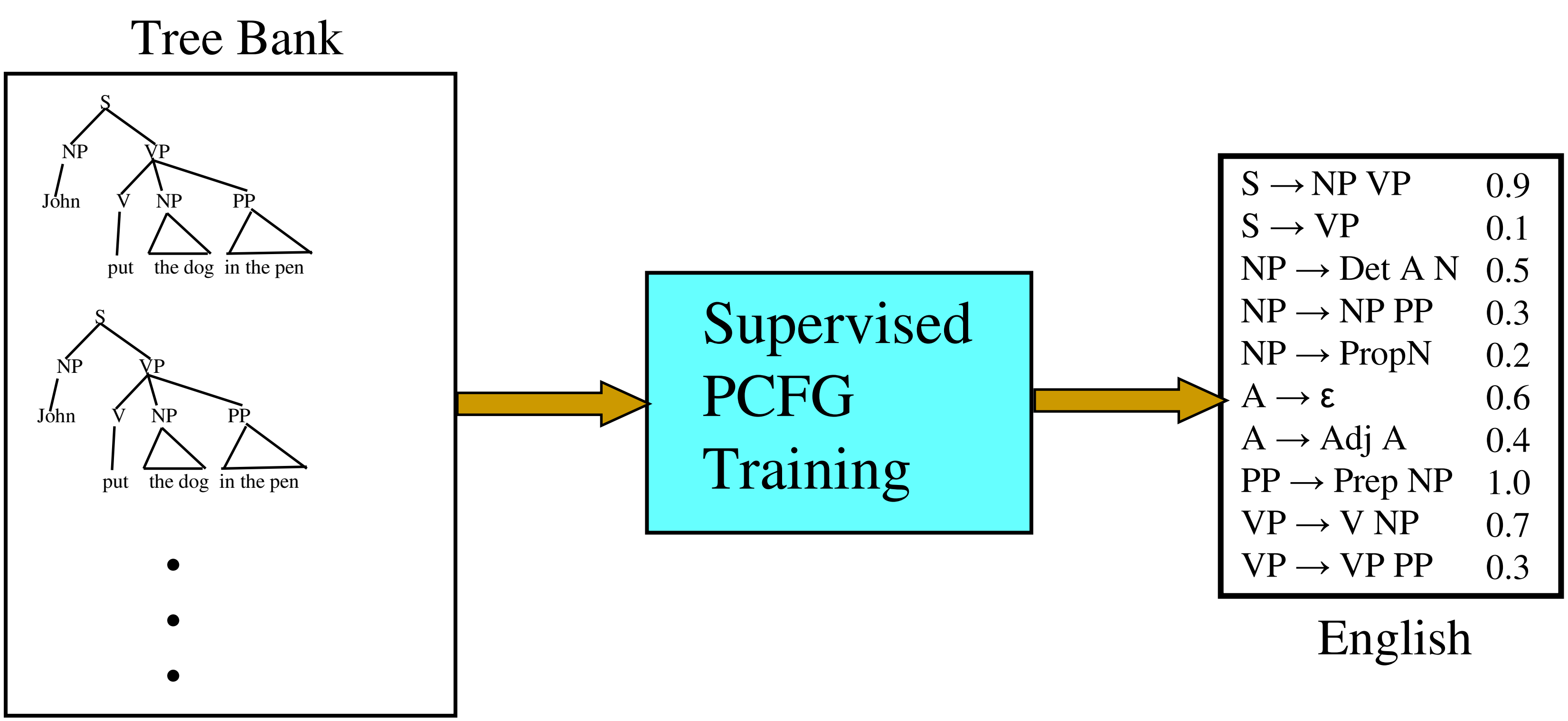
Probabilistic CKY Parser for Inside Computation

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S: .00001296 +.0000216 =.00003456
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Sum probabilities of multiple derivations of each constituent in each cell.

PCFG: Supervised Training

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



94

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.

