School of CIT –
Social Computing
Research Group

Technical
University
of Munich

Natural Language Processing
IN2361

Prof. Dr. Georg Groh

Chapter 13
Constituency Parsing

- content is based on [1]
- certain elements (e.g. equations or tables) were taken over or taken over in a modified form from [1]
- citations of [1] or from [1] are omitted for legibility
- errors are fully in the responsibility of Georg Groh
- BIG thanks to Dan and James for a great book!

Syntactic Parsing

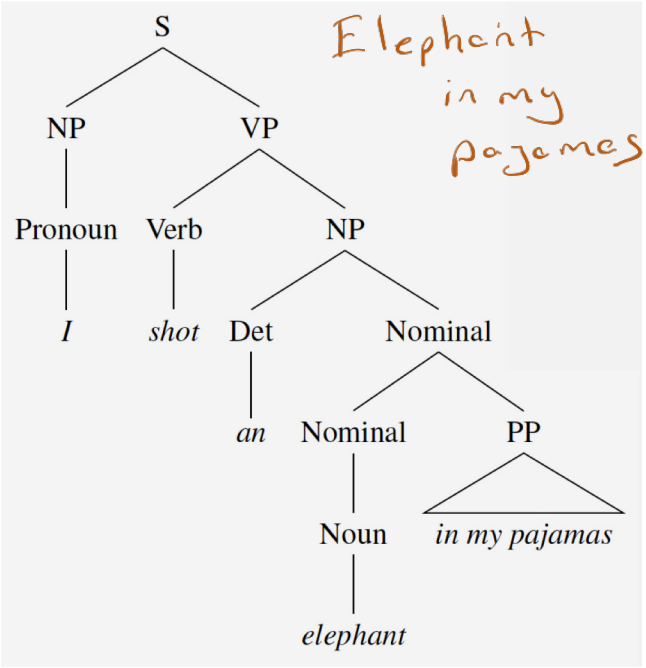
- Syntactic parsing: sentence → parse tree

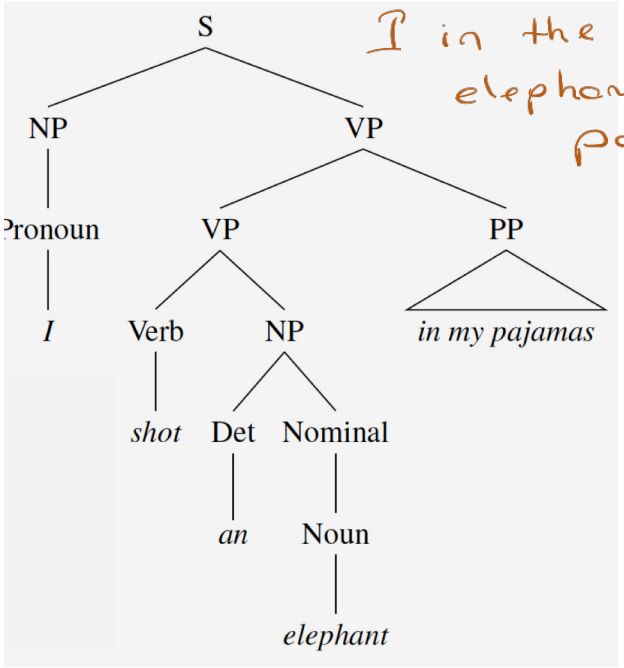
- applications:
  - grammar checking (e.g. in Word)
  - information extraction + question answering
    What books were written by British women authors before 1800?
  - etc

- classic "conflict":
  - symbolic reasoning (syntactic + semantic parsing + rules, logic) vs
  - sub-symbolic reasoning (deep learning)

Ambiguity

- (structural) Ambiguity: assign more than one parse tree to a sentence
  - attachment ambiguity: a particular constituent can be attached to the parse tree at more than one place



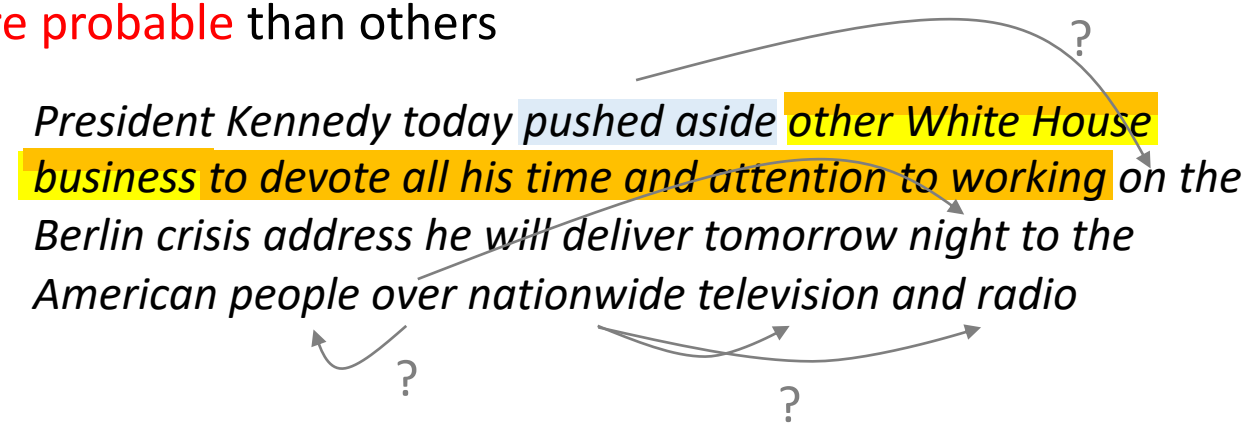


  - coordintion ambiguity: join different sets of phrases by *and*

[old [men and women]] [old men] and [women]

Disambiguation

- often some parse trees are semantically or statistically or contextually more probable than others



→ integrate statistical info into parsers to produce most plausible parse tree (see appendix B)

CYK Algorithm

- Cocke Younger Kasami dynamic programming algorithm for parsing CF grammars
  - dynamic programming (as in MinEditDistance, Viterbi, Forward etc. before): systematically fill in tables of solutions to sub-problems, at the end: assemble solution form these
  - two parts: recognizer + actual parser
  - typically: convert to Chomsky Normal Form (CNF) first

CF Grammars: Conversion to CNF

- Convert into Chomsky Normal Form (CNF) : $A \rightarrow B\,C$
  $A \rightarrow w$

1. eliminate terminals from non-pure right hand sides

$$INF\text{-}VP \rightarrow to\ VP \Longrightarrow \begin{cases} INF\text{-}VP \rightarrow TO\ VP \\ TO \rightarrow to \end{cases}$$

2. eliminate unit productions $A \rightarrow B$

   if $A \overset{*}{\Rightarrow} B$ by a chain of one or more unit productions and $B \rightarrow \gamma$ is a non-unit production in our grammar, then we add $A \rightarrow \gamma$ for each such rule in the grammar and discard all the intervening unit productions

