경희대학교
KYUNG HEE UNIVERSITY

Statistical Language Learning

(Chapter 7 ~ 8)

박형준

2019.01.08

Email : khuphj@gmail.com

NLP-LAB,
Department of Computer Engineering,
Kyung Hee University.

7. Learning Probabilistic Grammars

- 7.1 Why the Simple Approach Fails
- 7.2 Learning Dependency Grammars
- 7.3 Learning from a Bracketed Corpus
- 7.4 Improving a Partial Grammar

7.1 Why the Simple Approach Fails

[Problem]

Currently we do not have a grammar for English (or any other natural Ian-guage) that can handle the wide variety of sentences turning up in large cor-pora. This, of course, makes it impossible to apply our parsing technology to these texts.

We also stand to benefit in that the PCFG training scheme derived in chapter 6 suggest a very simple method for learning PCFGs, namely:

1.have the machine generate all possible PCFG rules
2.assign them some initial probabilities
3.run the training algorithm on a sample text
4.remove those rules with zero probability, leaving the “correct” grammar

The idea here is that when we generated the PCFG rules we generated right ones and wrong ones.

If the training scheme does its work properly, the wrong ones should get zero probability and can thus be removed, leaving the right ones.

In this chapter we look at work inspired by this simple idea.

7.1 Why the Simple Approach Fails

[Problem]

Unfortunately this simple idea does not work, at least not in the version stated. There are two reasons:

- Unless one places constraints on the kinds of context-free rules allowed, there is no bound on the number of possible context-free rules.
A corollary to this is that, even with constraints on the number of rules, often the number is so large as to make training impractical

- As we noted in introducing the training algorithm, this algorithm is guaranteed. only to find a local maximum. That is, even if we can get all the possible rules and even If the corpus is sufficiently representative, the training algorithm can find a bad set of rules just because it found a local maximum and not the global one.

✋ What is local maximum?

7.1 Why the Simple Approach Fails

[Sol.]

• Unless one places constraints on the kinds of context-free rules allowed, there is no bound on the number of possible context-free rules.
A corollary to this is that, even with constraints on the number of rules, often the number is so large as to make training impractical

Solution:

if one does not specify the non-terminal symbols of a context-free grammar, then obviously the number of rules is unbounded since one can keep introducing new non-terminals and making up new rules for them.

Thus any PCFG learner based on the above learning scheme must restrict the non-terminals to some finite set. Even if one specifies the non-terminals, the number of rules is unbounded since one can keep making up rules with longer and longer right-hand sides. So in addition the PCFG learner must impose rule-length restrictions.

7.1 Why the Simple Approach Fails

[Sol.]

• As we noted in introducing the training algorithm, this algorithm is guaranteed. only to find a local maximum. That is, even if we can get all the possible rules and even If the corpus is sufficiently representative, the training algorithm can find a bad set of rules just because it found a local maximum and not the global one.

Solution:

As for the local-maximum issue, it may not be obvious at first how serious a problem this really is.
However, Chaniak and Carroll report an experiment showing it is serious indeed.
The purpose was to find out how many different rule sets the system would come up with when the rules were started with different probabilities-nothing else changed from run to run
The results were unequivocal. The system was run 300 times, each time with a different random assignment of probabilities to the set of all possible rules. 300 different rule sets were found in this way. There were no duplicates

7.1 Why the Simple Approach Fails

[Sol.]

• As we noted in introducing the training algorithm, this algorithm is guaranteed. only to find a local maximum. That is, even if we can get all the possible rules and even If the corpus is sufficiently representative, the training algorithm can find a bad set of rules just because it found a local maximum and not the global one.

Solution:

For example, in “The dog went to the park” the rule ‘vp->verb pp” could have been used, but the rule “vp->pp verb” could not(assuming the normal interpretations of vp and pp)

It is useful to think of our grammar learner as searching for the right probabilities in a space in which each point corresponds to a particular probability setting for each of the possible rules. If there are a lot of local maxima, then we are going to get stuck in one.

Suppose, however, that we have further information on which part of the space has the true global maximum. Obviously this will help a lot. Thus, in addition to the aforementioned restrictions bounding the number of rules, we also require this rule-space information.

7.2 Learning Dependency Grammars

[Concept]

Carroll and Charniak [11,12] built a grammar-induction program in which the constraint on rules came from the program's restriction to dependency grammars.