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

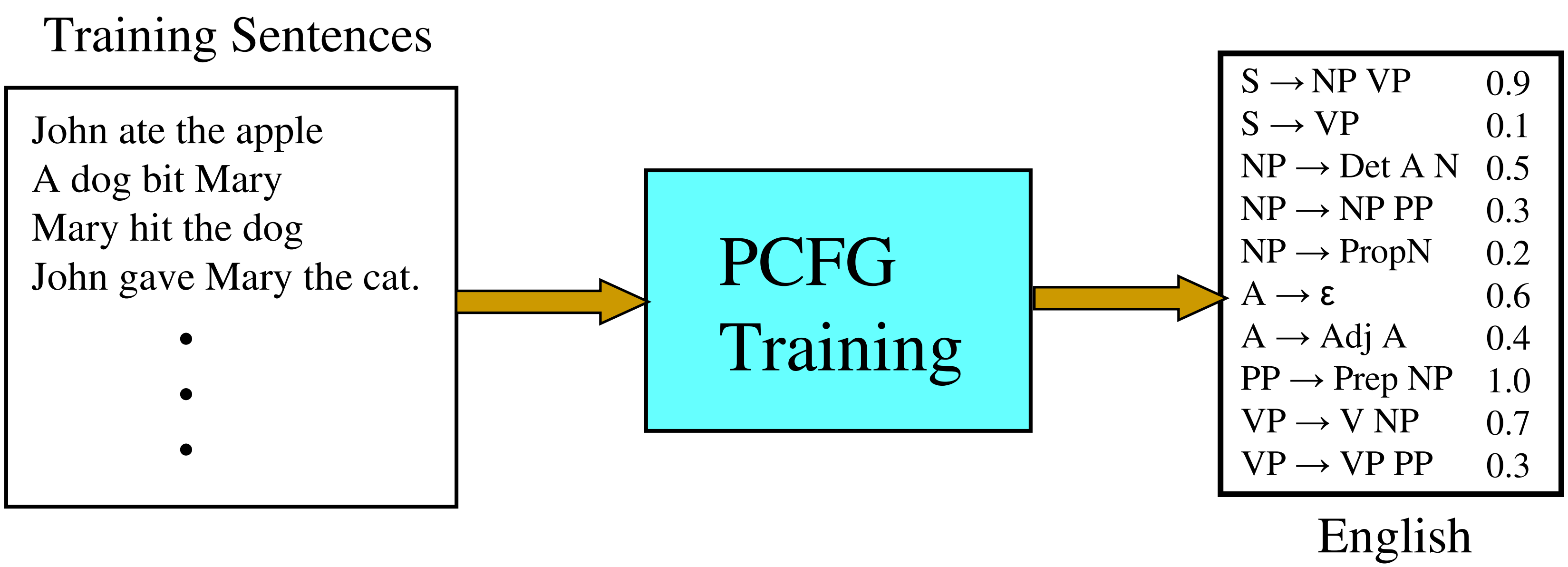
95

PCFG: Maximum Likelihood Training

- Given a set of sentences, induce a grammar that maximizes the probability that this data was generated from this grammar.
- Assume the number of non-terminals in the grammar is specified.
- Only need to have an unannotated set of sequences generated from the model. Does not need correct parse trees for these sentences. In this sense, it is unsupervised.

96

PCFG: Maximum Likelihood Training



Write Your Own CFG

- Palindromes
- We want to construct a grammar that creates palindromes
  - aabbaa, aababaa

We need G = (N, T, S, R)

Non-Terminal = Z
Terminals = (a, b, e)
Start Symbol = S
Rules : Set R : S → Z
Z → aZa
Z → bZb
Z → a
Z → b
Z → e

S
aSa
aaSaa
aabSbaa
aababaa

Write Your Own Probabilistic CFG

- Weighted Palindromes
- We want to construct a grammar that creates palindromes that has more ‘a’ symbols

S
aSa
aaSaa
aaaSaaa
aaaabaaaa

We need G = (N, T, S, R)

Non-Terminal = Z
Terminals = (a, b, e)
Start Symbol = S
Rules : Set R :

Rule	Rule Probabilities
S → Z	1
Z → aZa	0.3
Z → bZb	0.15
Z → a	0.4
Z → b	0.1
Z → e	0.05

