COSC343: Artificial Intelligence

Lecture 24: Statistical Grammars

Alistair Knott

Dept. of Computer Science, University of Otago

In today's lecture

- Parsing and ambiguity
- Wide-coverage grammars and the problem of proliferating ambiguity
- Probabilistic grammars, and how to learn them

The limits of a linear language model

We've seen that in order to describe the range of sentences in a human language, we need to build a grammar, which assigns each sentence a hierarchical structure.

- On this model, sentences are not simple sequences—they're *trees*.

What we need to do is to build a probability model which works with a grammar.

- One useful approach is to build a model that lets us *disambiguate* if there are multiple possible parse trees.

Parsing and ambiguity

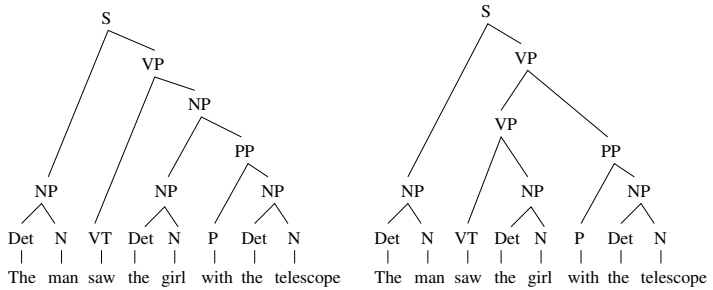
Consider the following context-free grammar:

S → NP, VP	PN → *John*
NP → Det, N	N → *man, girl, telescope*
NP → PN	VT → *saw*
PP → P, NP	Det → *the*
VP → VT, NP	P → *with*
NP → NP, PP	
VP → VP, PP	

Q: How will this grammar parse the following sentence?

The man saw the girl with the telescope.

Parsing and ambiguity



Note: a parser will find *all possible syntactic analyses* of a sentence.

Wide-coverage grammars

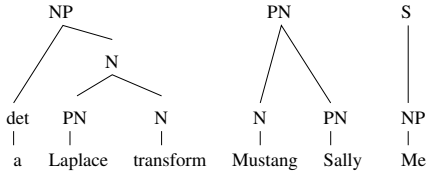
Until about 20 years ago, computational linguists focussed on building small grammars, with small lexicons.

But if we want to build useful natural language processing systems, we need grammars which deal with a **huge** range of sentence structures and words.

Problems in building 'real' grammars

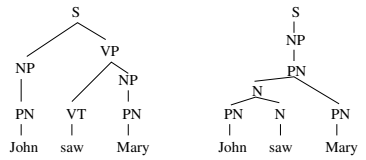
We certainly need include rules to deal with the following constructions:

- *Here's the equation for a* **Laplace transform**.
- **Mustang Sally** *walked into the bar.*
- *Who volunteers?* **Me**.



An example of overgeneration

How will the resulting grammar deal with the sentence *John saw Mary* ?



Point: A wide-coverage grammar gives us **spurious ambiguities**.

Towards a solution: probabilistic grammars

How can we disambiguate in these cases?

- A **logic-based** approach: find the meaning of each interpretation, and choose the one which fits best with world knowledge.
  But this sounds like an 'AI-complete' task.
- A **statistical approach**: look at *how frequently particular rules are actually used*.
  - *Laplace transform* and *Mustang Sally* are relatively rare constructions.
  - The correct syntactic structure of *John saw Bill* occurs very frequently.

Assembling a training corpus

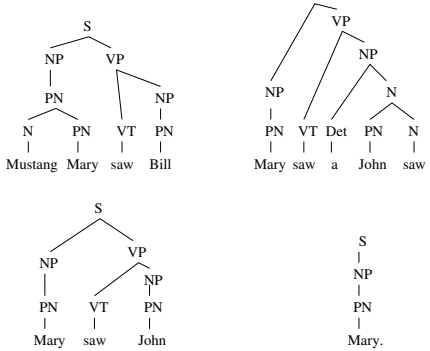
The first thing to do is to gather a (large) set of sentences which are representative of the kind of sentences you want to parse.

The method is then as follows:

- Parse each sentence by hand, to identify what the correct analysis is for each.
- Count the number of occurrences of each rule application.
- Use these counts to build a **probabilistic grammar**, where every rule is associated with a probability.