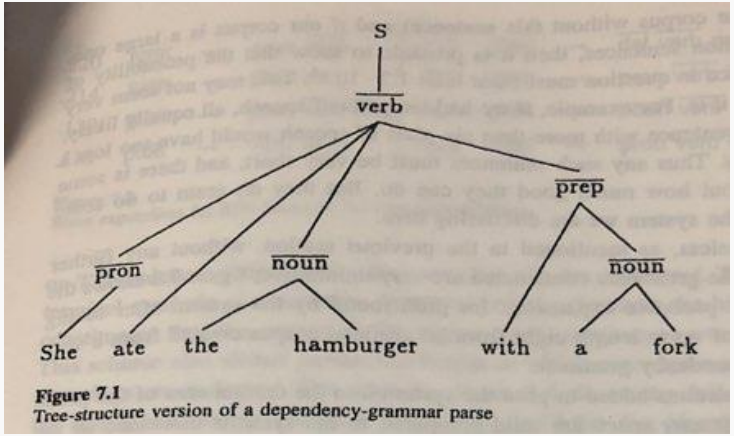
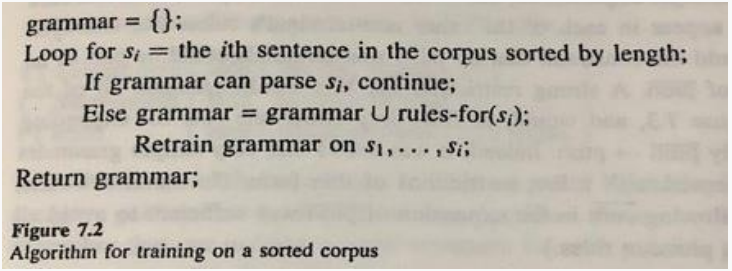


Figure 7.1
Tree-structure version of a dependency-grammar parse

Formally, a dependency grammar is a triple< N,S,R >

"She ate the hamburger with a fork"
Furthermore, for the purposes of the grammar learning experiments the right-hand sides of rules were arbitrarily limited to be of maximum length four. Thus there is only a finite number of possible rule

7.2 Learning Dependency Grammars



```
grammar = {};
Loop for s_i = the ith sentence in the corpus sorted by length;
    If grammar can parse s_i, continue;
    Else grammar = grammar ∪ rules-for(s_i);
        Retrain grammar on s_1, ..., s_i;
Return grammar;
```

Figure 7.2
Algorithm for training on a sorted corpus

[Concept]

Furthermore,
the corpus was sorted by sentence length,
with the idea that it would be easier to learn
a grammar for shorter sentences than longer ones,
and that the short-sentence grammar
could be extended to longer sentences by
adding rules as needed
This algorithm is summarized in figure 7.2.

7.2 Learning Dependency Grammars

[Concept]

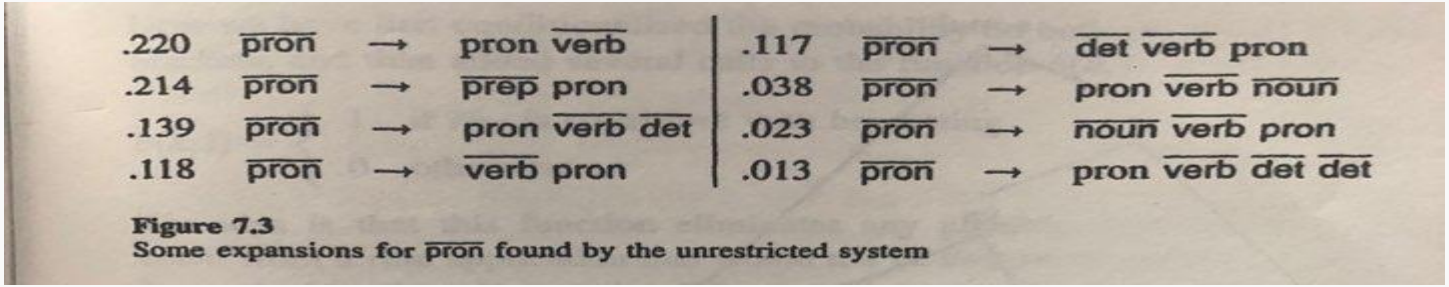
suppose a grammar assigns a relatively high probability to the sentence $w_{1,n}$,
but this sentence does not appear in the corpus.
(This is more plausible if we assume that $w_{1,n}$ are parts of speech and not actual words.)
The probability of this happening in a corpus of m sentences is (1– P($w_{1,n}$))^m

Chamiak and Carroll's system has a threshold on this number such that,
when the probability falls below the threshold, the system "considers" the possibility that this sentence really is not in the grammar, or at least not nearly as likely as the current grammar makes out.

To test this possibility, the system records the current cross entropy and then tries eliminating each of the rules in turn. In each case the grammar is trained with other (possibly new) rules for parsing the corpus sentences. if the cross entropy of any of these trials is below the original "current" cross entropy, then the grammar that produced the lowest of these is adopted as the new grammar.

