Statistical Constituency Parsing

March 27, 2020

vázlat

1. Valószínűségi környezetfüggetlen nyelvtan (PCFG)
2. A PCFG-k elemzése valószínűségi CKY algoritmussal
3. PCFG szabály-valószínűségek felderítése
4. PCFG nyelvtanok problémái
5. PCFG nyevtanok javítása: nem terminálisok feldarabolása
6. Valószínűségi lexikalizált CFG-k
7. Valószínűségi CCG elemzés
8. Elemzések kiértékelése

Valószínűségi környezetfüggetlen nyelvtan (PCFG)

Mi az a PCFG?

- a mondatok általában többértelműek (pl. attachment vagy mellérendelés értelmében)
- a többértelműségeket a CKY algoritmus képes reprezentálni, de kezelni nem
- ezért: valószínűségek hozzárendelése a környezetfüggetlen nyelvtan (CFG) szabályaihoz
    - CFG $\rightarrow$ PCFG
- az elemzett mondat minden reprezentációja kap egy valószínűséget $\rightarrow$ legvalószínűbb kiválasztása

Mi az a PCFG?

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta \ [p]$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$,
	and p is a number between 0 and 1 expressing $P(\beta \mid A)$
S	a designated **start symbol**

Mi az a PCFG?

A PCFG abban különbözik CFG-től, hogy minden egyes R szabályhoz valószínűséget rendel: $A \rightarrow \beta \ |p|$

$$= P(A \rightarrow \beta)$$

vagy:

$$P(A \rightarrow \beta|A)$$

vagy pedig:

$$P(RHS|LHS)$$

Ha veszünk egy nem terminális elemet és annak összes lehetséges derivációjá̋t: a valószínűségeik összege 1 lesz (köv. dián megszemlélhető):

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

Mini angol grammatika valószínűségekkel

Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that\ [.10] \mid a\ [.30] \mid the\ [.60]$
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book\ [.10] \mid flight\ [.30]$
$S \rightarrow VP$	[.05]	$\mid meal\ [.05] \mid money\ [.05]$
$NP \rightarrow Pronoun$	[.35]	$\mid flight\ [.40] \mid dinner\ [.10]$
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book\ [.30] \mid include\ [.30]$
$NP \rightarrow Det\ Nominal$	[.20]	$\mid prefer\ [.40]$
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I\ [.40] \mid she\ [.05]$
$Nominal \rightarrow Noun$	[.75]	$\mid me\ [.15] \mid you\ [.40]$
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston\ [.60]$
$Nominal \rightarrow Nominal\ PP$	[.05]	$\mid NWA\ [.40]$
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does\ [.60] \mid can\ [.40]$
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from\ [.30] \mid to\ [.30]$
$VP \rightarrow Verb\ NP\ PP$	[.10]	$\mid on\ [.20] \mid near\ [.15]$
$VP \rightarrow Verb\ PP$	[.15]	$\mid through\ [.05]$
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	

Figure 14.1

Egyértelműsítés PCFG-vel

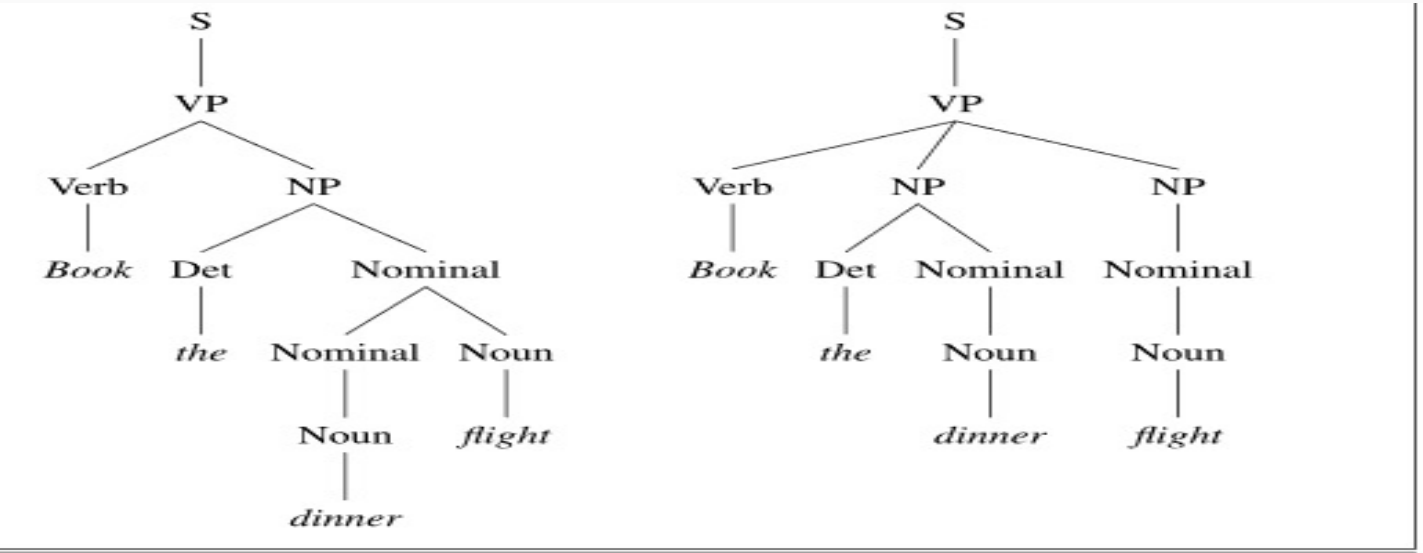
A PCFG egy mondat S minden egyes szintaxisfájához T rendel egy valószínűséget

Egy fa valószínűsége: a fában használt szabályok valószínűségének szorzata:

$$P(T) = \prod_{i=1}^{n} P(RHS_i|LHS_i)$$

Egyértelműsítés PCFG-vel: példa

Book the dinner flight



	Rules		P		Rules		P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40

Figure 14.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds

Egyértelműsítés PCFG-vel: példa

A bal és jobb oldali fa valószínűsége (*Book the dinner flight*):

$P(T_{left}) = .05 \times .20 \times .20 \times .20 \times .75 \times .30 \times .60 \times .10 \times .40 = 2.2 \times 10^{-6}$

$P(T_{right}) = .05 \times .10 \times .20 \times .15 \times .75 \times .75 \times .30 \times .60 \times .10 \times .40 = 6.1 \times 10^{-7}$

A legvalószínűbb derivációk kiválasztása:

$$\hat{T}(S) = \underset{T s.t. S=\text{yield}(T)}{\text{argmax}} P(T)$$

PCFG nyelvmodellezéshez

A PCFG fontos tulajdonságai nyelvmodellezéshez:

- egy mondat valószínűsége: a mondatból generált szintaxisfák valószínűségének összege:

$$\sum_{T\ s.t. S=yield(T)} P(T)$$

- hozzá tud rendelni valószínűségeket egy mondat részeihez is
- pl. a következő szó valószínűsége a mondat korábbi szavaiból (PCFG nyelvtanok ezt megengedik):

$$P(w_i|w_1, w_2, ..., w_{i-1}) = \frac{P(w_1, w_2, ..., w_{i-1}, w_i)}{P(w_1, w_2, ..., w_{i-1})}$$

A PCFG-k elemzése valószínűségi CKY algoritmussal

valószínűségi CKY algoritmus

A valószínűségi CKY algoritmus feltételezi, hogy a PCGF Chomsky normál alakban van:

1. minden szabályra igaz, hogy a jobb oldalán vagy két nem terminális van vagy egy terminális
2. $A \to BC$ vagy $A \to w$

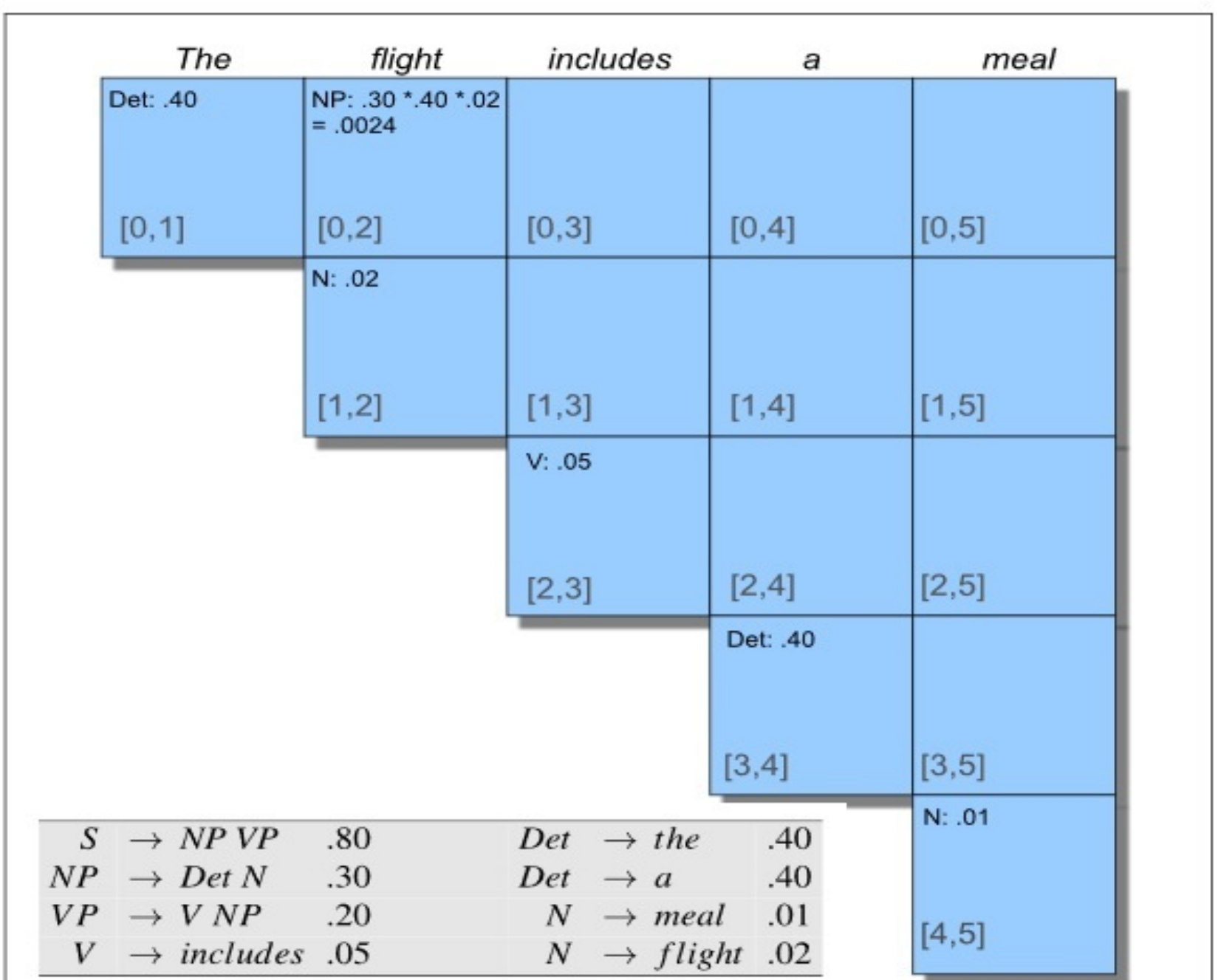
- a valószínűségi CKY algoritmus kétdimenziós tábla helyett háromdimenziós táblával dolgozik: $(n+1) \times (n+1) \times V$
- minden cella $[i, j, A]$ az $(n+1) \times (n+1) \times V$ mátrixban egy A típusú konstituens valószínűségi értéke

valószínűségi CKY algoritmus

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                                and its probability
  for j ← from 1 to LENGTH(words) do
    for all { A | A → words[j] ∈ grammar}
      table[j − 1, j, A] ← P(A → words[j])
    for i ← from j − 2 downto 0 do
      for k ← i + 1 to j − 1 do
          for all { A | A → BC ∈ grammar,
                        and table[i,k,B] > 0 and table[k, j,C] > 0 }
            if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
              table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
              back[i,j,A] ← {k,B,C}
    return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Figure 14.3 The probabilistic CKY algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky normal form. *back* is an array of backpointers used to recover the best parse. The *build_tree* function is left as an exercise to the reader.

