1. Visualising CKY parsing: The empty chart

Just the empty matrix

2. Visualising the Chart (0,1)

Unary branching rules: Det → the

3. Visualising the Chart (1,2)

Unary branching rules: N → frogs, Nom → N, NP → Nom

4. Visualising the Chart (2,3)

Unary branching rules: TV → ate

5. Visualising the Chart (3,4)

Unary branching rules: N → fish, Nom → N, NP → Nom, IV → fish, VP → IV

6. Visualising the Chart (0,2)

Binary branching rule: NP → Det Nom

- (0,1) & (1,2) ⇒ (0,2)
7. Visualising the Chart (1,3)

8. Visualising the Chart (2,4)
Binary branching rule: VP → TV NP
• (2,3) & (3,4) ⇒ (2,4)

9. Visualising the Chart (0,3)
(0,1) & (1,3) ¬⇒
(0,2) & (2,3) ¬⇒

10. Visualising the Chart (1,4)
Binary branching rule: S → NP VP
(1,3) & (3,4) ¬⇒
(1,2) & (2,4) ⇒ (1,4)

11. Visualising the Chart (0,4)
Binary branching rule: S → NP VP
(0,1) & (1,4) ¬⇒
12. From CKY Recogniser to CKY Parser

We cannot tell from the CKY chart as specified, the syntactic analysis of the input string.

We just have a chart recogniser, a way of determining whether a string belongs to the language generated by the grammar.

Changing this to a parser requires recording which existing constituents were combined to make each new constituent. This requires another field to record the one or more ways in which a constituent spanning (i,j) can be made from constituents spanning (i,k) and (k,j).

13. Ambiguity is still the problem

Features help express the grammar neatly, but they don’t change the ambiguity problem.

Active chart parsing gives us flexibility, but that doesn’t solve the ambiguity problem.

Examples of ambiguity:

- **global PP attachment**
  - “One morning I shot an elephant in my pyjamas…” (Groucho Marx)
  - Was the elephant in Groucho’s pyjamas?!
  - No, Groucho was wearing the pyjamas.
- **gerundive VP attachment**
  - We saw penguins flying over Antarctica
  - Who was flying, us or the penguins?
  - Contrast “We saw penguins carrying fish in their beaks” with “We saw penguins using high-powered binoculars”
- **coordination**
  - hot tea and coffee vs empty bottles and fag-ends
  - The most likely readings are that both beverages are hot, but only the bottles are empty.

14. Compositional semantics, again

The significance of attachment ambiguity is clear when we look at semantics.

Back to Python arithmetic expressions, with an even simpler grammar:

\[
\text{Expr} \rightarrow \text{Expr} \text{ Op} \text{ Expr} | \text{Var} | \text{Number}
\]

\[
\text{Op} \rightarrow + | - | \times | /\]

Assuming some more work to tokenize the input, this will give us two analyses for the string

\["x/y+1"\]

We see that the difference in attachment makes a very real difference.

15. Compositional semantics, cont’d

How does this relate to natural language?

- Many approaches to language understanding proceed in a similar way.
  - In what’s called a rule-to-rule way.
  - Associating a (compositional) semantic rule with each syntactic rule.

So, for example, for the kind of PP attachment ambiguity we keep encountering:

- Where the PP is attached determines where its semantic contribution is made.
- In other words, with respect to “saw the child with the telescope”

\[
\text{VP} \rightarrow \text{VP PP}
\]

The semantics of the PP contributes to the semantics of the (higher) VP.

- The seeing was done with the telescope.

\[
\text{VP} \rightarrow \text{VP PP}
\]

The semantics of the PP contributes to the semantics of the (higher) VP.

- The seeing was done with the telescope.

16. Back to ambiguity

We need a way to choose the best analysis from among many.

- Very many...

  - The average sentence in the Wall Street Journal dataset we will use in the labs this week has about 24 words.

  - And there are many local ambiguities, e.g., a broad-coverage CF-PSG will have thousands of parses.

And we need a sound basis for ranking these.

A gold standard provides at least partial solutions.

17. Treebanks: big investment, big reward

Mitch Marcus at the Univ. of Pennsylvania took this seriously.

- As did the US DARPA (Defense Advanced Research Projects Agency)...

  - In the context of their evaluation-led research funding approach.

The Penn Treebank project was launched in 1989.
18. Treebank examples and evolution

The first release contained 'skeletal' parses, that is, simple syntactic trees.

- With a few quirks based on a choice of underlying grammatical theory
- New grammatical gaps, this mostly involves the use of what are called traces to indicate missing material
- An example in relative clauses and some complement clauses
- It turns out if difficult to fit the traces properly

Here's an example:

```
NP (NP Battle-tested Japanese industrial managers)
|   |   |
NP  C  NP
   |   |
S   VP
```

This annotation was produced by manually correcting the output of a simple chunking parser, then removing the POS tags and adding some functional information.

19. Penn Treebank, mark two

The second release added much more information, including some hyphenated functional indications:

```
NP (NP Battle-tested Japanese industrial managers)
|   |   |
NP  C  NP
   |   |
S   VP
```

So, for example, given that in the NLTK treebank subset, there are 52041 subtrees labelled NP
And in 29201 of these there are exactly two children of NP-SBJ
So how do you evaluate a parser against a treebank?
For every node in every tree with non-lexical children
- Add a score to the grammar (or increment the count of an existing rule)
- No backtracking (CWS/CLAS 15622/Lab
To answer that, there is a variety of tools, we'll get to that
The result is guaranteed to give at least one parse to every sentence in the treebank

20. Deriving a grammar from a treebank

```
(ADJP Battle-tested Japanese industrial)
|   |   |
NP  C  NP
   |   |
S   VP
```

This is a simple and fairly small grammar
For a treebank-derived ruleset, the maximum likelihood probability estimate is simple:

\[
P(NT \rightarrow C_1C_2...C_n | NT) = \frac{\text{count}(NT \rightarrow C_1C_2...C_n)}{\text{count}(NT)}
\]

The functional markers here are
- SBJ Participant argument
- LOC Location modifier
- TIR Temporal modifier
- CVA Specifier of argument

21. Simple Probabilistic CF-PSGs

Given a treebank, we can easily compute a simple kind of probabilistic grammar
- Either directly with respect to treebank-derived rules
- Or we can use the sum of the costs for simpler calculations

In its simplest form, it just counts parenthesis-crossing
```
P(NT \rightarrow C_1C_2...C_n | NT) = \sum_{i=1}^{n} P(C_i)
```

22. Evaluation against a treebank

For every grammar, the sum of the probabilities of all non-leaf nodes is 1
So how do you evaluate a parser against a treebank?

If it turns out just looking at major constituent boundaries is surprisingly good
```
P(NT \rightarrow C_1C_2...C_n | NT) = \prod_{i=1}^{n} P(C_i)
```

23. PARSEVAL examples

```
P(NT \rightarrow C_1C_2...C_n | NT) = \prod_{i=1}^{n} P(C_i)
```

The functional markers here are
- SBJ Subject argument
- LOC Location modifier
- TIR Temporal modifier
- CVA Specifier of argument

24. Treebank grammar problems

The business likely using the treebank subset that ships with NLTK, there are 1788 different rules:
- Some are very common, such as S \rightarrow NP-SBJ VP
- Most occur only once
- As always, there's a trade-off between specificity and statistical significance
- More detailed symbols
- Means more syntactic
- Means fewer examples of each for getting good PARSEVAL scores
- Learned from treebanks, grammars tend to collapse many of the release-2 categories
- This is something new since the release-1 counterparts