7.2 Learning Dependency Grammars

[Limit]



.220	$\overline{\text{pron}}$	→	pron $\overline{\text{verb}}$	.117	$\overline{\text{pron}}$	→	$\overline{\text{det}}$ $\overline{\text{verb}}$ pron
.214	$\overline{\text{pron}}$	→	$\overline{\text{prep}}$ pron	.038	$\overline{\text{pron}}$	→	pron $\overline{\text{verb}}$ $\overline{\text{noun}}$
.139	$\overline{\text{pron}}$	→	pron $\overline{\text{verb}}$ $\overline{\text{det}}$	.023	$\overline{\text{pron}}$	→	$\overline{\text{noun}}$ $\overline{\text{verb}}$ pron
.118	$\overline{\text{pron}}$	→	$\overline{\text{verb}}$ pron	.013	$\overline{\text{pron}}$	→	pron $\overline{\text{verb}}$ $\overline{\text{det}}$ $\overline{\text{det}}$

Figure 7.3
Some expansions for $\overline{\text{pron}}$ found by the unrestricted system

Nevertheless, as mentioned in the previous section, without any further guidance the grammars constructed are very unintuitive; figure 7.3 shows the eight most probable expansions for $\overline{pron}$ found by the system after seeing sentences of up to length eight from an artificial corpus created from a very simple dependency grammar.

This system was tried only on comparatively simple corpora generated from simple hand-written dependency grammars (the most complicated grammar had 48 rules).
It was able to learn this and the two simpler ones up to minor variations in the probabilities associated with each rule.

7.3 Learning from a Bracketed Corpus

[Concept]

One interesting way to force the initial probabilities into the correct part of the parameter space was proposed by Pereira and Schabes.

Example:

"Salespeople sold the dog biscuits."
((Salespeople (sold (the dog biscuits))).)

(Salespeople (sold the dog biscuits) .)

((noun (verb (det noun noun))) fpunc)

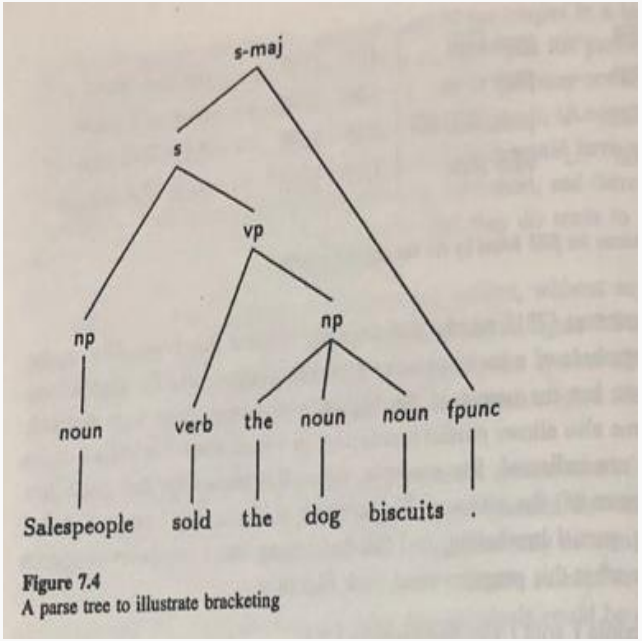


Figure 7.4
A parse tree to illustrate bracketing

7.3 Learning from a Bracketed Corpus

[Concept]

With this in mind, we can return to the equation derived in chapter 6 for estimating the counts for how many (prorated) times a rule $N^j \rightarrow N^p\, N^q$ is used in a certain sentence, equation 6.16, repeated here:

$$C(N^j \rightarrow N^p N^q) = \frac{1}{P(w_{1,n})} \sum_{k,l,m} \alpha_j(k,l) P(N^j \rightarrow N^p N^q) \beta_p(k,m) \beta_q(m+1,l)$$

(Again, this is for the "Chomsky-normal form", which is the form used in [35].) We do not go through the formal derivation to introduce the bracketing information, but instead simply give the new version:

$$C(N^j \rightarrow N^p N^q) = \frac{1}{P(w_{1,n}, \underline{brackets})} \sum_{k,l,m} \underline{b(k,l)} \alpha_j(k,l) P(N^j \rightarrow N^p N^q) \underline{b(k,m)} \beta_p(k,m) \underline{b(m+1,l)} \beta_q(m+1,l) \quad (7.2)$$

$$b(k,l) = \begin{cases} 1 & \text{if } N_{k,l} \text{ is consistent with bracketing} \\ 0 & \text{otherwise} \end{cases}$$

7.3 Learning from a Bracketed Corpus

[Limit]

The resulting parse was judged simply by how well it assigned the full brackets to the test sentences. (If you think about it, the particular non-terminals it assigned are not meaningful since, even if we gave them meaningful names, the fact that the system started out with all possible rules means that there was no way to, say, make vp correspond to what we normally think of as a verb phrase any more than to what we normally think of as a prepositional phrase, or anything else.)
When the input did not contain the brackets, the resulting brackets were correct only 37% of the time. When bracketing information was included this went up to 90%.
Interestingly, while training increased performance for the bracketed corpus from 37% to 90%, it did not improve performance for the unbracketed corpus at all.
This fact seems significant, but its interpretation is not obvious.

7.4 Improving a Partial Grammar

[Concept]

Of the three approaches described here, Briscoe and Waegner's[7] applies the greatest restrictions on the grammar in order to avoid local maxima.
Perhaps because of this, their method creates the most realistic grammars of the three.
Briscoe and Waegner start with the constraint that the grammar be in Chomsky-normal form with a restricted number of non-terminal symbols.