3. iteratively shorten long productions

$$A \rightarrow B\,C\,\gamma \Longrightarrow \begin{cases} A \rightarrow X1\,\gamma \\ X1 \rightarrow B\,C \end{cases}$$

CF Grammars: Conversion to CNF

$\mathscr{L}_1$ Grammar	$\mathscr{L}_1$ in CNF
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

Lexicon

$Det \rightarrow that \mid this \mid the \mid a$
$Noun \rightarrow book \mid flight \mid meal \mid money$
$Verb \rightarrow book \mid include \mid prefer$
$Pronoun \rightarrow I \mid she \mid me$
$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$Aux \rightarrow does$
$Preposition \rightarrow from \mid to \mid on \mid near \mid through$

CYK Recognizer

- organize possible rule applications into upper triangle (i.e. excluding the diagonal) of an (n+1) x (n+1) matrix (n: sentence length)

- first super-diagonal (cells [0,1], [1,2], [2,3], …, [n-1,n]) contain all possible POS of the words of the sentence. (words are the terminals)

- other than first super-diagonal: cell [i,j] contains the set of all non-terminals that represent all possible constituents spanning positions i through j of the sentence (position-notation: *0 Book 1 that 2 flight 3 tomorrow 4*)

- **work up+right** from first super diagonal to left upper corner (cell [0,n]): fill cell [i,j] with all A from rules $A \rightarrow B\ C$ where $B \in$ cell [i,k] (to the left) and $C \in$ cell [k,j] (below) for all possible k $(i<k<j)$. (for actual parser: also store k, B, C together with all possible multiple occurrences A_n of A in cell [i,j])

function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do** iterate over all columns, left to right

  for all $\{A \mid A \rightarrow words[j] \in grammar\}$ fill element of first superdiagonal with all possible POS A→w for the word

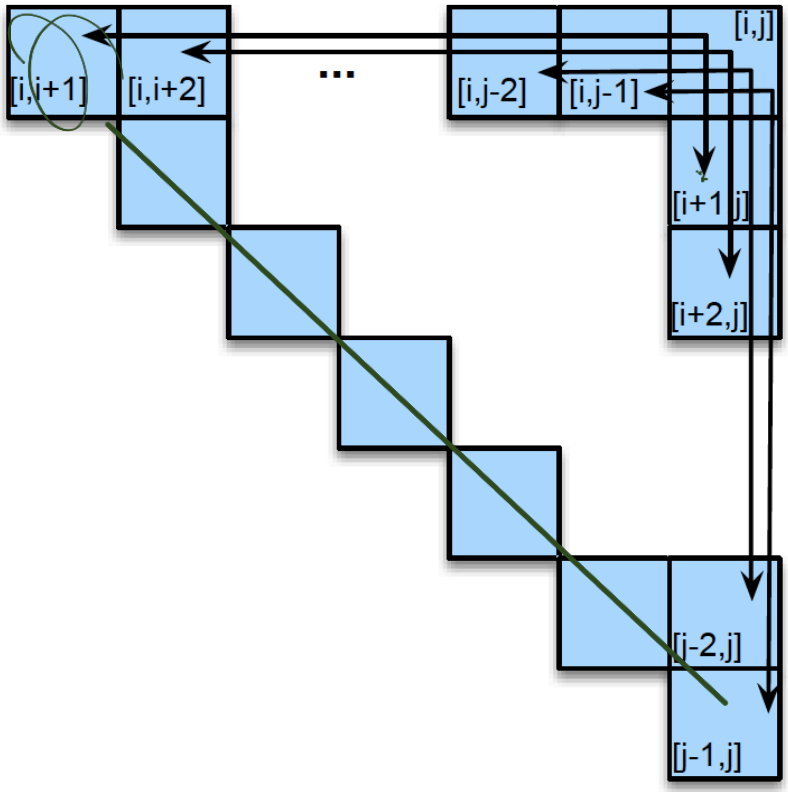
    $table[j-1, j] \leftarrow table[j-1, j] \cup A$

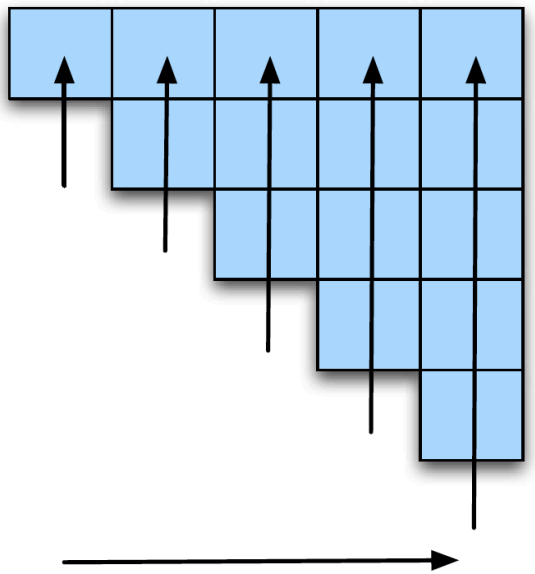
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do** iterate over the rows (the elements in the current column), bottom up

    for $k \leftarrow i+1$ **to** $j-1$ **do** for current (i,j), consider all possible k's to the left & down for non-terminal rule applications

