Statistical Parsing
Chapter 14

Lecture #9

October 2012

Parse Disambiguation

- In the previous chapter we have seen several instances of parsing ambiguity: coordination ambiguity and attachment ambiguity
- So far – we return every parse and let later modules deal with the ambiguity
- Can we use probabilistic methods to choose most likely interpretation?

Probabilistic Parsing

- Probabilistic Context Free Grammar: a probabilistic grammar which favors more common rules
- Augment each rule with its associated probability
- Modify parser so that it returns most likely parse (CKY Algorithm)
- Problems and augmentations to the basic model

Probability Model

- Attach probabilities to grammar rules – representing the probability that a given non-terminal on the rule's LHS will be expanded to the sequence on the rule's LHS.
- The expansions for a given non-terminal sum to 1
  - VP -> verb .55
  - VP -> verb NP .40
  - VP -> verb NP NP .05
- Probability captures P(RHS | LHS)

Figure 14.1

<table>
<tr><th>Grammar</th><th></th><th>Lexicon</th></tr>
<tr><td>S → NP VP</td><td>[.80]</td><td>Det → that [.10] | a [.30] | the [.60]</td></tr>
<tr><td>S → Aux NP VP</td><td>[.15]</td><td>Noun → book [.10] | flight [.30]</td></tr>
<tr><td>S → VP</td><td>[.05]</td><td>| meal [.15] | money [.05]</td></tr>
<tr><td>NP → Pronoun</td><td>[.35]</td><td>| flights [.40] | dinner [.10]</td></tr>
<tr><td>NP → Proper-Noun</td><td>[.30]</td><td>Verb → book [.30] | include [.30]</td></tr>
<tr><td>NP → Det Nominal</td><td>[.20]</td><td>| prefer; [.40]</td></tr>
<tr><td>NP → Nominal</td><td>[.15]</td><td>Pronoun → I [.40] | she [.05]</td></tr>
<tr><td>Nominal → Noun</td><td>[.75]</td><td>| me [.15] | you [.40]</td></tr>
<tr><td>Nominal → Nominal Noun</td><td>[.20]</td><td>Proper-Noun → Houston [.60]</td></tr>
<tr><td>Nominal → Nominal PP</td><td>[.05]</td><td>| NWA [.40]</td></tr>
<tr><td>VP → Verb</td><td>[.35]</td><td>Aux → does [.60] | can [40]</td></tr>
<tr><td>VP → Verb NP</td><td>[.20]</td><td>Preposition → from [.30] | to [.30]</td></tr>
<tr><td>VP → Verb NP PP</td><td>[.10]</td><td>| on [.20] | near [.15]</td></tr>
<tr><td>VP → Verb PP</td><td>[.15]</td><td>| through [.05]</td></tr>
<tr><td>VP → Verb NP NP</td><td>[.05]</td><td></td></tr>
<tr><td>VP → VP PP</td><td>[.15]</td><td></td></tr>
<tr><td>PP → Preposition NP</td><td>[1.0]</td><td></td></tr>
</table>

PEARSON *Speech and Language Processing*, Second Edition
Daniel Jurafsky and James H. Martin

Using a PCFS

- A PCFS can be used to estimate a number of useful probabilities concerning a sentence and its parse trees
  - Probability of a particular parse tree (useful for disambiguation)
  - Probability of a sentence or piece of a sentence (useful for language modeling)

How?

- A derivation (tree) consists of the set of grammar rules that are in the tree
- The probability of a tree is just the product of the probabilities of the rules in the derivation.

Figure 14.2

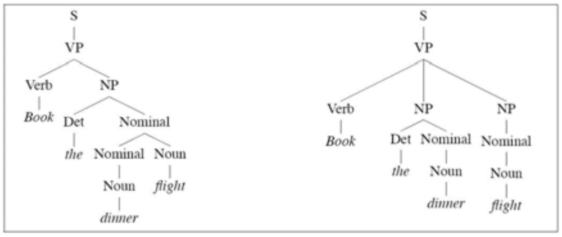


Figure 14.2 continued

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	→	flights	.40	Noun	→	flights	.40

Probability Model

$$P(T,S) = P(T)P(S\,|\,T) = P(T); \text{since } P(S\,|\,T) = 1$$

$$P(T,S) = P(T) = \prod_{n \in T} p(r_n)$$

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

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

Probability Model

- The probability of a word sequence P(S) is the probability of its tree in the unambiguous case (i.e., where there is exactly one tree).
- In the case where there is ambiguity (multiple trees) the probability of the sequence is the sum of the probabilities of the trees.