They also explicitly adopt a version of X theory in which two levels of bars are permitted. In order to tie down, say, noun to phrases in which nouns appear, rules have the following restrictions:

1. In the rule $\bar{X} \rightarrow$ y z, either y or z must be either an x or an $\bar{X}$
2. In the rule $\bar{\bar{X}} \rightarrow$ y z, either y or z must be an x, an $\bar{X}$ or an $\bar{\bar{X}}$

Thus, for example, both $\overline{noun}$ and $\overline{\overline{\text{noun}}}$ when expanded by these rules eventually expand into phrases containing at least one noun. Note, however, that double-barred non-terminals are not restricted to expand into single-barred ones that in turn expand into parts of speech and hence into words.
For example, in this scheme pronouns are $\overline{\overline{\text{nouns}}}$, as in "$\overline{\overline{\text{noun}}} \rightarrow$ he."
This makes sense if one thinks of a $\overline{\overline{\text{noun}}}$ as a complete noun phrase.

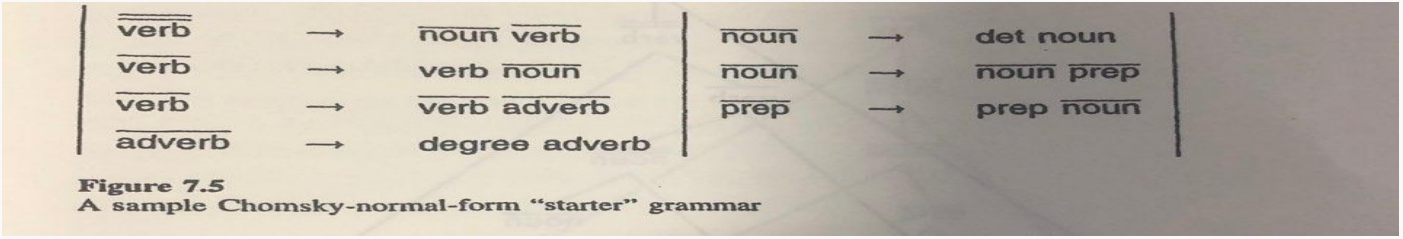
7.4 Improving a Partial Grammar

[Concept]
Briscoe and Waegner place a second restriction on their rules, the import of which is much less clear:
in any rule of the form “x → y z” in which both y and z are parts of speech (i.e., have zero bars), z must be either an adverb(부사) or a noun.
One thing we have not discussed is the fact that in $\bar{X}$ theory most words
are classified as having the features of ±noun and ±verb.
Nouns are +noun and –verb,
whereas adverbs are +noun and +verb.
The reasons for this are beyond the scope of our text. However,
it does serve to make the otherwise odd conjunction of nouns and adverbs in this second restriction slightly more reasonable,
since the rule actually says that z must be +noun.

✋ What is different +- noun?

7.4 Improving a Partial Grammar

[limit]



$\overline{\overline{\text{verb}}}$	→	noun $\overline{\text{verb}}$	$\overline{\text{noun}}$	→	det noun
$\overline{\text{verb}}$	→	verb $\overline{\text{noun}}$	$\overline{\text{noun}}$	→	$\overline{\text{noun prep}}$
$\overline{\text{verb}}$	→	$\overline{\text{verb adverb}}$	$\overline{\text{prep}}$	→	prep $\overline{\text{noun}}$
$\overline{\text{adverb}}$	→	degree adverb			

Figure 7.5
A sample Chomsky-normal-form "starter" grammar

While the overall structure of the parse is correct, there are a few errors.
Of these, the most striking, because the least explicable, is in the first tree in figure 7.7,
where the system splits up the obvious noun phrase "our time" by making a prepositional phrase "in our"
and using "time" in a very funny noun phrase whose topmost constituents are "Mr Moon time." This is strange because it requires what one would imagine to be a very unlikely rule, $\overline{prep}$ → prep det.