When you parse a sentence with a probabilistic grammar, disambiguation is easy: you just choose the *most likely parse*.

An example training corpus



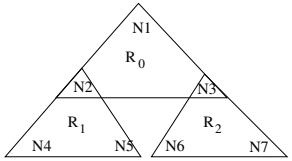
Counting rules in the corpus

Rule	Frequency in corpus
S → NP, VP	3
S → NP	1
NP → PN	6
NP → Det, N	1
VP → VT, NP	3
N → PN, N	1
N → mustang	1
N → saw	1
VT → saw	1
det → a	1
PN → Mary	2
PN → John	1
PN → N, PN	1
	Total: 23 rule applications

Computing the probability of a parse tree

Let's think of a tree as a set of **rule applications** $R_1 \ldots R_n$.

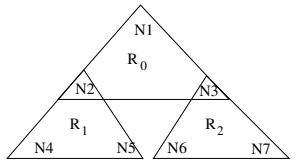
- Each rule application R_i has a **parent node** and one or more **child nodes**.



First idea: 'since we're using a context-free grammar, the probabilities of each rule application are *independent*. So we can multiply together $p(R_i)$ for each rule application in the tree to get the probabiltity of the whole tree.'

- But this isn't quite right: the children of the top rule application tell us what the parents of lower rule applications are.

Conditional probabilities in a parse tree



We need to work with the **conditional probabilities** of rule applications given knowledge of the parent node.

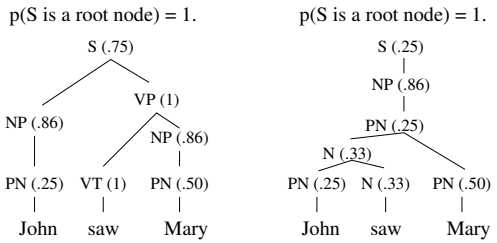
- Because we're working with a context-free grammar, these probabilities really are independent.
- So to get the probability of the whole tree, we first compute $p(R|Parent)$ for each rule application R, and then multiply all these probablities.
- We then multiply the result by the prior probability that the root node N_1 is a root node.

Estimating probabilities from the corpus

Rule	Frequency in corpus	Estimated $p(Rule\|Parent)$
S → NP, VP	3	3/4 = .75
S → NP	1	1/4 = .25
NP → PN	6	6/7 = .86
NP → Det, N	1	1/7 = .14
VP → VT, NP	3	1
N → PN, N	1	1/3 = .33
N → mustang	1	1/3 = .33
N → saw	1	1/3 = .33
VT → saw	1	1
det → a	1	1
PN → Mary	2	2/4 = .5
PN → John	1	1/4 = .25
PN → N, PN	1	1/4 = .25

Est'd prior prob of a node being a root node: 1 if it's S; 0 otherwise.

Using rule probabilities to disambiguate



Left-hand interpretation:
$p(tree) = 1 \times .75 \times .86 \times .25 \times 1 \times 1 \times .86 \times .5 = .0693$
Right-hand interpretation:
$p(tree) = 1 \times .25 \times .86 \times .25 \times .33 \times .25 \times .33 \times .5 = .0007$

Top-down and bottom-up probability models

When we defined the probability of a tree in terms of conditional probabilities of rule applications, we used a **top-down** probability model, involving:

- $p(\mathit{Rule\_Application}|\mathit{Parent})$;
- the prior probability of the root node in the tree being a root node.

Alternatively, we could use a **bottom-up** model, involving:

- $p(\mathit{Rule\_Application}|\mathit{Child}_1 \wedge \ldots \mathit{Child}_n)$;
- the prior probability of the words in the tree appearing in a sentence.

Summary and reading

In the section on language, you've seen:

- Something about how language relates to action and goals.
  (Linking to the 'agent-focused' part of the course.)
- Something about probabilistic language models.
  (Linking to the 'probabilistic reasoning' part of the course, and also to the 'learning' parts (neural networks, information theory, unsupervised learning. . .)
- Something about natural language syntax.
  (An example of 'symbolic AI'.)
- Something about parsing.
  (Linking to the 'search' part of the course.)

Reading for this lecture: AIMA Chapter 23.2.1
No reading for next lecture