Parsing to get most likely parse

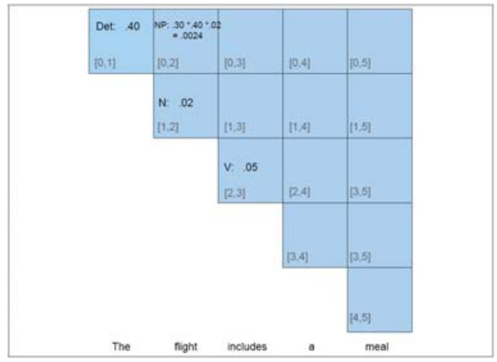
- Can do with a simple extension of our parsing algorithms – book does CKY (and indicates that is most used version).
- Essentially – give each constituent that is in the table a probability (they refer to this as another dimension in the table), when a new constituent , C, is found to be added to the table at cell [I, J], only add it if that cell either does not contain a constituent C or if the probability of this new constituent is less than the probability of the existing one (in which case, you overwrite the old one).
- Assuming rule is C -> c1 c2
- New prob = prob(rule) x prob(c1) x prob(c2)

Figure 14.3

```
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]
```

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

Figure 14.4



Learning PCFG Rule Probabilities

1. Learn from a treebank, a corpus of already parsed sentences
   – So for example, to get the probability of a particular VP rule just count all the times the rule is used and divide b the number of VPs overall
2. Count by parsing with a non-probabilistic parser.
   – Issue is ambiguity – inside-out-algorithm - sort of boot-strap – parse a sentence, compute a probability for each parse, use these probabilities to weight the counts, re-estimate the rule probabilities, and so on, until our probabilities converge.

Problems with PCFGs

- Probability model we are using is just based on the rules in the derivation… and these are context free rules
  – Poor independence assumptions miss structural dependencies between rules since cannot take into account in the derivation a rule is used.
  – Lack of sensitivity to lexical dependencies
    - Do have probability associated with N-> bank
    - But verb subcategorization and prepositional phrase attachment might depend on the particular words being used.
    - Use lexical heads as part of rule – then you run into problems with sparse data – so need to make independence assumptions to reduce amount of data needed.

Structural Dependencies between Rules

NP -> det NN .28
NP -> Pronoun .25

- These probabilities should depend on where the NP is being used:

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

- SolutionSplit non-terminal into 2 (e.g., using parent annotation NP^S vs NP^VP) and learn rule probabilities for split rules.

Figure 14.8

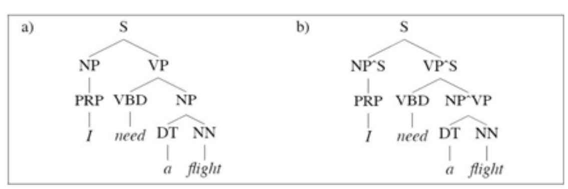


Figure 14.5

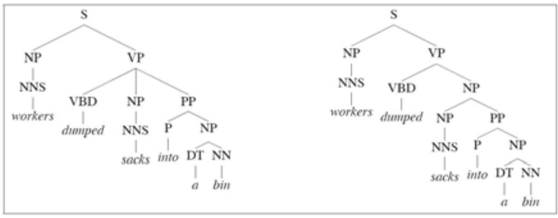
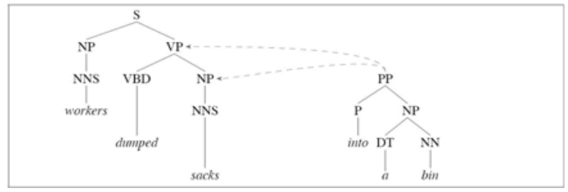


Figure 14.6 – KFM added to slide

Note: if we prefer the rule for VP attachment then it will be incorrect for sentences like "fishermen caught tons of herring"

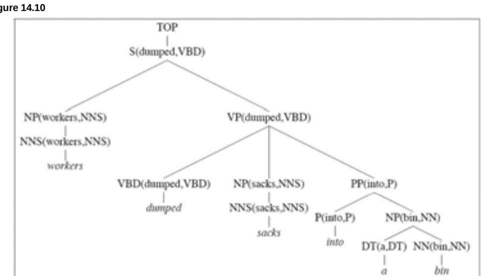


Seems lexical in nature: affinity between the verb "dumped" and the preposition "into" is greater than the affinity between the noun "sacks" and the preposition "into"

Lexical Dependencies

- Must add lexical dependencies to the scheme and condition the rule probabilities on the actual words
- What words?
  - Make use of the notion of the head of a phrase
  - The head is intuitively the most important lexical item in the phrase – and there are some rules for identifying
    - Head of an NP is its noun
    - Head of a VP is its verb
    - Head of a sentence comes from its VP
    - Head of a PP is its preposition
  - Use a lexicalized grammar in which each non-terminal in the tree is annotated with its lexical head

Figure 14.10



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

Issues with Learning

- Not likely to have significant counts in any treebank to actually learn these probabilities.
- Solution: Make as many independence assumptions as you can and learn from these
- Different modern parsers make different independence assumptions – E.g., Collins parser have head and dependents on left are assumed independent of each other and independent of depends on right (which make similar assumptions about the left dependents)

Summary

- Probabilistic Context-Free Grammars
  - Help us deal with ambiguity by preferring more likely parses
- Grammar rules have attached probabilities which capture the probability of the rule's RHS given its LHS (probabilities of all rules with same LHS sum to 1)
- We can compute the probability of a tree (product of the probabilities of the rules used)
- Can parse using augmented algorithms
- Can learn probabilities from a tree bank
- PCFGs have problems with independence assumptions and with lack of lexical conditioning
- Some solutions exist – problems of data sparcity