7.4 Improving a Partial Grammar

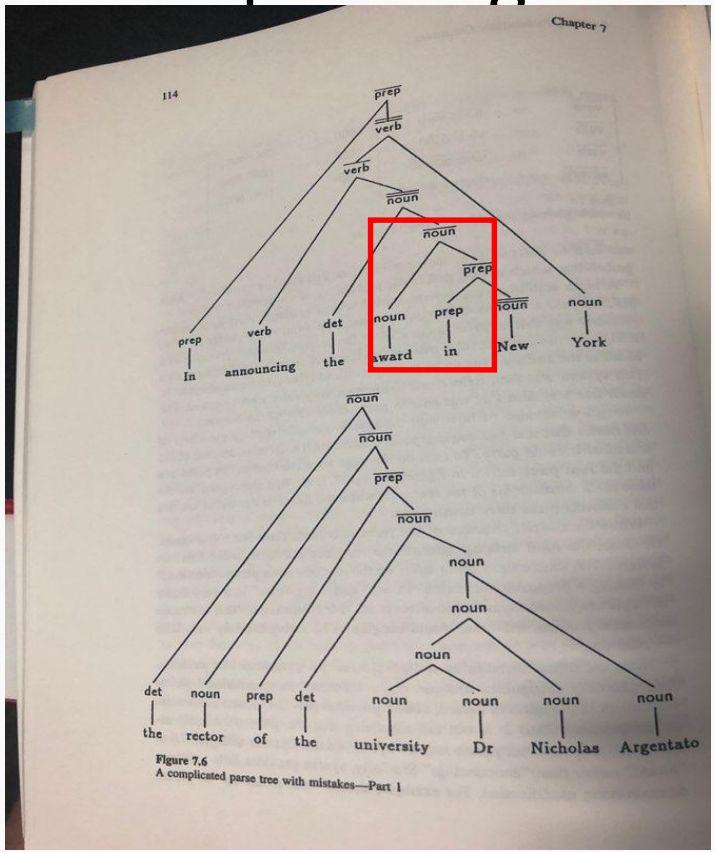


Figure 7.6
A complicated parse tree with mistakes—Part 1

[limit]

There are other mistakes as well, but these are somewhat less puzzling. It is generally recognized that the major influence on prepositional-phrase attachment is semantic in nature;
syntax provides only the loosest constraints on the process.
Thus it is not too surprising that the parse mistakenly attaches the prepositional phrase headed by “in” in the first tree of figure 7.6 to “award” rather than “announcing.”
Similarly, syntax provides little guidance on noun-noun modification.

7.4 Improving a Partial Grammar

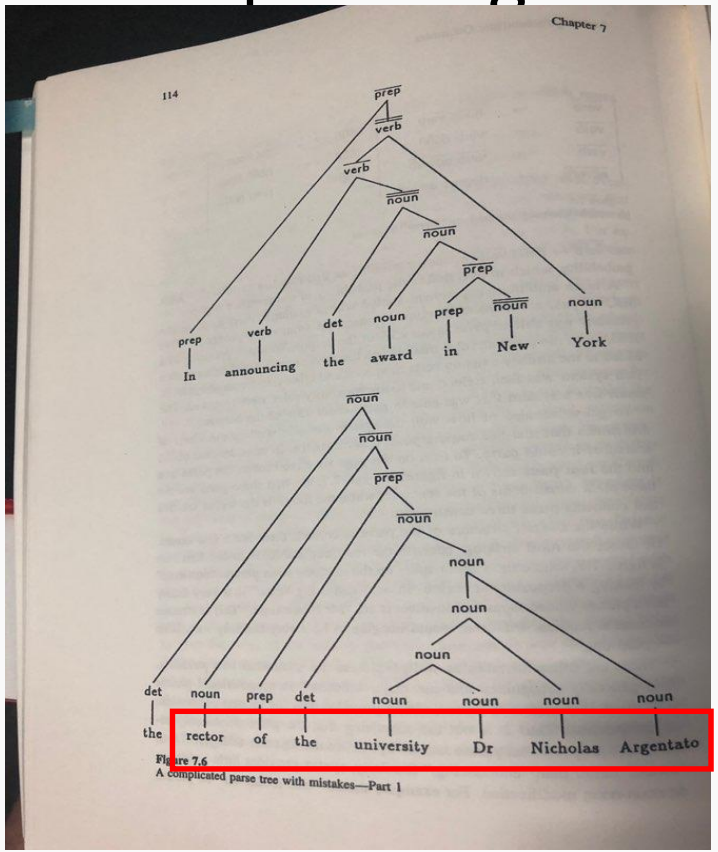


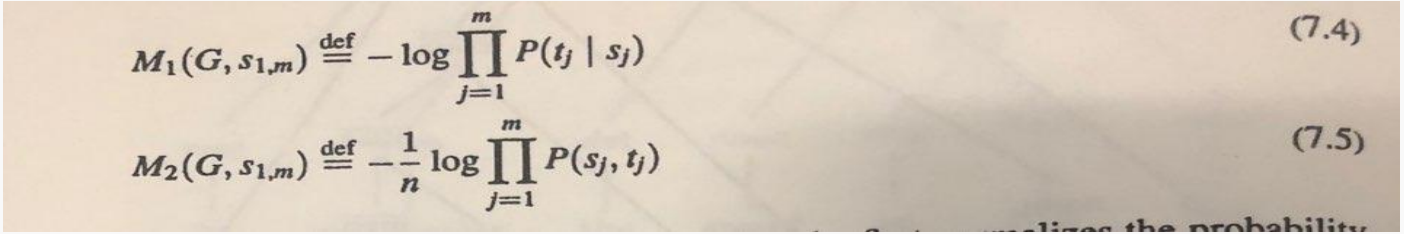
Figure 7.6
A complicated parse tree with mistakes—Part 1

[limit]

For example, the system split up "Dr Nicholas Argentato" but combined "university Dr" into a phrase. Presumably if the tag set distinguished honorifics(존대) and proper nouns from regular nouns it could have gotten this one right, but in general the problem is hard. Similarly for the mistake with "New York/' Thus it is probably not all that relevant how many mistakes there are in the most likely parse, as syntax in general does not have the information needed to make the right decision.