      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$

        $table[i,j] \leftarrow table[i,j] \cup A$



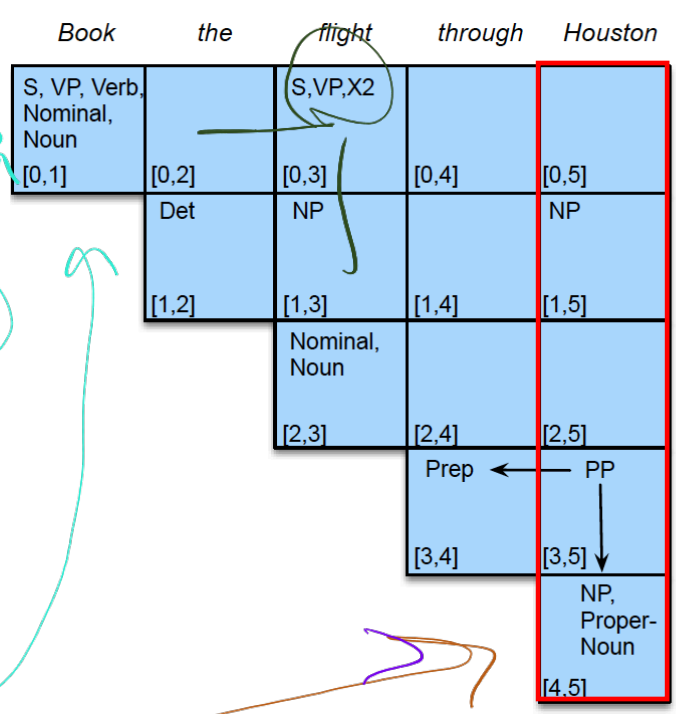


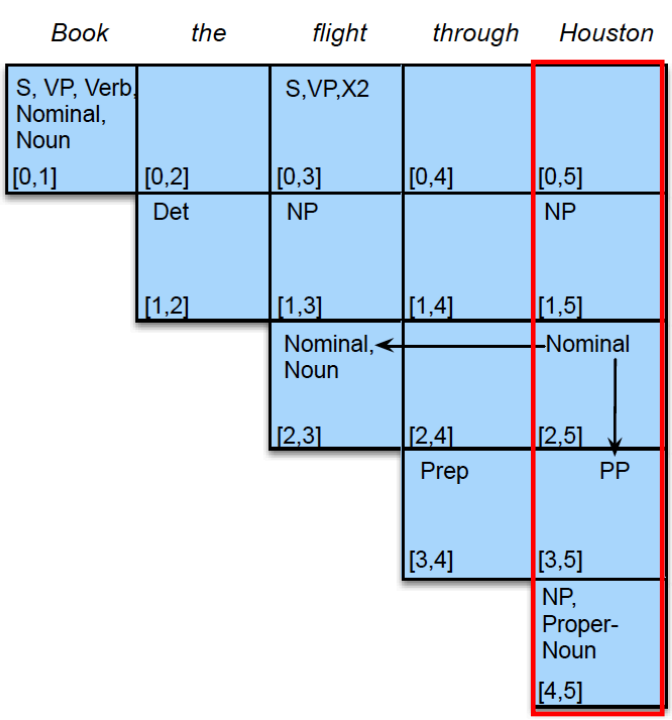
$\mathcal{L}_1$ in CNF

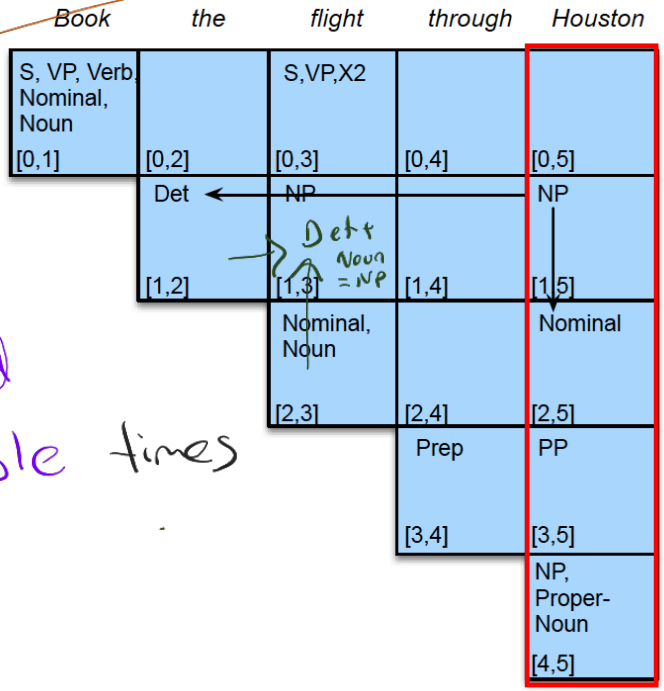
$S \rightarrow NP\ VP$
$S \rightarrow X1\ VP$
$X1 \rightarrow Aux\ NP$
$S \rightarrow book \mid include \mid prefer$
$S \rightarrow Verb\ NP$
$S \rightarrow X2\ PP$
$S \rightarrow Verb\ PP$
$S \rightarrow VP\ PP$
$NP \rightarrow I \mid she \mid me$
$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$
$VP \rightarrow X2\ PP$
$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

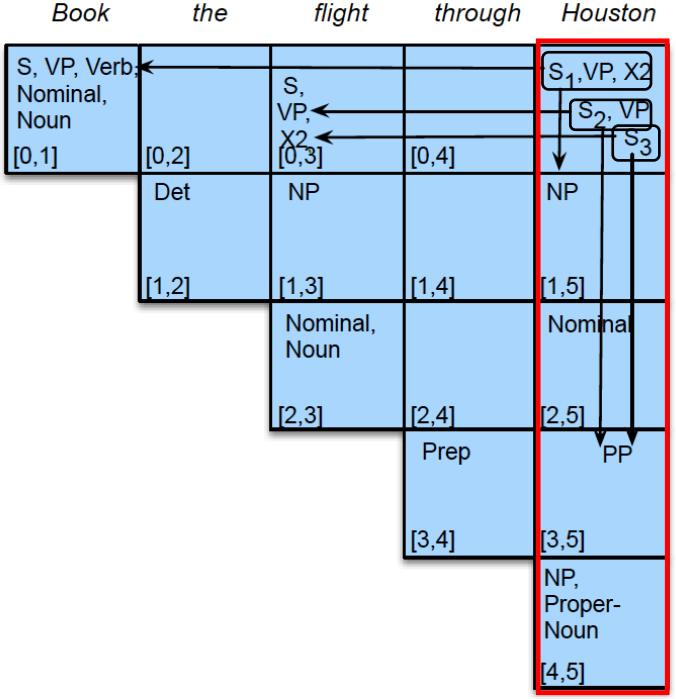
Lexicon

$Det \rightarrow that \mid this \mid the \mid a$
$Noun \rightarrow book \mid flight \mid meal \mid money$
$Verb \rightarrow book \mid include \mid prefer$
$Pronoun \rightarrow I \mid she \mid me$
$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$Aux \rightarrow does$
$Preposition \rightarrow from \mid to \mid on \mid near \mid through$