Mini példa mini grammatikával



S	$\rightarrow NP\ VP$	.80	Det	$\rightarrow the$	.40
NP	$\rightarrow Det\ N$	.30	Det	$\rightarrow a$	.40
VP	$\rightarrow V\ NP$	.20	N	$\rightarrow meal$	.01
V	$\rightarrow includes$	.05	N	$\rightarrow flight$	.02

Mini példa mini grammatikával (kitöltött verzió)

The	flight	includes	a	meal
Det: .40 [0,1]	NP: .30 *.40 *.02 = .0024 [0,2]	X [0,3]	X [0,4]	**S** .80*.0024 *.000012 = .00000002 [0,5]
	N: .02 [1,2]	X [1,3]	X [1,4]	X [1,5]
		V: .05 [2,3]	X [2,4]	**VP** .20*.05*.0012 = .000012 [2,5]
			Det: .40 [3,4]	**NP** .30*.40*.01 = .0012 [3,5]
				N: .01 [4,5]

S	$\rightarrow$ $NP\ VP$	.80	Det	$\rightarrow$ *the*	.40
NP	$\rightarrow$ $Det\ N$	.30	Det	$\rightarrow$ *a*	.40
VP	$\rightarrow$ $V\ NP$	.20	N	$\rightarrow$ *meal*	.01
V	$\rightarrow$ *includes*	.05	N	$\rightarrow$ *flight*	.02

PCFG szabály-valószínűségek felderítése

PCFG szabály-valószínűségek felderítése

De hogyan jussunk hozzá a PCFG szabályainak valószínűségéhez? Két mód áll rendelkezésre:

1. eset: meglévő Treebank-ból való kinyerés

$$P(\alpha \to \beta|\alpha) = \frac{Count(\alpha \to \beta)}{\sum_\gamma Count(\alpha \to \gamma)} = \frac{Count(\alpha \to \beta)}{Count(\alpha)}$$

2. eset: inside-outside algoritmus
   - induljon monden szabály egyenlő valószínűséggel → mondat elemzése → a megszámlált szabályok súlyozása a kezdeti valószínűségekkel → valószínűségek újrakalkulálása → egész folyamat ismétlése

PCFG: problémák

PCFG: problémák

1. **Poor independence assumpions**: a CFG környezetfüggetlensége miatt kevésbé jól modellezi a struktúrális függőségeket
2. **Lack of lexical conditioning**: a CFG szabályai nem veszik figyelembe az egyes szavak szintaktikai jellemzőit
   –> grammatikai kategóriák, prepozíciók, mellérendelések kétértelműsége

Poor independence assumpions

- a PCFG egyes szabályainak valószínűsége, mint például a $NP \rightarrow Det\ N$, függetlenek a szintaxisfa többi részétől
- ennek nem kellene így lennie:

	Pronoun	Non-Pronoun
Subject	91%	9%
Object	34%	66%

- pl. az angolban azok az *NP*-k, amik szintaktikailag alanyi pozíciót foglalnak el nagyobb eséllyel névmások
- míg a szintaktikailag tárgyi pocícióban: inkább nem névmások (hanem pl. *NNP* vagy *DT NN*)

Lack of lexical conditioning

A PCFG másik hiányossága, hogy érzéketlen a fában lévő szavakra, pedig:

1. egy fa valószínűségébe beleszámít az egyes szavak grammatikai kategóriájának valószínűsége is (pl. $V \rightarrow$ *sleep*, $NN \rightarrow$ *book*, stb)
2. a szavak figyelembevétele hozzásegít annak eldöntéséhez, hogy egy PP-t NP-hez vagy VP-hez kapcsoljunk
3. a mellérendelés egyérteműsítésében is segít (pl. *dogs in houses and cats*)

Példák PP attachment-re

1. példa: Workers dumped sacks into a bin $\rightarrow$ VP-hez kell csatolni az *into*-t és nem az NP-hez

2. példa: fishermen caught tons of herring $\rightarrow$ NP-hez kell csatolni a *of* -ot

Az angolban statisztikailag több esetben kapcsolódik PP NP-hez, de ez nem jó kiindulási pont

A fenti példák jól mutatják, hogy nem érdemes kiválasztani egy preferálandó frázist, amihez kötjük majd a PP-ket

PP attachment: *workers dumped sack into a bin*

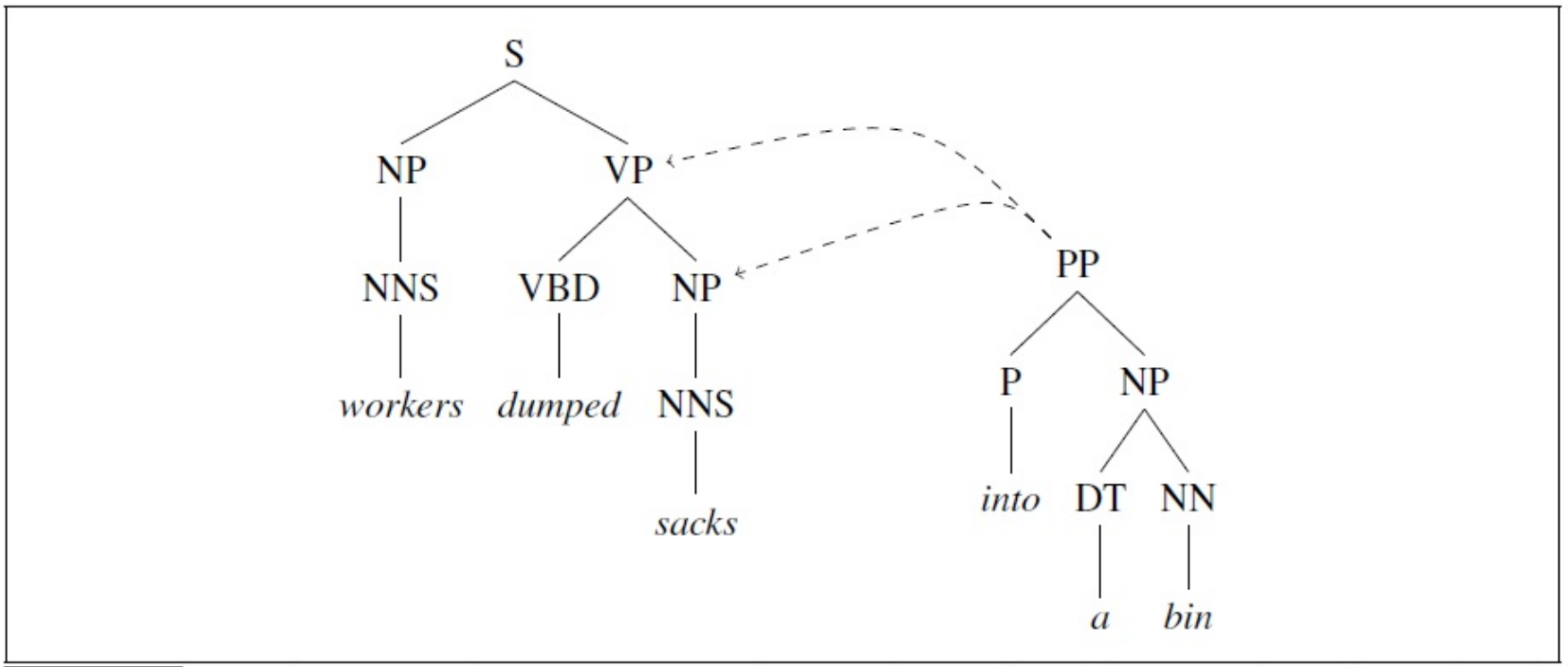


Figure 14.6 Another view of the preposition attachment problem. Should the *PP* on the right attach to the *VP* or *NP* nodes of the partial parse tree on the left?