7.4 Improving a Partial Grammar

[limit]



$$M_1(G, s_{1,m}) \stackrel{\text{def}}{=} -\log \prod_{j=1}^{m} P(t_j \mid s_j) \quad (7.4)$$

$$M_2(G, s_{1,m}) \stackrel{\text{def}}{=} -\frac{1}{n} \log \prod_{j=1}^{m} P(s_j, t_j) \quad (7.5)$$

S = sentence, t = correct parses of s

M1 is more forgiving of a grammar that assigns low probability to all of the parses of a sentence, provided that G assigns the right parse the highest probability.
낮은 확률을 부여하는 문법을 좀 더 너그러이 용서하는 것이다.

M2, on the other hand, would rather that the right parse have a high probability, even if not the highest. Either one, however, is probably a better measure than correct attachment percentage
높은 확률을 가지는 것을 선호한다.

8. Syntactic Disambiguation

- 8.1 Simple Methods for Prepositional Phrases
- 8.2 Using Semantic Information
- 8.3 Relative-Clause Attachment
- 8.4 Uniform Use of Lexical/Semantic Information

8.1 Simple Methods for Prepositional Phrases

[*Concept*]

We remarked on this briefly in chapter 5 and noted there how prepositional-phrase attachment is particularly difficult for purely grammatical approaches because prepositional phrases attach to noun or verb phrases with approximately equal probability, as in :

- Sue bought a plant with Jane.

- Sue bought a plant with yellow leaves.

let A be a random variable representing this attachment decision.
As such it can take one of two values, verb if the pp attaches to the verb, and np1 if to the noun. Let w be the words in the text outside of "verb np1(prep np2)". We assume
P(A | prep, verb, np1, np2, w) = P(A | prep, verb, np1, np2)

This assumption is, false.
For example, “Fred saw a movie with Arnold Schwarenegger"

8.1 Simple Methods for Prepositional Phrases

[*Concept*]

We remarked on this briefly in chapter 5 and noted there how prepositional-phrase attachment is particularly difficult for purely grammatical approaches because prepositional phrases attach to noun or verb phrases with approximately equal probability, as in :

- Sue bought a plant with Jane.

- Sue bought a plant with yellow leaves.

let A be a random variable representing this attachment decision.
As such it can take one of two values, verb if the pp attaches to the verb, and np1 if to the noun. Let w be the words in the text outside of "verb np1(prep np2)". We assume
P(A | prep, verb, np1, np2, w) = P(A | prep, verb, np1, np2)

This assumption is, false.
For example, “Fred saw a movie with Arnold Schwarenegger"

8.1 Simple Methods for Prepositional Phrases

[*Concept & problem*]

Thus we next assume that the only thing relevant in a noun phrase is the head noun. For example, in "Sue bought a plant with yellow leaves" the only relevant aspect of "a plant" is "plant" and of "yellow leaves" is "leaves".
we assume that:
P(A | prep, verb, np1, np2, noun1, noun2) = P(A | prep, verb, noun1, noun2) --- (8.2)

Unfortunately, this is still far from being something for which we can collect statistics. Assume English has 10^4 nouns, 10^4 verbs, and 10 prepositions. Then the total number of combinations would be 10^13.
Obviously further simplifications are required

More formally, let f(p, n, a) be some measure of how much a prepositional phrase with preposition p and head noun n wants to attach to a—either a verb or a head noun of a noun phrase. Then we assume that:
P(A = noun | prep, verb, noun1, noun2) > P(A = verb| prep, verb, noun1, noun2)
<=> f(prep, noun2, noun1) > f(prep, noun2, verb)

= 10^9 + parameters — one set on how much the prepositional phrase likes the verb and another on how much it likes the noun.

8.1 Simple Methods for Prepositional Phrases

[Solution]
One experiment done in this vein is that by Hindle and Rooth [28].
Their solution to the sparse-data problem was to make a further simplifying assumption—namely, that the head noun of the prepositional phrase has no effect on the probabilities.

p(A | prep, verb, noun1, noun1) = P(A | prep, verb, noun1) (8.4)

P(A = noun | prep, verb, noun1) > P(A = verb | prep, verb, noun1)
<=> f(prep,noun1) > f(prep, verb), (8.5)

f(prep, verb) = P(prep | verb) (8.6)

f(prep, noun) = E(prep | noun) (8.7)

P(prep I verb) ≈ C(prep attached to verb)/C(verb) (8,8)

8.1 Simple Methods for Prepositional Phrases

[Solution]
One experiment done in this vein is that by Hindle and Rooth [28].
Their solution to the sparse-data problem was to make a further simplifying assumption—namely, that the head noun of the prepositional phrase has no effect on the probabilities.

p(A | prep, verb, noun1, noun1) = P(A | prep, verb, noun1) (8.4)

P(A = noun | prep, verb, noun1) > P(A = verb | prep, verb, noun1)
<=> f(prep,noun1) > f(prep, verb), (8.5)

f(prep, verb) = P(prep | verb) (8.6)

f(prep, noun) = E(prep | noun) (8.7)

P(prep I verb) ≈ C(prep attached to verb)/C(verb) (8,8)

8.2 Using Semantic Information

[*Concept*]

Before chapter, However, collecting data for the 109 data points required if we were to condition also on the head noun in the prepositional phrase does not seem practical, even if most of probabilities are effectively zero.