Write Your Own Probabilistic CFG

```
# Rules for creating full sentences.

1 ROOT S .
1 ROOT S !
1 ROOT is it true that S ?     # mixing terminals and nonterminals is ok.

# The basic grammar rules.  Here's what the abbreviations stand for:
#    S  = sentence
#    NP = noun phrase
#    VP = verb phrase
#    PP = prepositional phrase
#    Det = determiner (sometimes called "article")
#    Prep = preposition
#    Adj = adjective

1 S       NP VP
1 VP      Verb NP
1 NP      Det Noun
1 NP      NP PP
1 PP      Prep NP
1 Noun    Adj Noun
```

Example from Jason Eisner and Noah Smith's paper

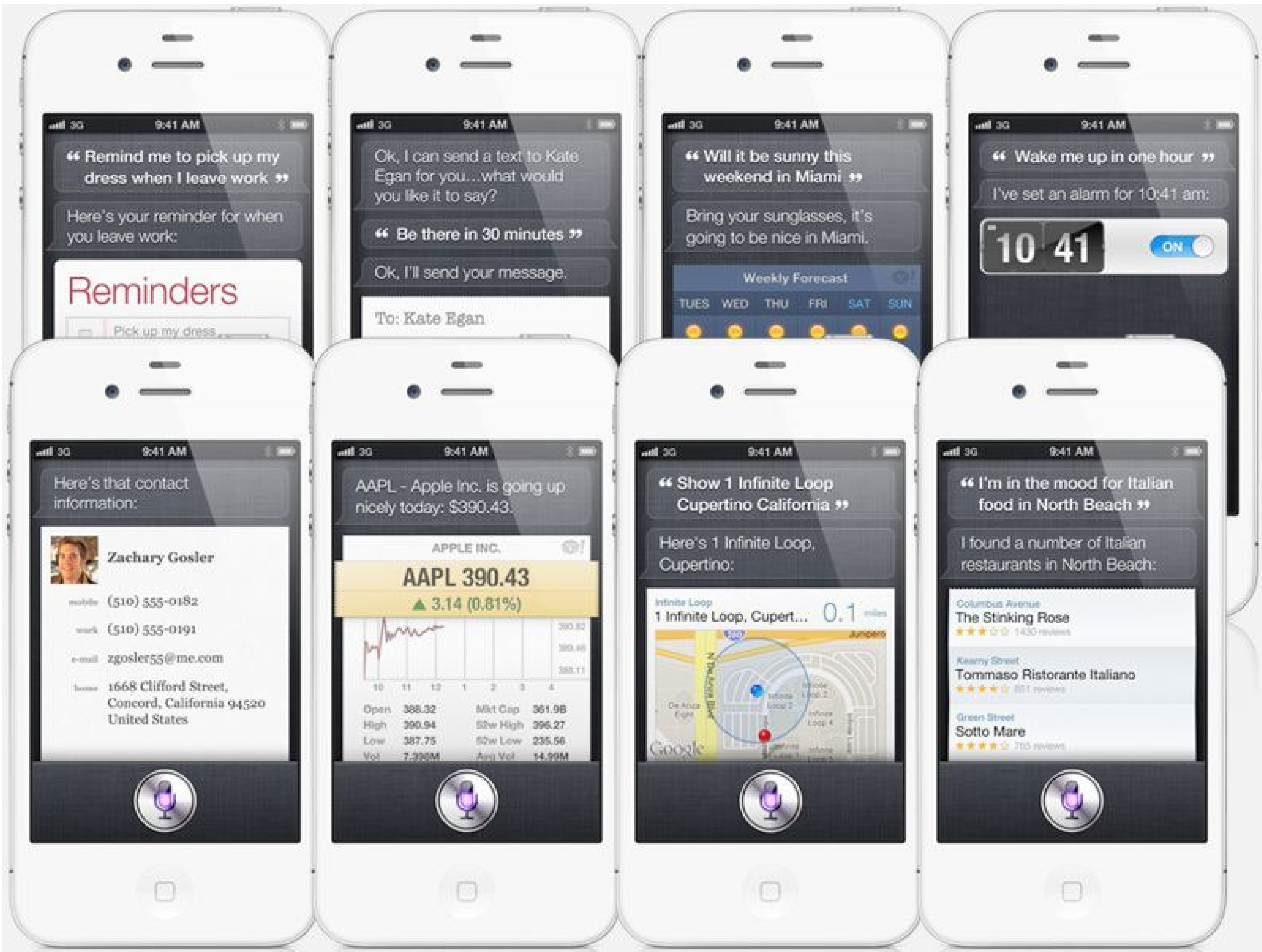
Write Your Own Probabilistic CFG

```
# Vocabulary.  Your program can see that "ate" is a terminal
# symbol because there exists no rule for rewriting it.
# Any symbol that can rewrite as a terminal (or a string of
# terminals, like "chief of staff") is called a "preterminal."  Notice
# that a preterminal is a special kind of nonterminal.

1 Verb   ate
1 Verb   wanted
1 Verb   kissed
1 Verb   understood
1 Verb   pickled

1 Det    the
1 Det    a
1 Det    every

1 Noun   president
1 Noun   sandwich
1 Noun   pickle
1 Noun   chief of staff
1 Noun   floor
```