Példák PP attachment-re

Mi határozza meg, hogy míg az 1. példában VP attachment-re van szükségünk, addig a 2. példában az NP attachment-re?

- az 1. példában a *dumped* és az *into* között nagyobb az affinitás, mint a *sack* és az *into* között $\rightarrow$ VP attachment
- viszont a 2. példában a *tons* és *of* között nagyobb az affinitás $\rightarrow$ NP attachment

PCFG nyevtanok javítása

Nem terminálisok tovább bontása (split)

Hogyan lehetne a struktúrális függőségeket (**poor independence assumpions**) jobban modellezni?

- split: $\text{Non\_Term}_{subcat\_one}$ és $\text{Non\_Term}_{subcat\_two}$
- ha tudjuk, hogy angolban az *NP*-k alanyi pozícióban inkább névmások, míg a szintaktikailag tárgyi pocícióban inkább nem névmások: $\rightarrow NP_{subject}$ és NP_{object}
- így az $NP_{subject} \rightarrow PRP$ és $NP_{object} \rightarrow PRP$ külön-külön más valószínűséget fog kapni

Nem terminálisok tovább bontása: implementáció

Egy lehetséges mód a megvalósítására: **parent annotation**

- minden csomópontot felcímkézünk a szülője nevével

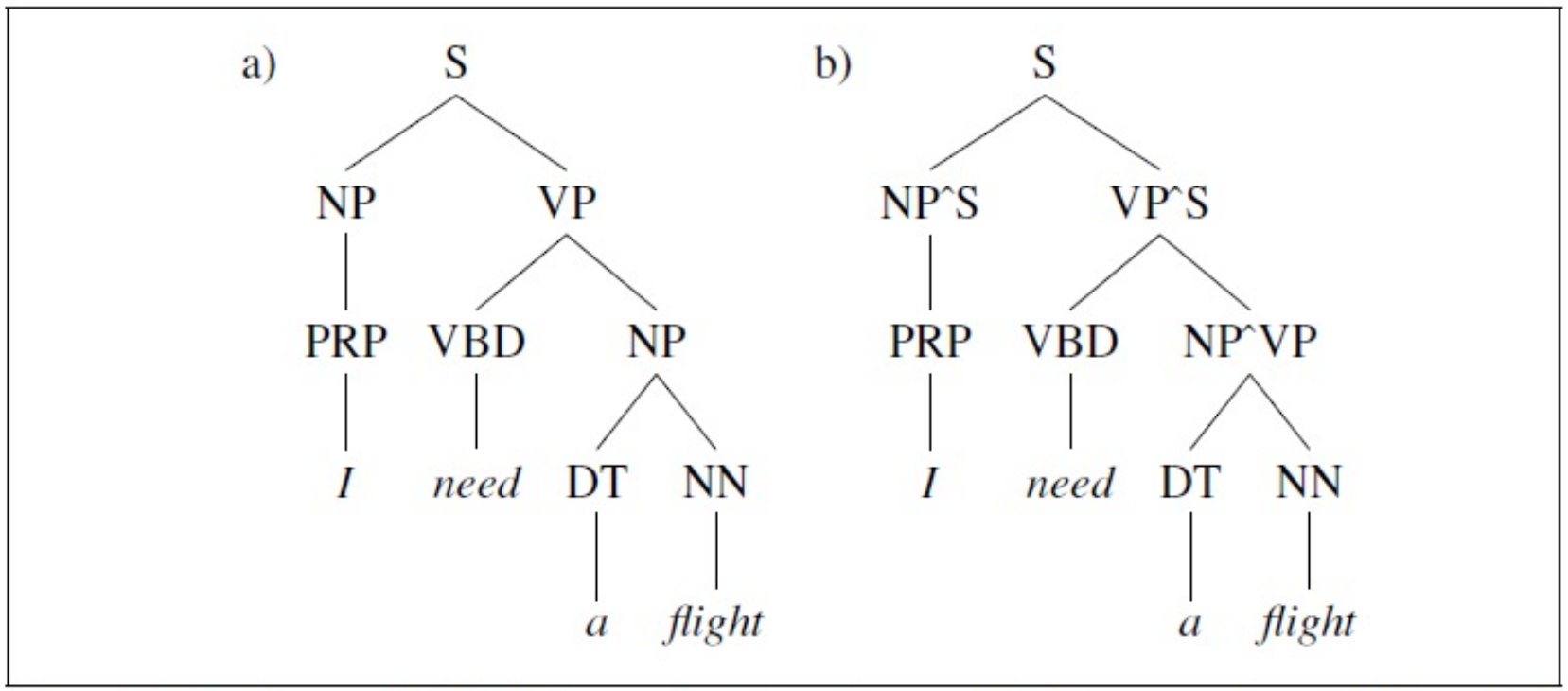


Figure 14.8 A standard PCFG parse tree (a) and one which has **parent annotation** on the nodes which aren't pre-terminal (b). All the non-terminal nodes (except the pre-terminal part-of-speech nodes) in parse (b) have been annotated with the identity of their parent.

Parent annotaion továbbjavítása

A parent annotation koncepcióját is tovább lehet fejleszteni

- nem csak a nem terminálisokat, hanem preterminálisokat is tovább bontjuk (szülővel való felcímkézés)