One solution to this problem is to condition not on the head nouns (nor, for that matter, on the verb) but rather on semantic tags associated with them.

For example, we might label “artist,” “Jane,” “plumber,” “Ted” as human, “Friday," “June,” “yesterday" as time, “hammer,” “leaves,” “pot” as object, and so on. This particular classification is somewhat arbitrary, but one can see how to use it to distinguish the two attachment problems given at the start of this chapter:

- Sue bought a plant with Jane.
- Sue bought a plant with yellow leaves.

8.2 Using Semantic Information

[*Concept*]
In terms of our earlier formalization, they assume that

$$P(A = \text{noun} \mid \text{prep}, \text{verb}, \text{noun}_1, \text{noun}_2)$$
$$> P(A = \text{verb} \mid \text{prep}, \text{verb}, \text{noun}_1, \text{noun}_2)$$
$$\Leftrightarrow f(\text{prep}, S(\text{noun}_2), S(\text{noun}_1)) > f(\text{prep}, S(\text{noun}_2), S(\text{verb})) \quad (8.130)$$

where S(x) is the semantic tag of x and where

f(prep, S(noun2), S(verb)) = P(S(noun2), S(verb) |prep)

8.2 Using Semantic Information

[*Problem*]
figure 8.2 shows significant triples involving the Italian preposition "da" (the corpora for this work were in Italian).

Attachment location semantics	Prep	Head noun semantics	Italian example	English translation
artifact	da	artifact	biancheria da letta	linens for bed
physical_act	da	human_entity	venduto da i soci	sold by the shareholders
artifact	da	place	articoli da spiaggia	items for beach
physical_act	da	building	comprare da oleifici	buy from oil refineries

Figure 8.2
Significant semantic pairs for the Italian preposition "da"

8.3 Relative-Clause Attachment

[*Concept*]

Another source of syntactic ambiguity is the attachment of relative clauses. Consider the following examples:
(1) Fred awarded a prize to the dog and Bill, who trained it.

(2) Fred awarded a prize to Sue and Fran, who sang a great song.

(3) Fred awarded a prize for penmanship that was worth $50.00.

(4) Fred awarded a prize for the dog that ran the fastest.

Often a distinction is made between restrictive relative clauses and unrestrictive relative clauses.
Example (4) above is the former, while (1-3) are the latter.

$$P(\text{relative clause attaches to } x \mid \text{main verb of clause} = v)$$
$$> P(\text{relative clause attaches to } y \mid \text{main verb of clause} = v)$$
$$\Leftrightarrow P(x = \text{subject/object} \mid v) > P(y = \text{subject/object} \mid v)$$

8.3 Relative-Clause Attachment

[*Problem*]
in all the examples found by Fisher and Riloff the noun phrase serves as the subject o
the relative clause, and that is the only case they handle. However, they do find examples in which the relative clause is passive; that is, the entity in the subject position is logically the direct object and the noun phrase provided by a “by” prepositional phrase, if such a phrase exists, is the sentence’s logical subject. For example.
Sue ate a cookie that was baked by Fran.
… Fran who baked a cookie (원래는 이게 맞음)

Thus this scheme starts by collecting “subject-verb” and “verb-object” pairs to see which verbs tend to take what sorts of things as either their subjects or objects.