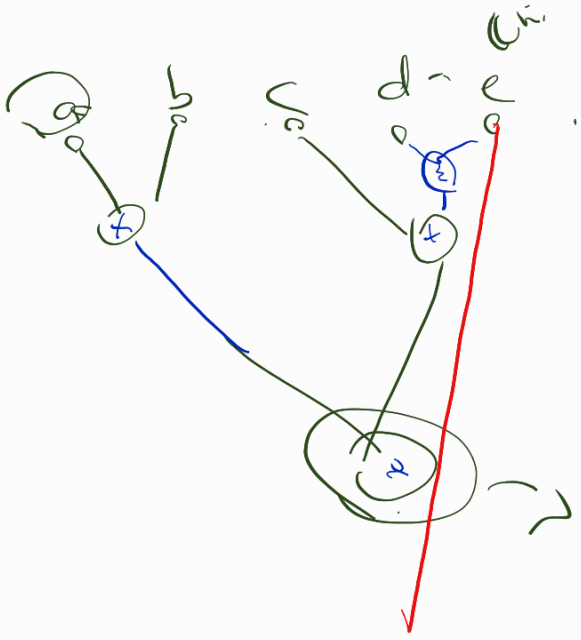








Don't use word double times



1-) Dont have overlaps

2-) Relations should be consecutive

3-) Dont Forget the Rope

CYK Parser

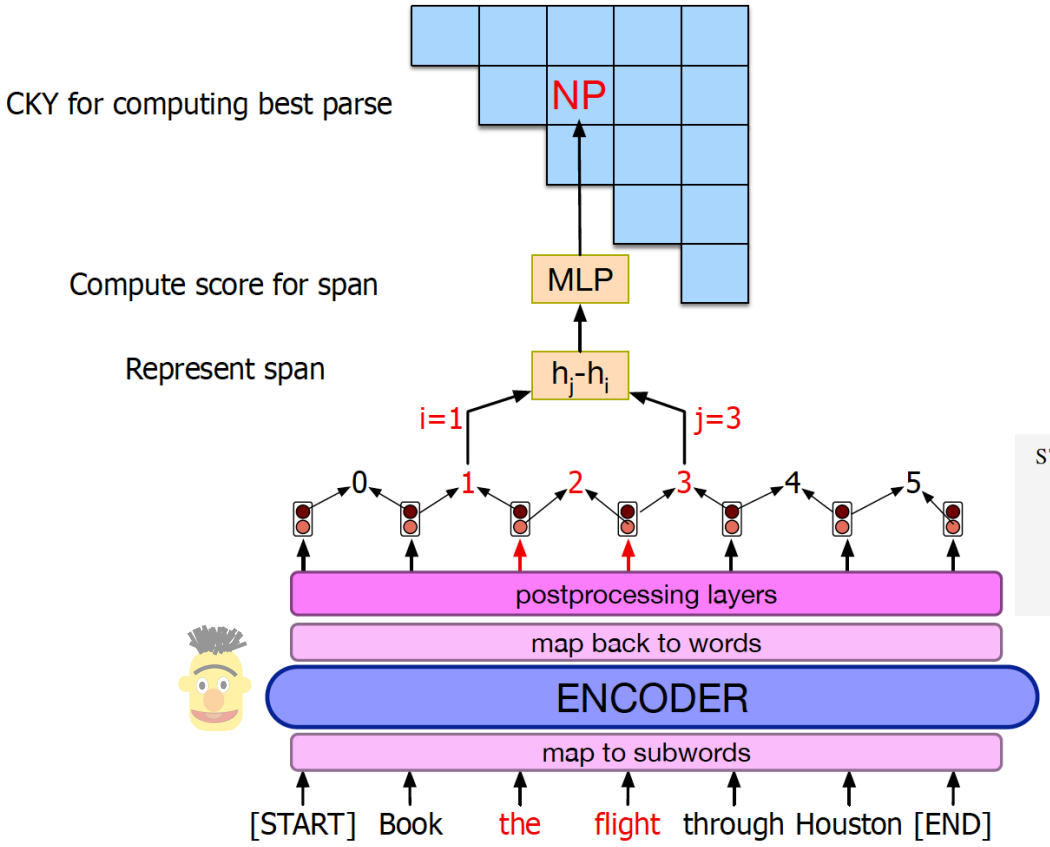
- For every instance S_i of S in cell [0,n]: recursively repeat:
  - trace back all possible rules (productions) S → B C that led to this S_i
  - for each such rule, trace back all instances of B in its corresponding cell and trace back all instances of C in its corresponding cell
  - and so forth

- For every instance S_i of S in cell [0,n] we may get an exponential number of possible parse trees

- resulting CNF parse trees may be linguistically unwieldy → convert back into original grammar or user adapted parsing algorithm that can parse non-CNF CF grammars

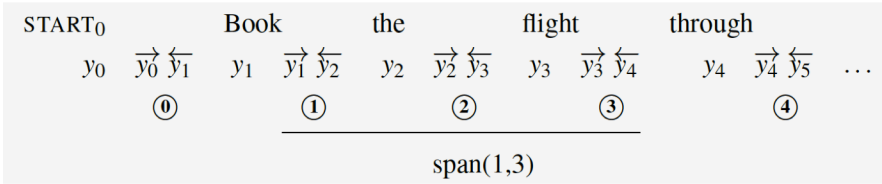
Span-Based Neural Constituency Parsing

- CYK produces a lot of possible trees → disambiguate? → PCFGs *or* (span-based) neural approaches ☺
- neural span-based constituency parsing (neural CKY): assign scores to possible trees



score for each labeled constituent span $s(i, j, l)$

$$v(i,j) = [\overrightarrow{y_j} - \overrightarrow{y_i} \; ; \; \overleftarrow{y_{j+1}} - \overleftarrow{y_{i+1}}]$$



Span-Based Neural Constituency Parsing

- integrating labelled span-scores $s(i, j, l)$ into parse tree via:

$$T = \{(i_t, j_t, l_t) : t = 1, \ldots, |T|\}$$

$$s(T) = \sum_{(i,j,l) \in T} s(i, j, l)$$

$$\hat{T} = \operatorname*{argmax}_T s(T)$$

- to actually compute argmax:
  - use greedy: choose best scoring span each (these spans might not correctly „combine“ to a parse tree)
  - or use CYK algorithm idea:
    (no grammar needed
    (grammar is implictly
    learned by NN via finding
    good intermediate representations)):

$$s_{\text{best}}(i, i+1) = \max_l s(i, i+1, l)$$

$$s_{\text{best}}(i, j) = \max_l s(i, j, l) + \max_k [s_{\text{best}}(i, k) + s_{\text{best}}(k, j)]$$

Performance Evaluation of Classifiers - PARSEVAL

$$\textbf{labeled recall:} = \sum_s \frac{\#\text{ of correct constituents in hypothesis parse of } s}{\#\text{ of correct constituents in reference parse of } s}$$

$$\text{or} \quad \frac{\sum_s \#\text{ of correct constituents in hypothesis parse of } s}{\sum_s \#\text{ of correct constituents in reference parse of } s}$$

$$\textbf{labeled precision:} = \sum_s \frac{\#\text{ of correct constituents in hypothesis parse of } s}{\#\text{ of total constituents in hypothesis parse of } s}$$

$$\text{or} \quad \frac{\sum_s \#\text{ of correct constituents in hypothesis parse of } s}{\sum_s \#\text{ of total constituents in hypothesis parse of } s}$$

$$F_1 = \frac{2PR}{P+R}$$

cross bracketing errors: e.g. predicted: (A(BC)) but correct ((AB)C)

- modern parsers: F1 > 0.9 & < 0.01 cross bracketing errors
- For some applications (and e.g. for lexical parsers such as CCG): more interested in extracting appropriate predicate-argument relations or grammatical dependencies, rather than a specific derivation → better metric of parser usefulness for further semantic processing → use alternative evaluation metrics

Chunking (Partial Parsing)

- Chunking: identifying and classifying flat, non-overlapping segments of a sentence (chunks)

  chunks constitute the basic, non-recursive phrases corresponding to the major content bearing POS (noun phrases, verb phrases, adjective phrases, and prepositional phrases)

  [$_{NP}$ The morning flight] [$_{PP}$ from] [$_{NP}$ Denver] [$_{VP}$ has arrived.]