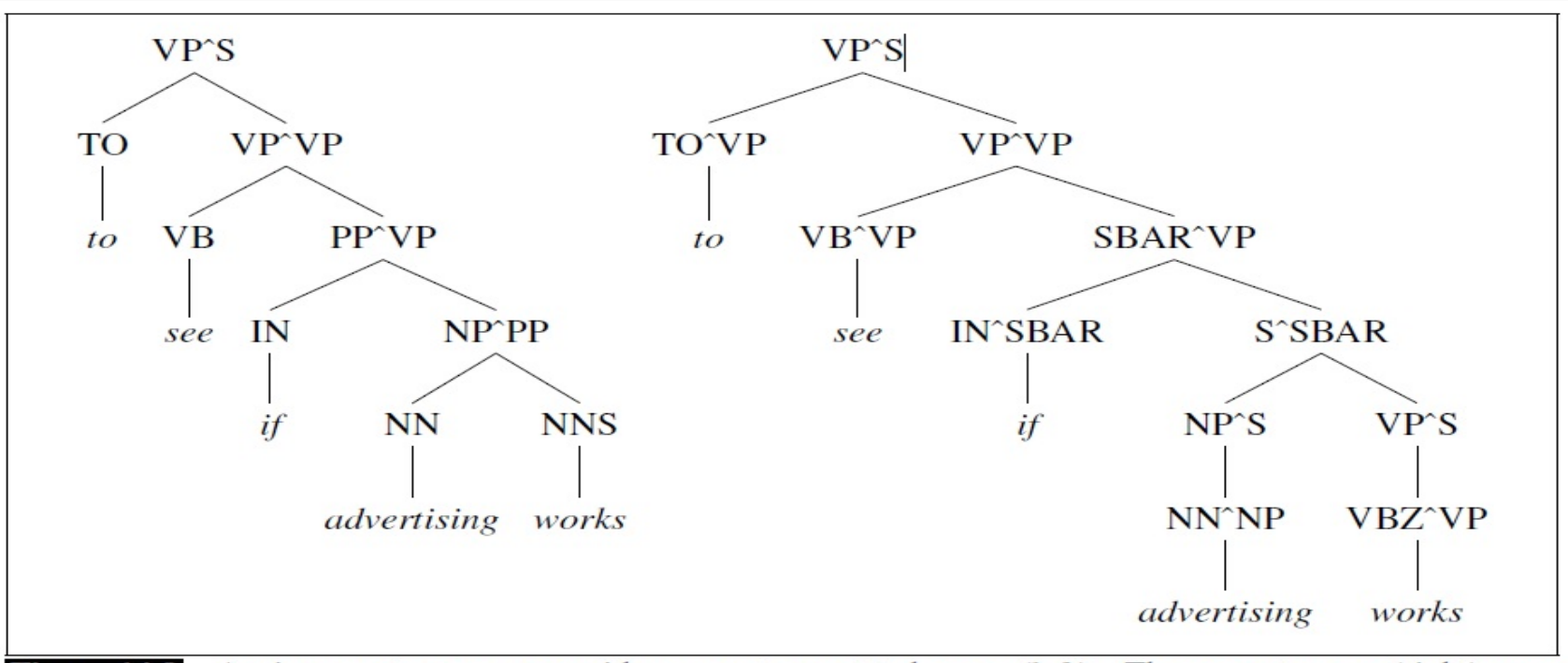


Figure 14.9 An incorrect parse even with a parent-annotated parse (left). The correct parse (right), was produced by a grammar in which the pre-terminal nodes have been split, allowing the probabilistic grammar to capture the fact that *if* prefers sentential complements. Adapted from Klein and Manning (2003b).

Hátrányok

Azért ennek is vannak hátrányai:

- a csomópontok felcímkézése növeli a nyelvtan méretét $\rightarrow$ kevesebb tanítóanyag a nyeltan szabályaihoz
- így vigyázni kell a bontás (split) mértékére, illetve jó algoritmust találni rá (pl. split and merge algoritmus)

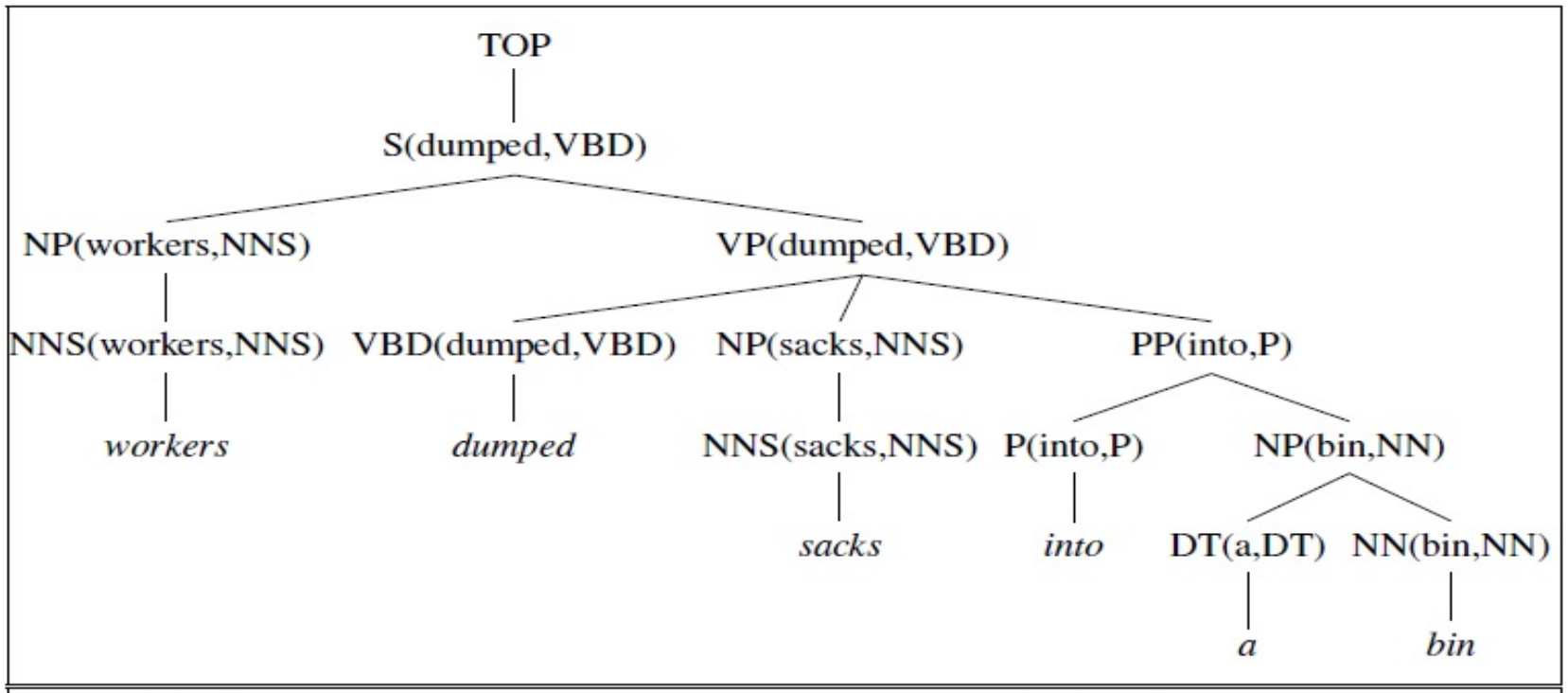
Valószínűségi lexikalizált CFG-k

Valószínűségi lexikalizált CFG-k

- ezekben a modellekben lehetségessé válnak a lexikalizált szabályok (pl. **Collins parser** vagy **Charniak parser**)
- a konstituensek kiegészülnek egy lexikális fejjel
- minden nem terminális mellett fel van tüntetve a feje és a fej part-of-speech tag-je
- a $VP \rightarrow VBD\ NP\ PP$ szabály így lesz kiegészítve:

$$VP(dumped, VBD) \rightarrow VBD(dumped, VBD)\ NP(sacks, NNS)\ PP(into, P)$$

Valószínűségi lexikalizált CFG-k



Internal Rules

TOP	→	S(dumped,VBD)	
S(dumped,VBD)	→	NP(workers,NNS)	VP(dumped,VBD)
NP(workers,NNS)	→	NNS(workers,NNS)	
VP(dumped,VBD)	→	VBD(dumped, VBD)	NP(sacks,NNS) PP(into,P)
PP(into,P)	→	P(into,P)	NP(bin,NN)
NP(bin,NN)	→	DT(a,DT)	NN(bin,NN)

Lexical Rules

NNS(workers,NNS)	→	workers
VBD(dumped,VBD)	→	dumped
NNS(sacks,NNS)	→	sacks
P(into,P)	→	into
DT(a,DT)	→	a
NN(bin,NN)	→	bin

Figure 14.10 A lexicalized tree, including head tags, for a WSJ sentence, adapted from Collins (1999). Below we show the PCFG rules needed for this parse tree, internal rules on the left, and lexical rules on the right.

Valószínűségi lexikalizált CFG-k

- minden PCFG szabály bal oldala tartalmazza az egyik jobb oldali konstituens headword-jét és head tag-jét
- csak az internális szabályok probabilitását kell kiszámolni
- mivel túl specifikus szabályokat fog tartalmazni a nyelvtan, az alábbi MLE-t nem érdemes alkalmazni

$$\frac{Count(VP(dumped, VBD) \rightarrow VBD(dumped, VBD)\ NP(sacks, NNS)\ PP(into, P))}{Count(VP(dumped, VBD))}$$

Collins Parser

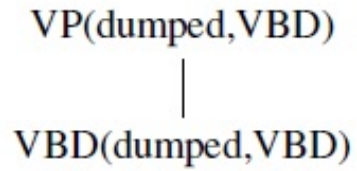
Minden lépésnek saját valószínűségi értéke van, amit a végén összeszorzunk:

1. a szabály fejének legenerálása
2. a fej dependenseinek legenerálása egymás után balról jobbra
3. ha bal vagy jobb oldalon nincs dependens, a STOP speciális szimbólumot arra az oldalra

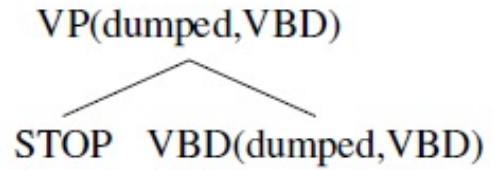
$P_H \rightarrow$ a fej generálása
$P_L \rightarrow$ baloldali dependensek generálása
$P_R \rightarrow$ jobboldali dependensek generálása

Collins Parser

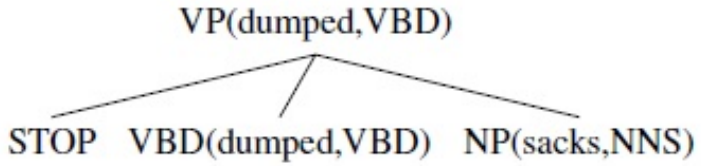
1. Generate the head VBD(dumped,VBD) with probability
P(H|LHS) = P(VBD(dumped,VBD) | VP(dumped,VBD))



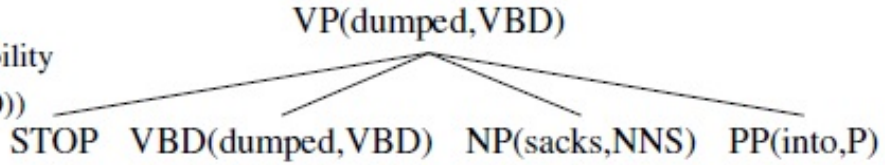
2. Generate the left dependent (which is STOP, since there isn't one) with probability
P(STOP| VP(dumped,VBD) VBD(dumped,VBD))



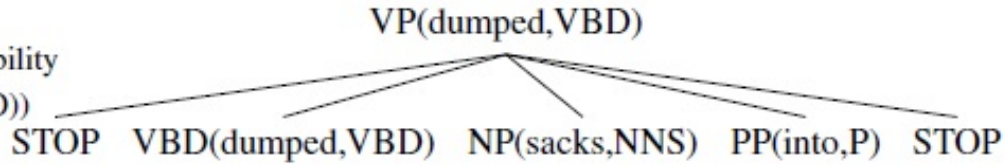
3. Generate right dependent NP(sacks,NNS) with probability
P_r(NP(sacks,NNS| VP(dumped,VBD), VBD(dumped,VBD))



4. Generate the right dependent PP(into,P) with probability
P_r(PP(into,P) | VP(dumped,VBD), VBD(dumped,VBD))



5) Generate the right dependent STOP with probability
P_r(STOP | VP(dumped,VBD), VBD(dumped,VBD))



Collins Parser

Tehát a

$$P(VP(dumped, VBD) \rightarrow VBD(dumped, VBD)\ NP(sacks, NNS)\ PP(into, P))$$

kiszámítása a következőképpen történik:

$$\begin{aligned} & P_H(VBD|VP, dumped) \times P_L(STOP|VP, VBD, dumped) \\ & \quad \times P_R(NP(sacks, NNS)|VP, VBD, dumped) \\ & \quad \times P_R(PP(into, P)|VP, VBD, dumped) \\ & \quad \times P_R(STOP|VP, VBD, dumped) \end{aligned}$$

MLE egy kisebb egységre

$$\frac{Count(VP(dumped, VBD)\ with\ NNS(sacks)\ as\ a\ daughter\ \ somewhere\ on\ the\ right)}{Count(VP(dumped, VBD))}$$

Collins parser

A lépések képletekkel

1. *fej*

$$P_H(H(hw, ht)|P, hw, ht)$$

2. *bal konstituensek*

$$\prod_{i=1}^{n+1} P_L(L_i(lw_i, lt_i)|P, H, hw, ht)$$

ahol

$$L_n + 1(lw_{n+1}, lt_{n+1}) = STOP$$

3. *jobb konstituensek*

$$\prod_{i=1}^{n+1} P_R(R_i(rw_i, rt_i)|P, H, hw, ht)$$

ahol

$$R_n + 1(rw_{n+1}, rt_{n+1}) = STOP$$

Valószínűségi CCG elemzés

Ismétlés

A CCG három részből áll:

- kategóriák halmaza
- lexikon, ami szavakhoz kategóriákat rendel
- a szabályok halmaza, ami meghatározza, hogyan lehet a kategóriákat kombinálni

Kategóriák lehetnek:

- atomikus elemek: pl. S vagy NP
- függvények: $(S\backslash NP)/NP$, ami keres egy NP-t jobb oldalon és visszaad egy $(S\backslash NP)$-t

Többértelműség a CCG-ben

CFG: a többértelműség a szabályokból számrazik
CCG: elsősorban a lexikonban kiosztott kategóriákból

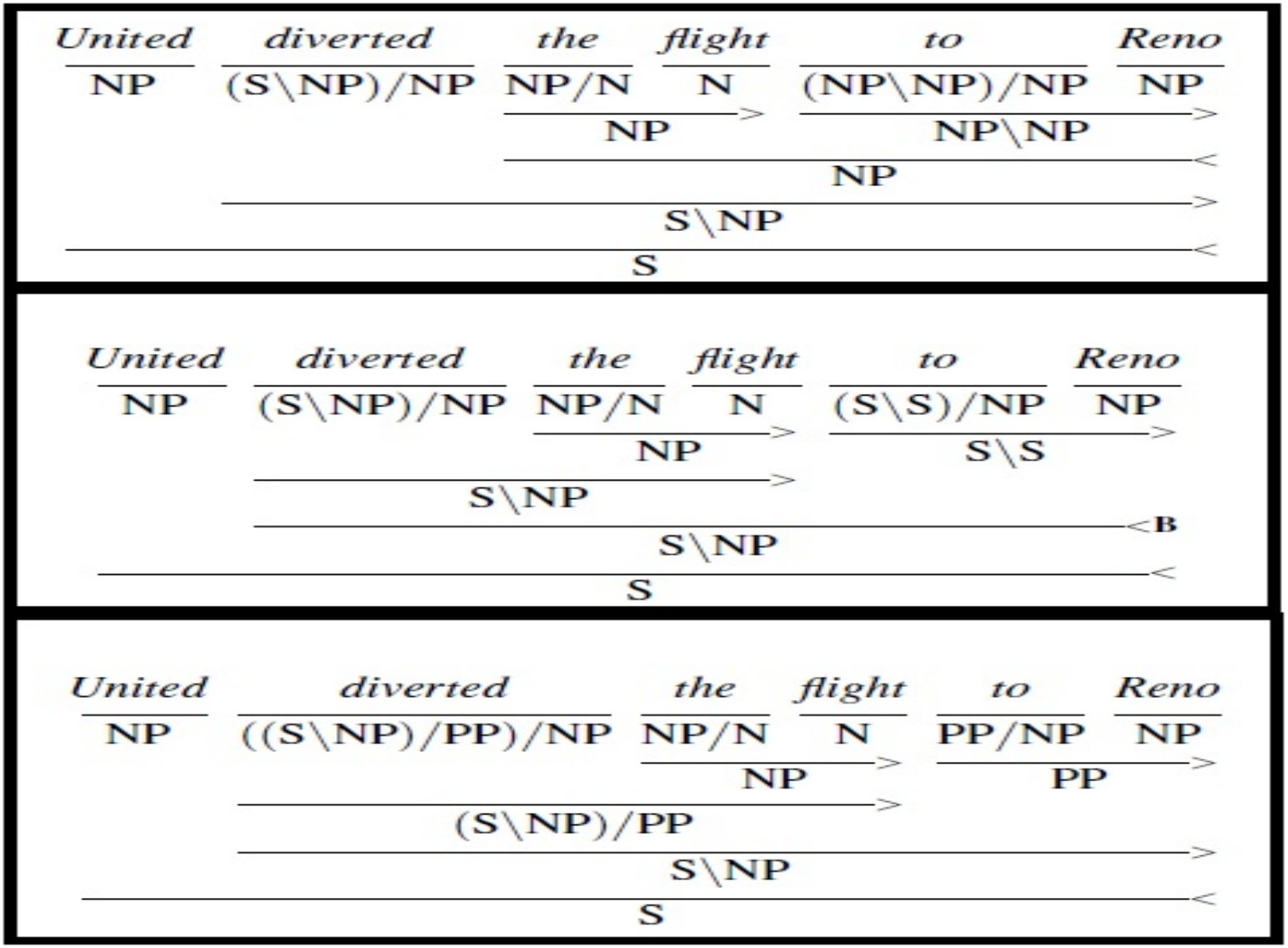
Vegyük a következő példamondatot:

United diverted the flight to Reno.

A P *to*-hoz három kategóriát rendelhetünk:

- módosíthatja a *the flight*-ot $\rightarrow$ (NP\NP)/NP
- módosíthatja a VP-t $\rightarrow$ (S\S)/NP
- illetve lehet a V *dIverted* argumentuma is $\rightarrow$ PP/NP

Többértelműség a CCG-ben



CCG Parsing Frameworks

PCKY: rögzíti az összes inputból kinyert konstituens helyét, kategóriáját, valószínűségét.

CCG-ben lévő sok lexikai kategória, amit egy szó felvehet + kombinatorikus szabályok $\rightarrow$ túl sok lehetőség, zombi konstituensek

A fenti probléma szupertaggeléssel (supertagging) orvosolható:

- minden szónak megtalálni a legvalószínűbb kategóriáját és azokat használni

Supertagging

- a supertaggelés hasonló a part-of-speech taggeléshez
- a lexikalizált nyelvtanok esetében használatosak
- CCG supertaggerek olyan treebankokra támaszkodnak, mint a CCGbank
- a CCGbank több, mint 1000 lexikális kategóriát tartalmaz
- összehasonlításképpen: a Penn Treebank tagset-ben 45 POS típus van.