8.4 Uniform Use of Lexical/Semantic Information

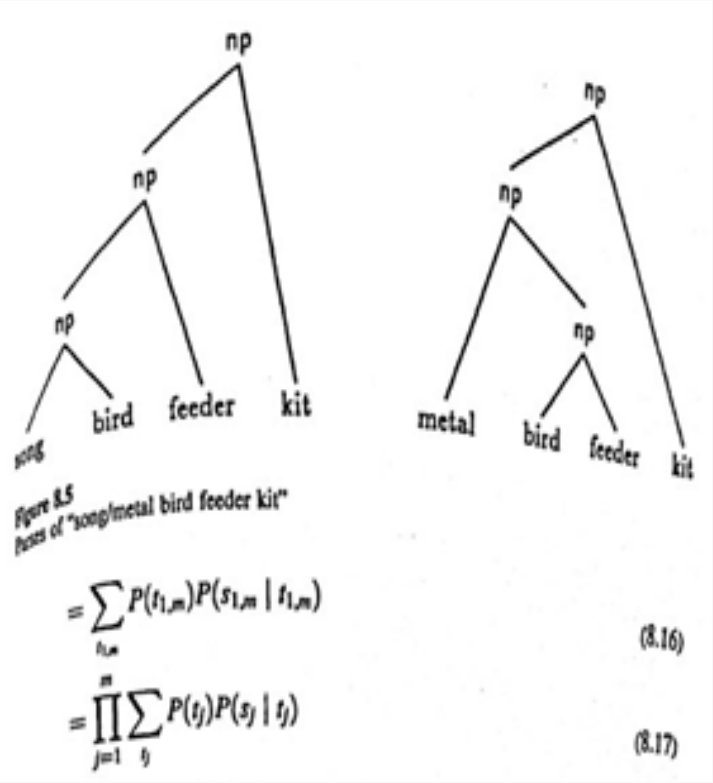


Figure 8.5
Parses of "song/metal bird feeder kit"

$$= \sum_{t_{1,m}} P(t_{1,m}) P(s_{1,m} \mid t_{1,m}) \qquad (8.16)$$

$$= \prod_{j=1}^{m} \sum_{t_j} P(t_j) P(s_j \mid t_j) \qquad (8.17)$$

[*Concept*]

combinations as “song bird feeder kit,”
“metal bird feeder kit,” and
“novice bird feeder kit."
The most reasonable parses for the first two are shown in figure 8.5.
Presumably if we collected statistics we would find that “song” modified “bird” more than either “feeder” or “kit,”
while “metal” had the opposite tendency and “novice” was more likely to modify “kit” than any of the other words

Thus here too statistics could be placed in the service of syntactic disambiguation.

8.4 Uniform Use of Lexical/Semantic Information

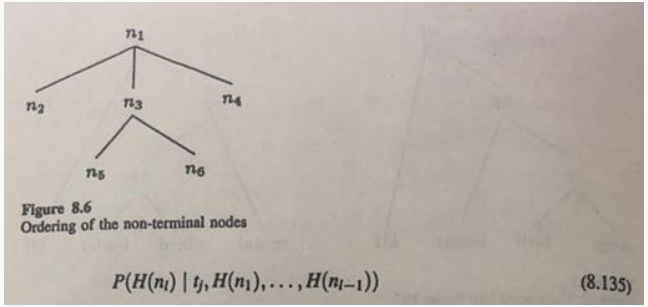


Figure 8.6
Ordering of the non-terminal nodes

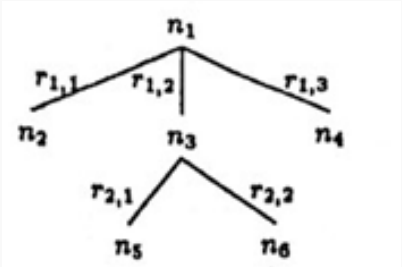


Figure 8.7
Syntactic relations between nodes

[*Concept*]

The idea now is to assume that the probability that a word is the head of a given constituent is independent of everything except
(1) the word that is the head of its immediate parent
(2) the syntactic relationship between it and its immediate parent.

we call $r_{a,b}$, the relationship between the head of the ath rule of the grammar and the bth constituent on its right-hand side. Say the two rules used in producing the tree in figure 8.6 are rules 1 and 2.

We see there that the syntactic relationship between n1 and n3 is called $r_{1,2}$

8.4 Uniform Use of Lexical/Semantic Information

[*Concept*]

Let the parent (mother) of n be M(n) and
let the syntactic relation between n and M(n) in t_j be R(n, t_j).
Then we assume that:

$$P(H(n_l) \mid t_j, R(n_l, t_j), H(n_1), \ldots, H(M(n_l)), \ldots, H(n_{l-1})) = P(H(n_l) \mid R(n_l, t_j), H(M(n_l)))$$

Using this assumption to simplify equation 8.18, we get

$$P(s_j \mid t_j) = P(H(n_1)) \prod_{l=2}^{l} P(H(n_l) \mid H(M(n_l)), R(n_l, t_j)) \quad (8.136)$$

To make this more concrete, consider the following context-free rules:

r_1 : vp → verb np pp

r_2 : np → det noun pp

r_3 : pp → prep np

8.4 Uniform Use of Lexical/Semantic Information

[*Concept & Problem*]

But the major contribution to the number of parameters required is
not the number of syntactic relations but the number of lexical items,
since we need to collect statistics for each pair of such items.

Ultimately, to make the statistics-gathering tractable, words will have to be grouped into clusters with similar distributional properties.

This is what the semantic tags of [19] and [4] were intended to do. However,
in both of these studies the tags were attached by hand and thus the total number of words tagged was small(문제점)

8.4 Uniform Use of Lexical/Semantic Information

[*Solution*]

the tags were 'de ned' in terms of their semantics but were to be used to predict more “surfacey” phenomena like syntactic disambiguation.
Thus we are assuming that words that behave in the same way on the surface have semantic similarities.

If this is indeed the case, we might be able to create such groups automatically by collecting statistics on how words behave at the surface level and grouping words with similar statistics.
In the next chapter we see that this is indeed a promising approach.