Write Your Sentence Generator

- Possible sentence that can be generated?
  - the president ate every sandwich !
  - the president understood the chief of staff .
- Can these sentences be generated?
  - president understood the chief of staff .
  - the chief of staff pickled the chief of staff !
- Make the grammar generate more questions?

Human-Machine Dialog



Simple Grammar for Simple Dialog System

- What kind of answers should the Dialog system be able to generate?
  - Ok. I will send your message.
  - I have set an alarm for 4 a.m.
  - Here’s the contact information.
  - Your flight is at 6 p.m.
- Grammar design may be governed by the domain

Writing Grammar for Simple Dialog System

ROOT	S .
S	NP VP
VP	Verb NP
NP	Det Noun
NP	NP PP
PP	Prep NP
Noun	Adj Noun
Verb	send
Verb	set
Verb	contact
Noun	message
Noun	alarm
Noun	flight

.
.
.

Rule Probabilities

.

ROOT	S .
S	NP VP
VP	Verb NP
NP	Det Noun
NP	NP PP
PP	Prep NP
Noun	Adj Noun
Verb	send
Verb	set
Verb	contact
Noun	message
Noun	alarm
Noun	flight

.
.
.

- Count the rewrite rules from Penn Treebank corpus

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

Rule Probabilities

.

ROOT	S .	
S	NP VP	
VP	Verb NP	
NP	Det Noun	
NP	NP PP	
PP	Prep NP	
Noun	Adj Noun	
Verb	send	0.5
Verb	set	
Verb	contact	
Noun	message	
Noun	alarm	
Noun	flight	

.
.
.