- criteria for a chunk:
  - chunks do not contain other chunks of the same (or other) type
  - boundaries of a chunk: headword + pre-head modifiers;
    ignore post-head modifiers → assignment ambiguities due to overlap
    aka post-head(i-1) ∩ pre-head(i) are vastly reduced

Chunking as Classification Task

- Chunking: sequence classification (sequence labeling) task
- chunking: find limits for a chunk and classify it → BIO notation
  - for each of the n original labels: introduce
    - B version (beginning) and
    - I version (internal);
  - additional label: O (outside)
  - n original labels → 2n+1 new labels

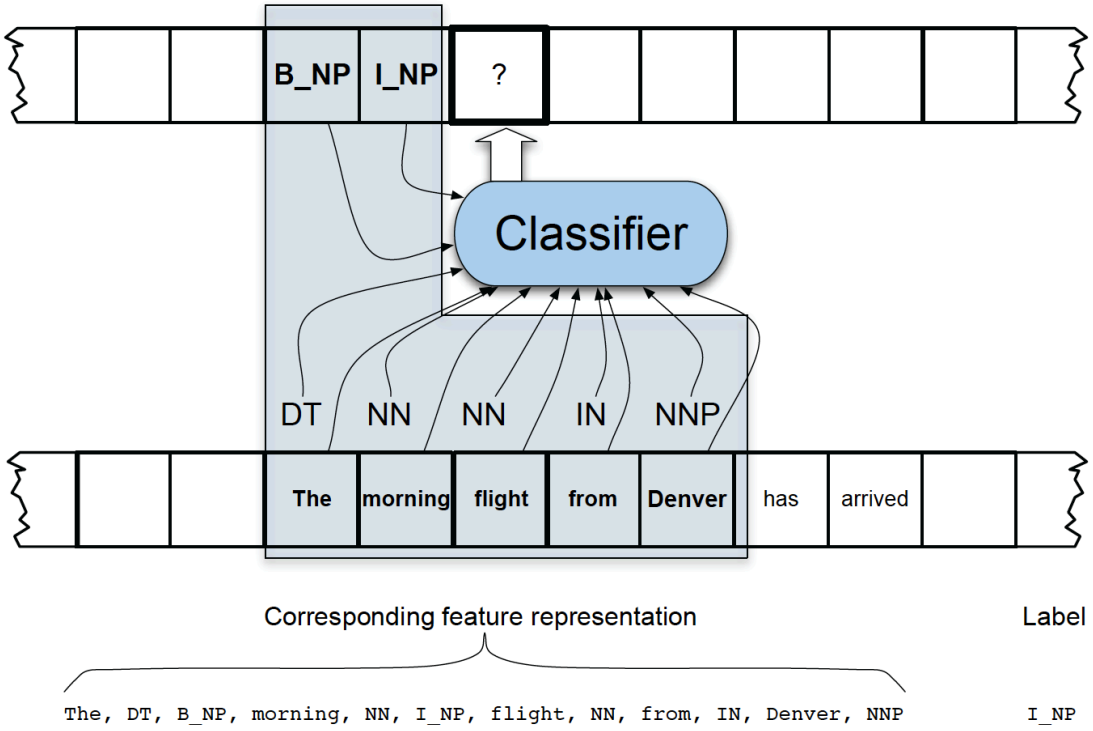
The morning flight from Denver has arrived
B_NP I_NP I_NP B_PP B_NP B_VP I_VP

The morning flight from Denver has arrived.
B_NP I_NP I_NP O B_NP O O

for an NP-only chunker

Chunking as Classification Task

- 2n+1 class labels;
  features per word in the sequence: from frame surrounding the word: e.g.
  - the word itself plus the two preceding words,
  - their parts-of-speech
  - the chunk tags of the preceding inputs in the window
  - ….



Ground Truth for Chunking

- from parse trees in Treebanks:
  - concentrate on the non-terminals corresponding to the chunks that we are interested in (e.g. NP, VP)
  - boundary determination: in the respective phrase: find the head word (with head word detection rules) + all words preceding it; ignore words after head word

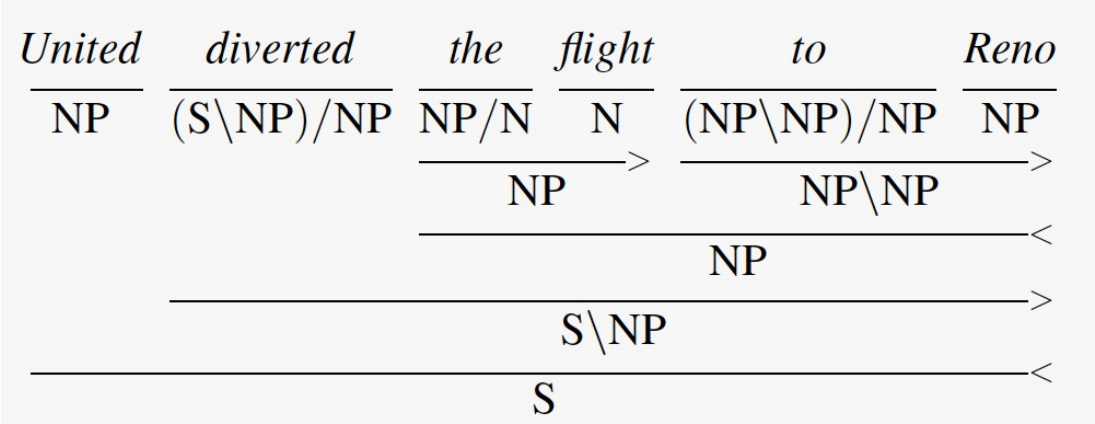
CCG Parsing

- ambiguity example: *United diverted the flight to Reno* :

traditional CFG: choice of internal rule determines parse

$Nominal \rightarrow Nominal\ PP$	United diverted [the flight to Reno]
$VP \rightarrow VP\ PP$	United [diverted to Reno] the flight
$VP \rightarrow Verb\ NP\ PP$	United [diverted the flight to Reno]

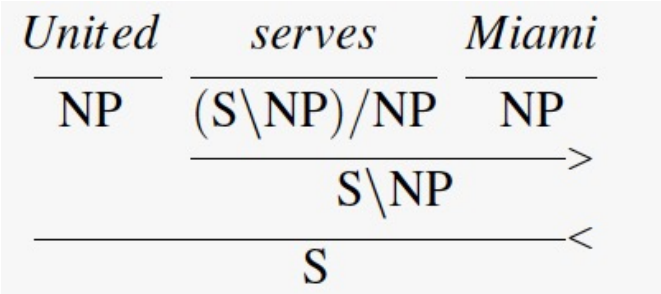
traditional CCG (lexicalized grammar): choice of lexical rule determines parse



