Accelerated Natural Language Processing 2016

Lecture 16: Parsing probabilistic CFGs: Beam search, figures of merit

Henry S. Thompson
20 October 2016

1. Treebank grammar problems

The bad news: just using the treebank subset that ships with NLTK, there are 1798 different expansions for **S**

- Some are very common, such as S → NP-SBJ VP
- Most occur only once (Zipf again)

As always, there's a tradeoff between specificity and statistical significance

- More detailed symbols
  - means more symbols
  - means fewer examples of each

For getting good PARSEVAL scores

- learned-from-treebank grammars tended to collapse many of the release-2 categories back to something more like their release-1 counterparts

2. The realities of PCFG parsing

Before we look more closely at some of the in-principle problems with massive PCFGs

- Such as we get in the case of built-from-treebanks grammars

We'll look at some practical difficulties

Multiple tags per terminal (word)

- Plus 100s, if not 1000s, of rules for some non-terminals (categories)

Means 100s of thousands of edges in a probabilistic chart parser

If we're working with spoken language, the numbers are even worse

- As there will be multiple alternative hypotheses about the words in the utterance

- "people can easily recognise speech"
- "people can easily wreck a nice beach"

Finding *all* the parse trees, so that you are sure to find the best, is often therefore out of the question

- Charniak reports, for instance, that getting 95% of the way to finding *all* parses
- Of a 30-word sentence from the Brown corpus
- With a PCFG constructed from the Brown corpus
- Took 130,000 edges

More recent experiments

- with highly optimised representations of the parse trees
- required 24 *gigabytes* of storage to hold complete sets of parses

3. Best-first? Not so fast. . .

So although what I said last week is true in principle

- That is, that maintaining an *ordered* edge queue makes a chart parser best-first
- In *practice* the cost of doing so is very high
    - *Prohibitively* high for broad-coverage probabilistic grammars
- Because it turns into breadth-first search across all possible parses constructed left-to-right

Why is this?

- In the first instance, because of the product of probabilities problem

4. Multiplying probabilities

. . . produces small numbers quickly

So short analyses are almost always are more probable than long ones

- Consider the trivial case of the two word phrase "the men"
- Here are the MLE estimates of five relevant probabilities
    - Taken from the same data we used in the lab last week

DT → 'the'	0.49455
JJ → 'the'	0.0008570
NNS → 'men'	0.001653
NP-SBJ → DT NNS	0.017011
NP-SBJ → JJ NNS	0.010468

And here are the costs (that is $-\log_2(\text{prob})$)

DT → 'the'	1.02
JJ → 'the'	10.19
NNS → 'men'	9.24
NP-SBJ → DT NNS	5.88
NP-SBJ → JJ NNS	6.58

We'll see that even though analysing 'the' as **DT** is *500* times more likely than analysing it as **JJ**

- We'll still keep taking both analyses forward through a best-first parse to the very end

5. Multiplying probabilities, cont'd

Using those costs, what happens as we run a chart parser bottom-up

- Doing lowest-cost first sorted insertion into the agenda

DT → 'the'	1.02
JJ → 'the'	10.19
NNS → 'men'	9.24
NP-SBJ → DT NNS	5.88
NP-SBJ → JJ NNS	6.58

Agenda		**Chart**	
$_0$the$_1$	0.0		
$_1$men$_2$	0.0		
		$_0$the$_1$	0.0
		$_1$men$_2$	0.0
$_0$DT → · 'the'$_0$	1.02		
$_1$NNS → · 'men'$_1$	9.24		
$_0$JJ → · 'the'$_0$	10.19		
		$_0$DT → · 'the'$_0$	1.02
$_0$DT['the']$_1$	1.02		
$_1$NNS → · 'men'$_1$	9.24		
$_0$JJ → · 'the'$_0$	10.19		
		$_0$DT['the']$_1$	1.02
$_0$NP-SBJ → · DT NNS$_0$	5.88		
$_1$NNS → · 'men'$_1$	9.24		
$_0$JJ → · 'the'$_0$	10.19		
		$_0$NP-SBJ → · DT NNS$_0$	5.88
$_0$NP-SBJ → DT · NNS$_1$	6.90		
$_1$NNS → · 'men'$_1$	9.24		
$_0$JJ → · 'the'$_0$	10.19		
		$_0$NP-SBJ → DT · NNS$_1$	6.90
		$_1$NNS → · 'men'$_1$	9.24
$_1$NNS['men']$_2$	9.24		
$_0$JJ → · 'the'$_0$	10.19		
		$_1$NNS['men']$_2$	9.24
$_0$JJ → · 'the'$_0$	10.19		
$_0$NP-SBJ[DT NNS]$_2$	16.14		
		$_0$JJ → · 'the'$_0$	10.19

abbreviating slightly . . .