- Count the rewrite rules from Penn Treebank corpus

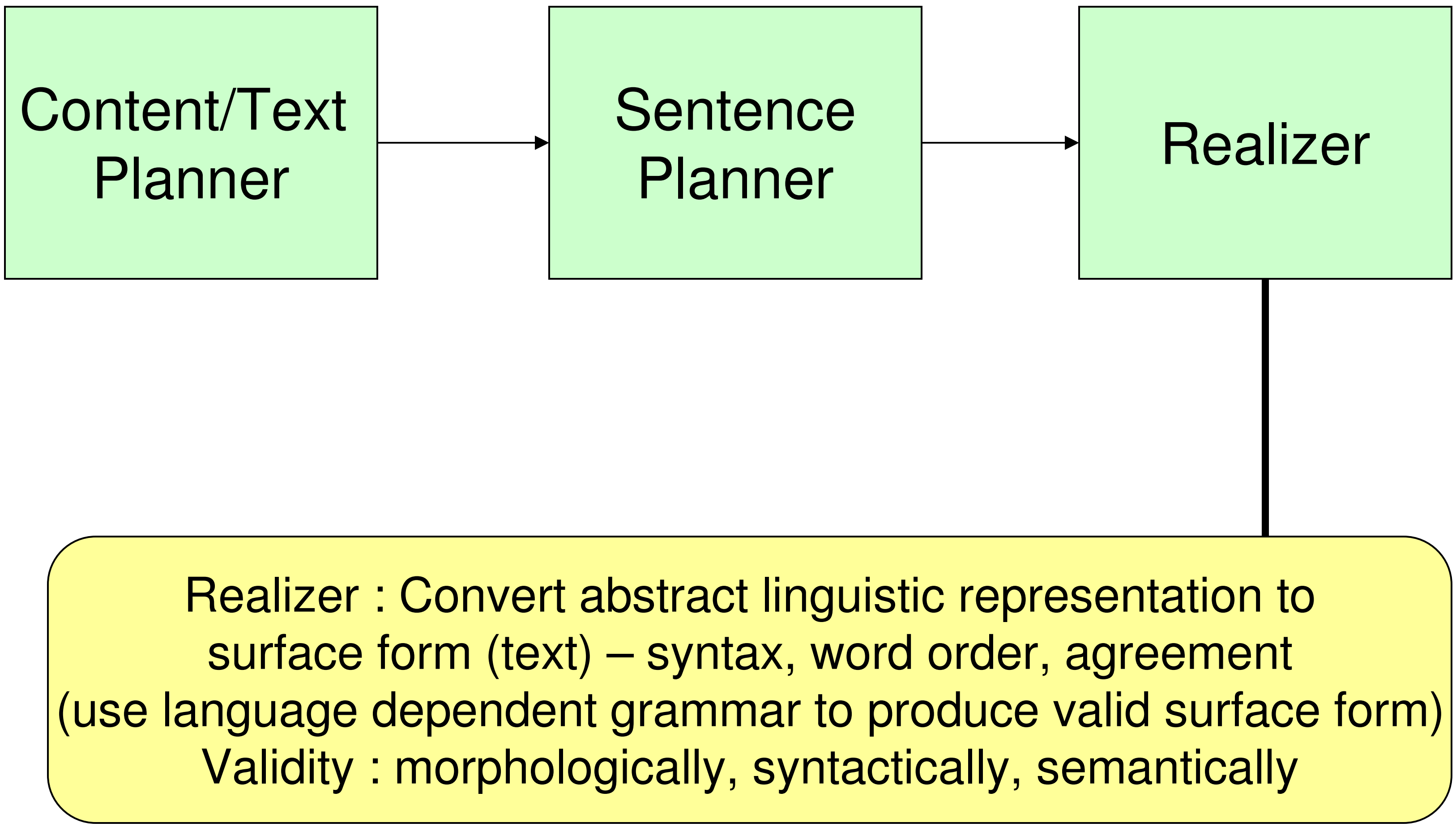
$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\sum_{\gamma} \text{count}(\alpha \to \gamma)} = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

- Verb → send [25] Rewrite count
- Verb → set [12]
- Verb → contact [13]