Repetition: Combinatory Categorial Grammars

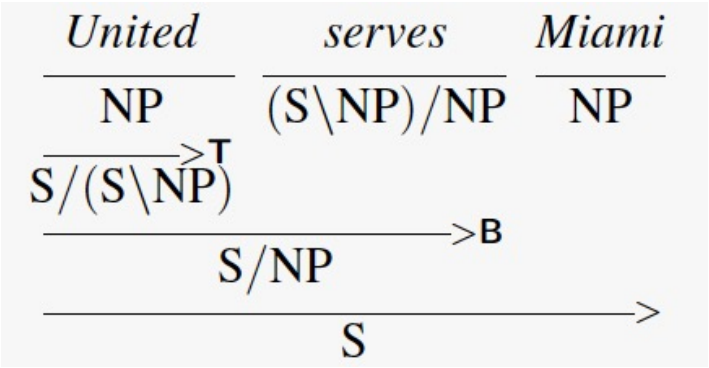
- $X/Y \;\; Y \;\Rightarrow\; X$ $\quad$ X/Y : function that seeks its argument to the right
  $Y \;\; X\backslash Y \;\Rightarrow\; X$ $\quad$ $X\backslash Y$: function that seeks its argument to the left

- Lexicon: $\quad$ *flight* : N
  Miami : NP
  cancel : $(S\backslash NP)/NP$
  (transitive verb (one object))



- function composition B: $\quad$ $X/Y \;\; Y/Z \;\Rightarrow\; X/Z$
  $Y\backslash Z \;\; X\backslash Y \;\Rightarrow\; X\backslash Z$

- type raising T: $\quad$ $X \;\Rightarrow\; T/(T\backslash X)$
  $X \;\Rightarrow\; T\backslash(T/X)$
  with T any category

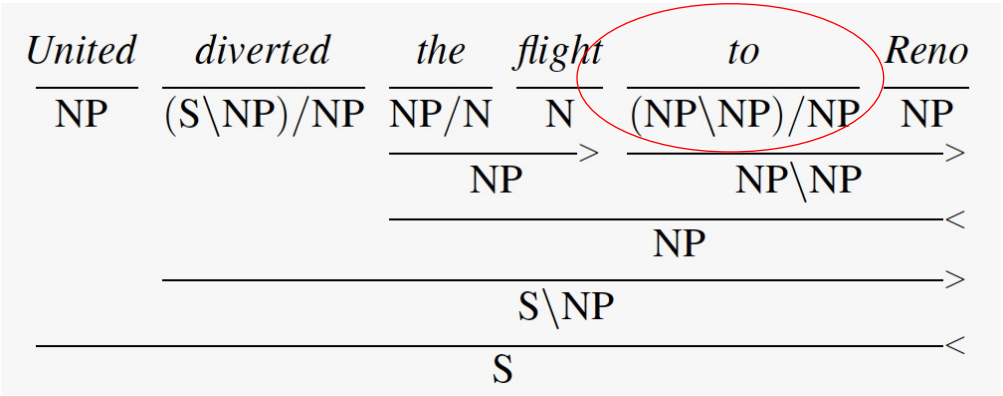


CCG Parsing

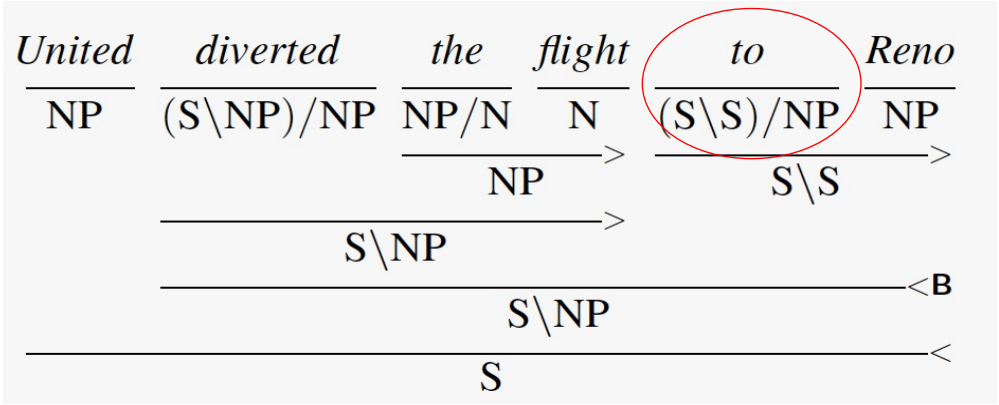
- ambiguity example (ctd.): *United diverted the flight to Reno* :

  traditional CCG (lexicalized grammar): choice of lexical rule determines parse (ctd.)

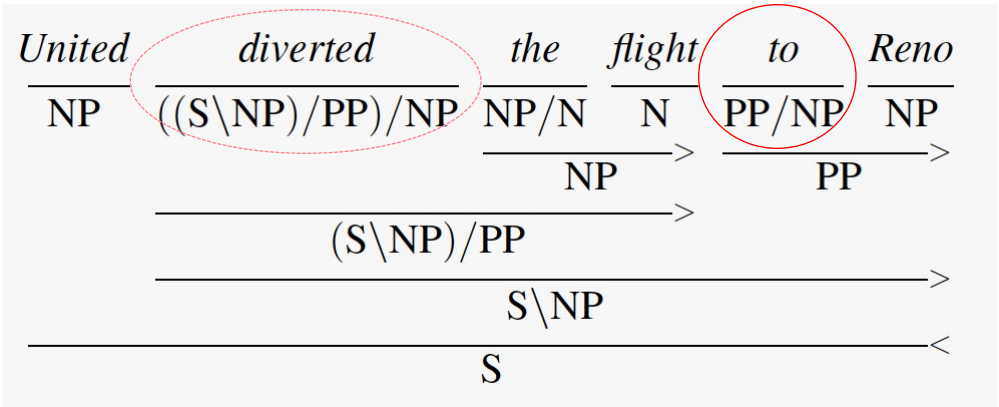
to Reno modifies *flight*



to Reno modifies the verb-phrase: *to* starting a subordinate sentence



divert as di-transitive verb and *to Reno* as simple prepositional phrase argument



Supertagging

- CYK for CCGs: a lot of possible lexical categories (lexical rules) for each word AND very few but very generally applicable internal rules → combinatorial explosion: each cell [i,j] will exist in a lot of variations [i,j,$A^{(n)}$] (most of them useless (zombie constitutents)) → CYK does not work!

- → possible remedy: accurately assess and exploit only the most likely lexical categories for each word: Supertagging as a previous step before parsing (→less combinatorial explosion) and / or applying heuristic search algorithms (e.g. A*)

Supertagging

- in CCG bank of CCG parses: over 1000 lexical categories: keep only those that occur more than 10 times → 425 supertags (lexical categories) (compared to 45 POS tags in the Penn Treebank)

- Supertagging (analogous to POS-tagging): sequence labeling problem: approach as before: discriminative CRF model: train model with MLE; for argmax: Viterbi

$$\begin{aligned}
\hat{T} &= \operatorname*{argmax}_T P(T|W) \\
&= \operatorname*{argmax}_T \prod_i P(t_i|w_{i-l}^{i+l}, t_{i-k}^{i-1}) \\
&= \operatorname*{argmax}_T \prod_i \frac{\exp\left(\sum_i w_i f_i(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}{\sum_{t' \in \text{tagset}} \exp\left(\sum_i w_i f_i(t', w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}
\end{aligned}$$

e.g. use symmetric window of l=k=2

Supertagging

- However: even with additional features f_i : large number of supertags and high per-word ambiguity → too many errors by Viterbi

- Instead of most likely supertag-sequence (by Viterbi) we actually need a probability distribution over possible supertags for each word

United	serves	Denver
N/N: 0.4	$(S\backslash NP)/NP$: 0.8	NP: 0.9
NP: 0.3	N: 0.1	N/N: 0.05
S/S: 0.1	...	...
$S\backslash S$: .05		
...		