Maximum entropy Markov model (MEMM)

A legjobb tag szekvencia $\hat{T}$ kiválasztása

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

Supertagging

A legjobb tag szekvencia T̂ ált. túl sok nem megfelelően kiválasztott tag-et tartalmazna

- az összes lehetséges szó-tag pár valószínűségére lenne szükség
- bármely tag valószínűsége: azon supertag-szekvenciák valószínűségének összege, amik tartalmazzák a vizsgált tag-et a tag pocíciójában
- ezeket az értékeket hatékonyan ki lehet számolni a HMM-ekhez használt forward-backward algoritmussal

CCG parszolás A* algoritmussal

Az A* algoritmus célja, hogy minimális lépésben találja meg a legjobb derivációt heurisztikus kereséssel

- minden lépés során a legköltséghatékonyabb részelemzést adja át az agendának és azt bővíti tovább
- *f-költség*függvényt használ
- *f-költségnek* két része van:
  - $g(n)$, az n állapot költsége
  - $h(n)$, költségbecslése annak ,hogy n állapot felhasználva eljussunk a befejezett elemzésig
- a legkisebb értékű $g(n) + h(n)$ úton halad tovább az algoritmus

CCG parszolás A* algoritmussal

- állapot $\rightarrow$ *élek(edges)* végigjárt konstituensekkel
- él $\rightarrow$ konstituens eleje és vége, grammatikai kategóriája, *f-költsége*

Heurisztikus függvények

Hogyan állapítsjuk meg egy CCG deriváció minőségét?

- vegyük egy deriváció szavaihoz rendelt supertag-ek valószínűségének szorzatát
- ha van egy mondat S és egy deriváció D, ami tartalmazza a supertag-sekvenciát T (negatív logaritmussal érjük el, hogy az alacsonyabb költség legyen a jobb):

$$P(D, S) = P(T, S)$$

$$= \prod_{i=1}^{n} -\log P(t_i|s_i)$$

Heurisztikus függvények

Hogyan állípítsuk meg egy él *f-költségét*?

- az *f-költcség* $g(n) + h(n)$
- a $h(n)$-nek soha nem szabad túlbecsülnie a végső elemzés költségét
- így feltételezi, hogy a következő szavak a legvalószínűbb supertag-gel fognak fendelkezni

Heurisztikus függvények

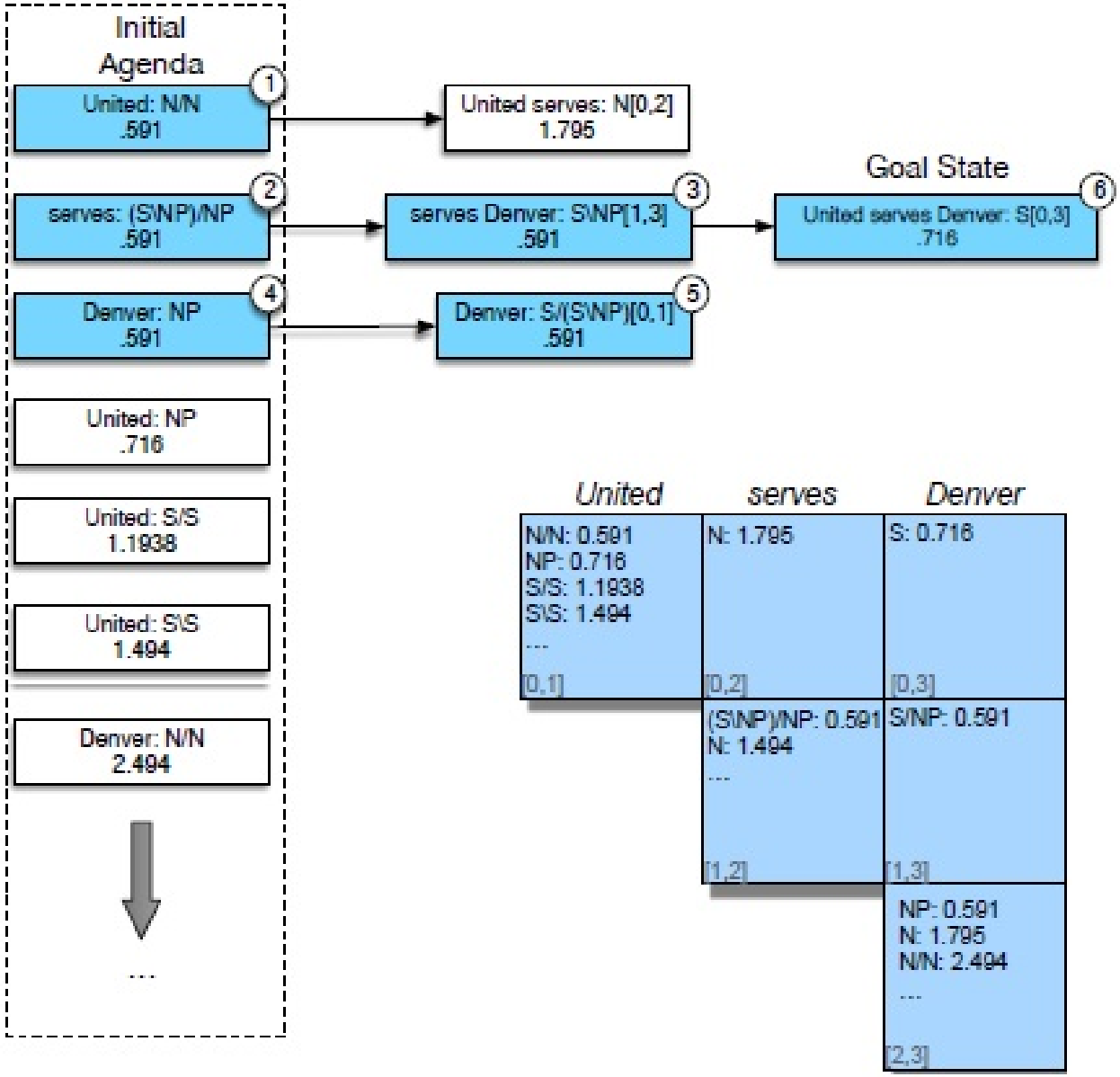
Egy él *f-költségének* kiszámolása

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

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

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

Heurisztikus függvények



Elemzések kiértékelése

PARSEVAL mérés

A hipotetikus fában lévő konstituensek mennyi hasonlóságot mutatnak a kézi-annotált gold standard referenciafában lévőkhöz?

- egy hipotetikus konstituens C_h helyes, ha a referencia fában van egy vele azon konstituens C_r megegyező kezdő- és végponttal, szimbólummal
- ezután meg tudjuk mérni a fedést/pontosságot

Fedés és pontosság meghatározása

- Fedés/pontosság:

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

$$\textit{labeled precision} := \frac{\#\textit{of correct constituents in hypothesisparse of s}}{\#\textit{of total constituents in hypothesis parse of s}}$$

- F-mérték $\rightarrow$ mennyire van ballanszban a kettő:

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

Frame Title

- a $\beta > 1 \rightarrow$ magasabb fedés
- a $\beta < 1 \rightarrow$ magasabb pontosság
- $\beta = 1 \rightarrow$ kiegyensúlyozott fedés és pontosság (F_1)

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