		$_0$JJ[the]$_1$	10.19
		$_0$NP-SBJ → · JJ NNS$_0$	6.58
$_0$NP-SBJ[DT NNS]$_2$	16.14		
$_0$NP-SBJ → JJ · NNS$_1$	16.77		

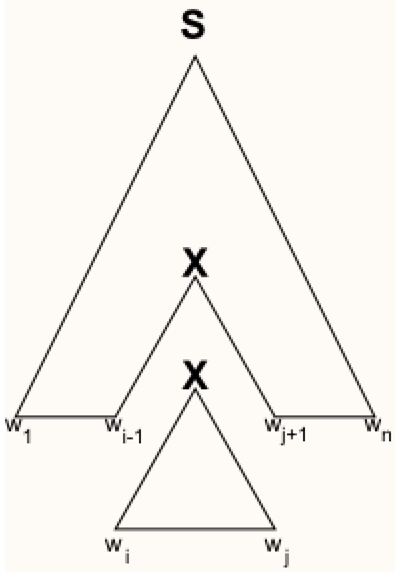
And so it will go on

- As soon as the **S → NP-SBJ ...** edges that are about to go into the agenda, and then the chart, consume the NP
- Their cost will soar above the 16.77 of the silly active NP-SBJ edge, which will take another step forward
- *Doubling* the number of edges on the agenda shortly thereafter
    - As all the waiting empty active **S** edges consume the NB-SBJ it results in

6. Ordering the agenda: details

What we've been using to order the agenda is called the **inside** probability

- In practice, the inside **cost**



- That is, the probability for some node X that it expands to cover what it covers
- $P(\mathrm{NT} \rightarrow^* w_\mathrm{i} \ldots w_j \mid \mathrm{NT})$
- It's also helpful to define the notion of **outside** probability:
    - The probability that the *rest* of the tree is what it is
    - $P(S \rightarrow^* w_1 \ldots w_{\mathrm{i}-1} X w_{\mathrm{j}+1} \ldots w_n)$

Using the inner probability to sort the agenda will clearly prefer smaller trees

- We need to introduce some kind of normalisation to avoid this
- Understanding as we do so that we may thereby put at risk our goal of getting *the best* parse first

7. Figures of merit

The name for what we're looking for is a **figure of merit**

- That is, some non-decreasing measure of (partial) subtree cost

There are lots of possibilities

- Of which most obvious is also the simplest
- Inner cost, normalised by word span

This would clearly have the desired effect in our worked example above

- The cost of the first inactive **NP-SBJ** edge is divided in half
    - From `16.14` to `8.07`
    - Thereby ensuring that it will be processed *before* the implausible 'the'-as-adjective hypothesis

Note that normalising in the cost domain uses the *arithmetic* mean

- because we've been *summing* costs

In the probability domain, we use the *geometric* mean

- because we've been *multiplying* probabilities

Using the left half of the outer cost as well improves performance further

- In principle
- But in practice takes too much time to compute

See Caraballo and Charniak 1996 for the details

8. Beam search

Even with a good figure of merit, our chart will still grow very large

- If we pursue every hypothesis, no matter how expensive

So standard practice is to **prune** the agenda

- That is, set a maximum number of edges we will hold
- Or a maximum delta between the best and worst that we will hold

The result is called **beam search**

- And the relevant parameter the **beam width**

Whenever the agenda is full

- that is, has the number of entries specified by the beam width
- and we need to insert an edge

There are two possibilities (ignoring ties)

- If the new edge is more expensive than the most expensive edge in the agenda
    - We discard the new edge
- Otherwise we discard the current most expensive edge
    - and insert the new edge at its appropriate place in the agenda

9. Evaluating PCFG parsers

With figures of merit and beam search

- We've definitely lost any guarantee of best-first
- Indeed, we may not even have best-*ever*
    - If our beam-width is so narrow that it makes us discard some expensive prefix
    - Of what would ultimately have been the cheapest analysis

- And in any case, we need to evaluate our ranking

So evaluation has to look not only at the first and/or best parse

- when comparing to the gold standard

But also the top 5

- Or top 10
- Etc.

Asking, e.g.

- How often does the correct parse appear within the top 10?
- What is the position of the parse with the best PARSEVAL score among the top 20?

10. Back to intrisinc PCFG problems

The expansion of the tagset for the Penn Treebank was for a good reason

- For example, distinguishing subject NPs (**NP-SBJ**) from others (plain **NP**)
- Even though this reduced the sample size for both

It was an attempt to address one part of the problem with PCFGs

- Namely, that they are context-free :-)

Linguistically, that's not a problem

- But for parsing, it can be

Although what an NP *is* doesn't depend on context

- An NP is an NP wherever it occurs
- Any expansion of that non-terminal in the grammar is allowed anywhere

The *probabilities* of different expansions *do* change with context

11. Probabilities in context, cont'd

For example, in the Switchboard corpus of transcribed telephone conversations, the probability of **NP → Pronoun**

in subject position 0.91
in object position 0.34

That's extreme, because of the source, presumably

But even for our little test corpus, the effect is still there:

in subject position 0.15
in object position 0.012

Given that, the use of a special category for NPs in subject position makes sense

- This can be automated and generalised
- By splitting categories by their parents
- E.g. **NP^S** (instead of **NP-SBJ**) vs. **NP^VP** and **NP^PP**