→ compute it with a modified version of forward backward algorithm (see chapter Appendix A) or with RNNs (see later)

RNNs for Supertagging

- Recurrent Neural Networks (RNNs) also do well for this task:
  - using vector-valued features for words (word-embeddings)
  - use entire sequence for assessing a word instead of sliding window
  - avoid high-level features (such as POS tags) because of error propagation

CCG Parsing using the A* Algorithm

- A*-algorithm: complete (finds a solution if ex.), optimal (finds best solution), optimally complex (no less complex alg. with same heuristic), informed (using heuristic) search algorithm
  - iteratively choose next state n with optimal estimated cost f(n)= g(n)+h(n), where g(n) actual current actual cost to n from start node and h(n) non-overestimating estimate to target state from n
  - each state: either in closed list (treated completely, g correct), or open list (known but not yet fully treated, g may shrink), or unknown

CCG Parsing using the A* Algorithm

```
function CCG-ASTAR-PARSE(words) returns table or failure

  supertags ← SUPERTAGGER(words)
  for i ← from 1 to LENGTH(words) do
      for all {A | (words[i], A, score) ∈ supertags}
          edge ← MAKEEDGE(i − 1, i, A, score)
          table ← INSERTEDGE(table, edge)
          agenda ← INSERTEDGE(agenda, edge)
  loop do
      if EMPTY?(agenda) return failure
      current ← POP(agenda)
      if COMPLETEDPARSE?(current) return table
      table ← INSERTEDGE(chart, edge)
      for each rule in APPLICABLERULES(edge) do
          successor ← APPLY(rule, edge)
          if successor not ∈ in agenda or chart
              agenda ← INSERTEDGE(agenda, successor)
          else if successor ∈ agenda with higher cost
              agenda ← REPLACEEDGE(agenda, successor)
```

edges represent spans of symbols (==completed constituents)
$w_{i,j}$ (words from i to j)

agenda ←→ open-list

table: (current) parse table

chart: (current) upper triangle of n+1 x n+1 PCYK matrix

CCG Parsing using the A* Algorithm

how to compute heuristics f(n)?

- simplifying assumption: ignoring usage of other rules, we assume that probability of a derivation D of a sentence $S = (w_1, w_2, \dots)$ = product of probabilities of supertags $T = (t_1, t_2, \dots)$ for words chosen in D

$$\begin{aligned} P(D,S) &= P(T,S) \\ &= \prod_{i=1}^{n} P(t_i|w_i) \end{aligned}$$

taking negative log → convert probability to a cost:

$$\begin{aligned} P(D,S) &= P(T,S) \\ &= \sum_{i=1}^{n} -\log P(t_i|w_i) \end{aligned}$$

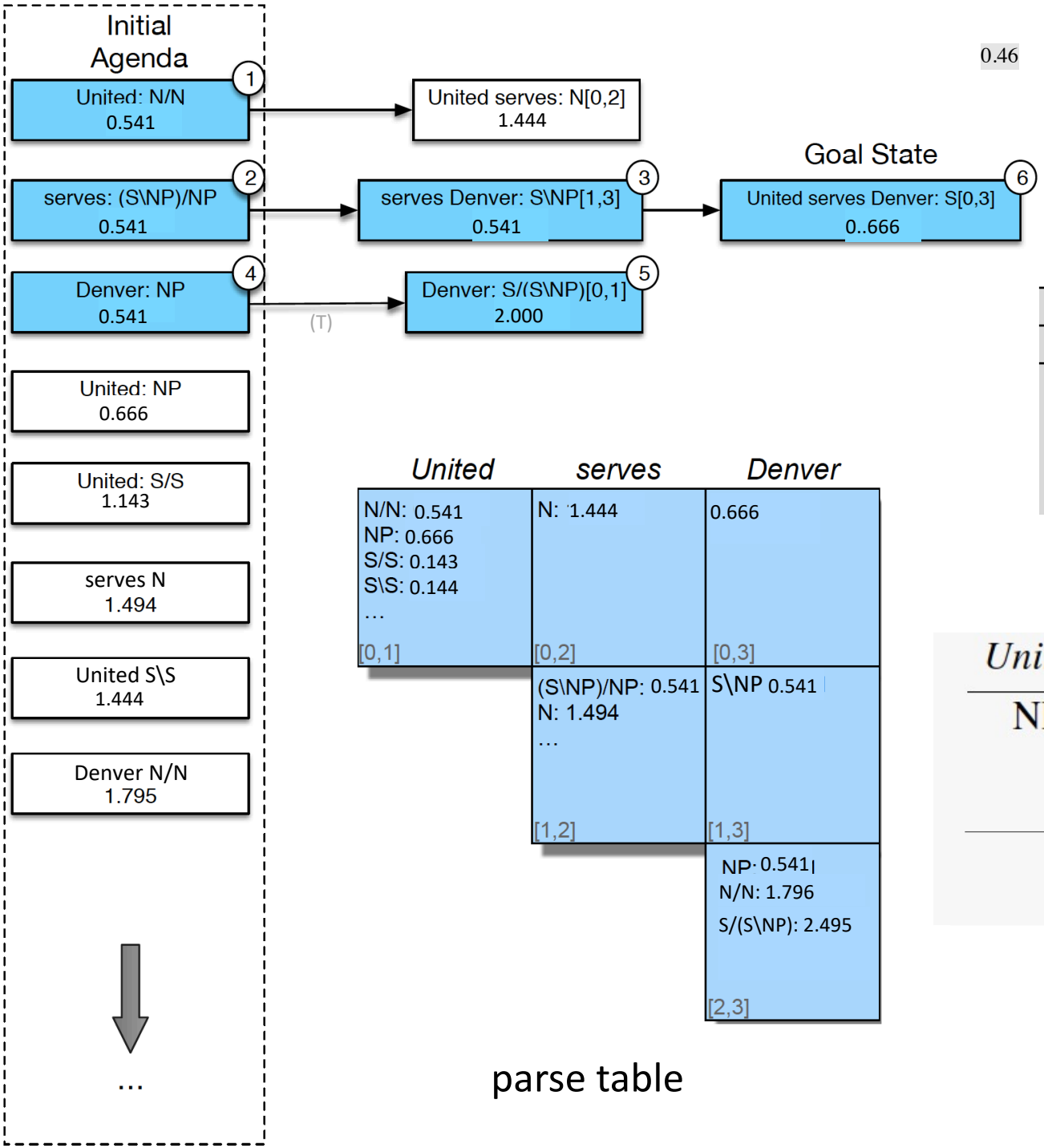
CCG Parsing using the A* Algorithm

- → $f(n) = g(n) + h(n)$ = inside cost + outside cost:
  - g(n) cost of current span (sub-sequence ("sub-string") of words currently considered)
  - h(n) estimated cost for words outside of span: assume that each of these outside words is assigned most probable supertag (→ never overestimate costs)
- →

$$f(w_{i,j}, t_{i,j}) = g(w_{i,j}) + h(w_{i,j})$$

$$= \sum_{k=i}^{j} -\log P(t_k|w_k) +$$

$$\sum_{k=1}^{i-1} \min_{t \in tags}(-\log P(t|w_k)) + \sum_{k=j+1}^{N} \min_{t \in tags}(-\log P(t|w_k))$$