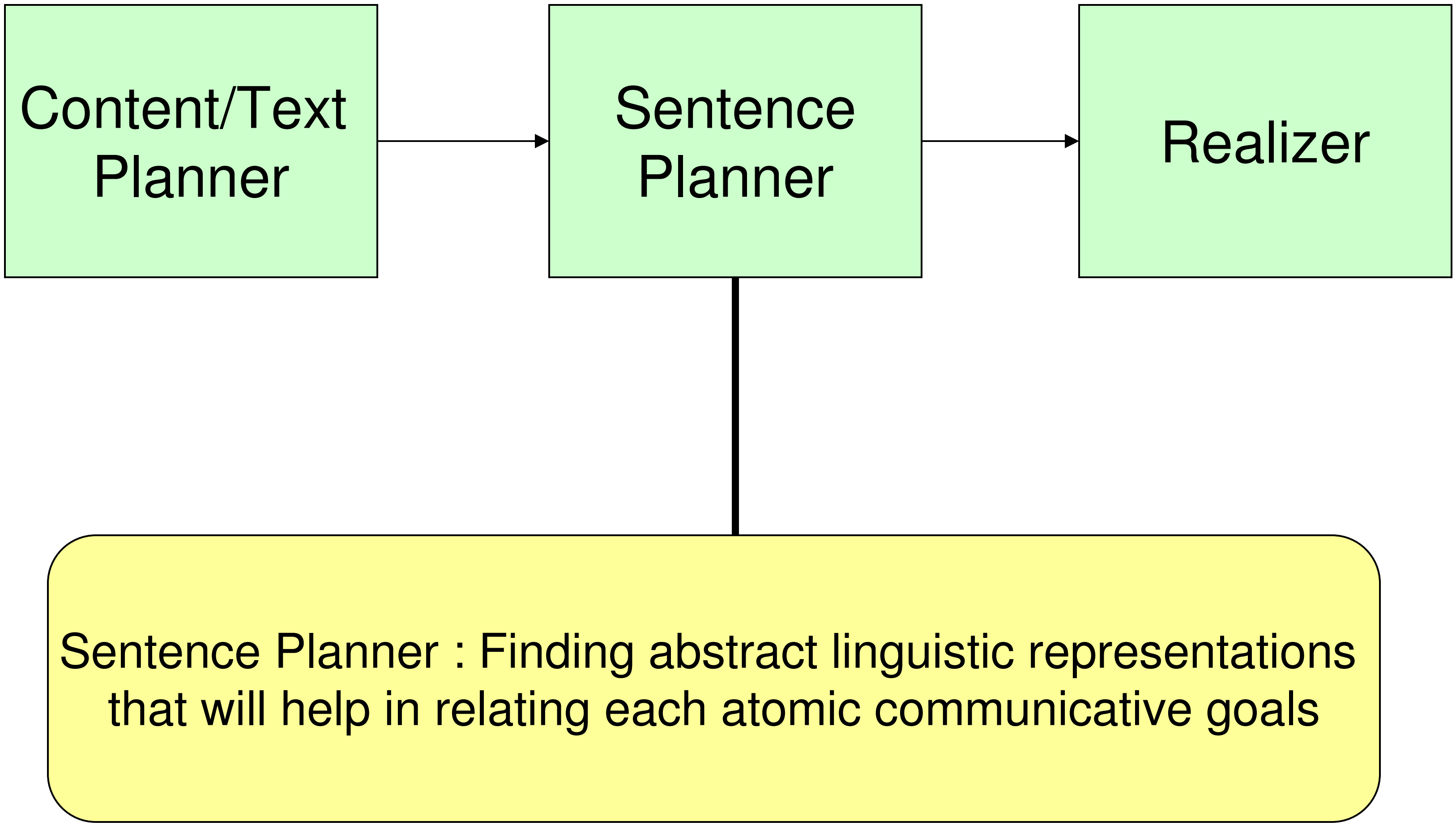
- 25/(25+12+13) for Verb→Send

  =0.5

NLG Components



NLG Components



Similarity

- While clustering documents we are essentially finding ‘similar’ documents
- How we compute similarity makes a difference in the performance of clustering algorithm
- Some similarity metrics
  - Euclidean distance
  - Cross Entropy
  - Cosine Similarity
- Which similarity metric to use?

Similarity for Words

- Edit distance
  - Insertion, deletion, substitution
  - Dynamic programming algorithm
- Longest Common Subsequence
- Bigram overlap of characters
- Similarity based on meaning
  - WordNet synonyms
- Similarity based on collocation

Similarity of Text : Surface, Syntax and Semantics

- Cosine Similarity
  - Binary Vectors
  - Multinomial Vectors
- Edit distance
  - Insertion, deletion, substitution
- Semantic similarity
  - Look beyond surface forms
  - WordNet, semantic classes
- Syntactic similarity
  - Syntactic structure
- Many ways to look at similarity and choice of the metric is important for the type of clustering algorithm we are using

NLP/ML Tools

- Weka
- Stanford NLP Tools
  - Parsers, taggers, chunkers, NE recognizer
- Ratnaparkhi's NE Tagger
- NLTK
- OpenNLP