0.46

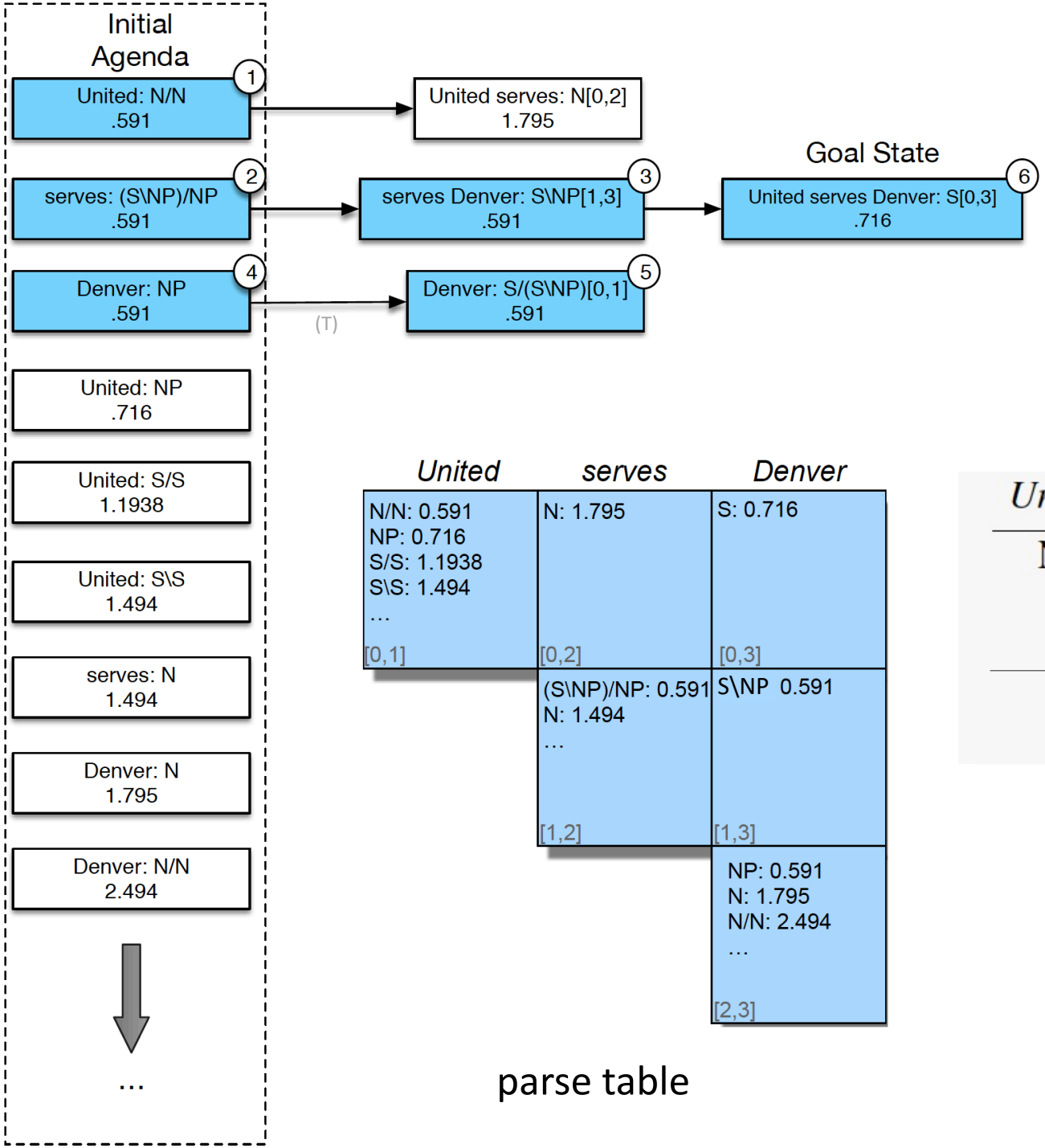
United	serves	Denver
N/N: 0.4	$(S\backslash NP)/NP$: 0.8	NP: 0.9
NP: 0.3	N: 0.1	N/N: 0.05
S/S: 0.1	…	…
$S\backslash S$: .05		S/(S\NP) 0.01
…		

$-\log_{10} P$		
United	serves	Denver
N/N: 0.398	(S\NP)/NP 0.097	NP: 0.046
NP: 0.523	N: 1.000	N/N: 1.301
S/S: 1.000		S/(S\NP) 2.000
S\S: 1.301		

United	*serves*	*Denver*
N/N: 0.541 NP: 0.666 S/S: 0.143 S\S: 0.144 … [0,1]	N: 1.444 [0,2]	0.666 [0,3]
	(S\NP)/NP: 0.541 N: 1.494 … [1,2]	S\NP 0.541 [1,3]
		NP 0.541 N/N: 1.796 S/(S\NP): 2.495 [2,3]

parse table

United	*serves*	*Denver*
NP	(S\NP)/NP	NP
	S\NP (>)	
	S (<)	



United	*serves*	*Denver*
N/N: 0.591 NP: 0.716 S/S: 1.1938 S\S: 1.494 … [0,1]	N: 1.795 [0,2]	S: 0.716 [0,3]
	(S\NP)/NP: 0.591 N: 1.494 … [1,2]	S\NP 0.591 [1,3]
		NP: 0.591 N: 1.795 N/N: 2.494 … [2,3]

parse table

$$\dfrac{\dfrac{\textit{United}}{\text{NP}} \quad \dfrac{\dfrac{\textit{serves}}{(\text{S}\backslash\text{NP})/\text{NP}} \quad \dfrac{\textit{Denver}}{\text{NP}}}{\text{S}\backslash\text{NP}}\,>}{\text{S}}\,<$$

This is the original version from the book.
The actual costs are not in 100 % correspondence to the table on slide 30 (assuming $\log_{10}$ as stated in caption of figure 14.12)

Bibliography

(1) Dan Jurafsky and James Martin: Speech and Language Processing (3rd ed. draft, version Jan 2022); Online: https://web.stanford.edu/~jurafsky/slp3/ (URL, Oct 2022); this slideset is especially based on chapter 13.

Recommendations for Studying

- minimal approach:

  work with the slides and understand their contents! Think beyond instead of merely memorizing the contents

- standard approach:

  minimal approach + read the corresponding pages in Jurafsky [1]

- interested students

  == standard approach