Accelerated Natural Language Processing 2018

Lecture 10: Why grammar?

Henry S. Thompson
8 October 2018

1. Taking a step back

The first three weeks have introduced the concept of a **language model**

- Of the information theoretic variety

Along with a number of other key technologies and methodologies:

- Finite state machines and transducers
- N-gram models
- Hidden Markov models
- Viterbi search and friends
- Smoothing and interpolation

These all find a place in systems we can understand in terms of the **noisy channel model**

- Or, perhaps more accurately, *meta*model

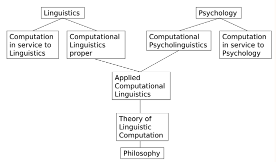
Where we're headed next moves away, at least temporarily, from that narrative

2. Our complex history

The present state of NLP can be understood best by reference to the history of the computational encounter with natural language

First closely parallel to, latterly increasingly separated from, the history of linguistic theory since 1960

Situated in relation to the complex interactions between linguistics, psychology and computer science:



Originally all the computational strands except the 'in service to' ones were completely invested in the Chomskian rationalist perspective

- With a corresponding commitment to
  - formal systems
  - representationalist theories of mind
  - so-called 'strong AI'

3. An aside about hyphenated disciplines

Interdisciplinary work runs the risk of mistaking its hyphen for a license to make up the rules

- That is, to more-or-less ignore the normative structure of *either* of the contributing disciplines

But this just results in a loss of credibility

- From *both* sides

So actually much *more* so than unhyphenated practioners

- Computational linguists of whatever variety have to be very explicit about the rules they are working by

4. Linguistics + Computing

Computational linguistics initially drew heavily on 20th-century linguistics

So drew extensively on algebra, logic and set theory:

- Already well established in formal language theory
  - Turing, Church, Tarski
- Adopted and significantly developed by computer scientists for use in compilers
  - Aho, Hopcroft, Ullman
- Exploited for natural language analysis
  - Chomsky, Montague

Then added parsing and 'reasoning' algorithms to grammars and logical models

parsing
: Top down, bottom up, dynamic programming. . .

reasoning
: model building, theorem proving, truth maintenance

5. The empir[icist] strikes back

Starting in the late 1970s, in the research community centred around the (D)ARPA-funded Speech Understanding Research effort, with its emphasis on evaluation and measurable progress, things began to change

(D)ARPA funding significantly expanded the amount of digitised and transcribed speech data available to the research community

Instead of systems whose architecture and vocabulary were based on linguistic theory (in this case acoustic phonetics), new approaches based on statistical modelling and Bayesian probability emerged and quickly spread

> *Every time I fire a linguist my system's performance improves*

> Fred Jellinek, head of speech recognition at IBM, c. 1980 (allegedly)

6. It all started with speech recognition

The speech signal under-determines what we 'hear'

- By the late 1970s, this was regularly thought of in terms of a lattice of possible words with varying degrees of support from the acoustic evidence
- Consider for example [this short utterance]

r ɛ k ə n a i s b ii ʧ

You heard:

recognise

speech

But I said:

a *nice*

wreck *beach*

And there are more possibilities:

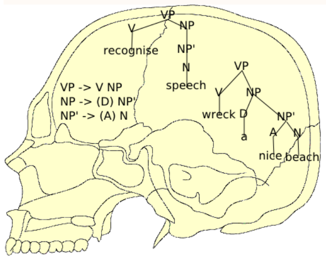
reckon

an *ice*

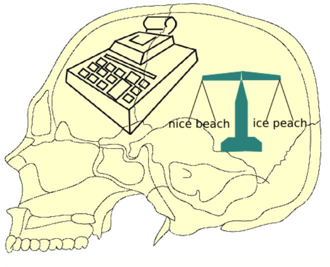
7. Speech recognition, cont'd

So how do we select the right path through the word lattice?

Is it on the basis of a small number of powerful things, like grammar rules and mappings from syntax trees to semantics?



Or a large number of very simple things, like word and bigram frequencies?



In practice, the probability-based approach performs *much* better than the rule-based approach

8. Up the speech chain

The publication of 6 years of digital originals of the *Wall Street Journal* in 1991 provided the basis for moving the Bayesian approach up the speech chain to morphology and syntax

Many other corpora have followed, not just for American English

And the Web itself now provides another huge jump in the scale of resources available

To the point where even semantics is at least to some extent on the probabilistic empiricist agenda

9. The new intellectual landscape

Whereas in the 1970s and 1980s there was real energy and optimism at the interface between computational and theoretical linguistics

- by the end of the century, the overwhelming success of the empiricist programme in the applied domain had separated them once again

While still using some of the terminology of linguistic theory

- computational linguistics practioners became increasingly detached from theory itself
- And theory suffered a, perhaps connected, loss of energy and sense of progress

Within cognitive psychology, significant energy began going in to erecting a theoretical stance consistent with at least some of the new empiricist perspective

But the criticism voiced 35 years ago by Herb Clark, who described cognitive psychology as "a methodology in search of a theory", remains pretty accurate

And within computer science in general, and Artificial Intelligence in particular, the interest in "probably nearly correct" solutions, as opposed to contructively true ones, is dominant

Even philosophy of mind has begun to explore a Bayesian perspective

10. The even newer landscape

Some aspects of the rationalist programme have been making a bit of a comeback

- Of course, in some sub-disciplines they never completely disappeared

One reason for this is very concrete:

- The volume of language data required to train really good e.g. n-gram models is enormous
- And out of reach for all but a handful of languages
  - There just isn't enough Portuguese or Swahili
  - To say nothing of Telugu or Inupiak

So noisy channel architecture systems will struggle to handle most of the world's languages

There's another area where data scarcity is relevant too

- That is, human language acquisition
- Children's exposure to language is measured in small numbers of millions of words
- Not enough to provide an accurate language model

11. Reasons to love trees

Parse trees, that is

They're a means to at least three distinct ends:

- A satisfactory explanatory account of the nature of human language
- A useful scaffolding for semantics
- A remedy for the sparse data problem

12. The nature of human language

Two related perspectives:

Artefactual
: The original structuralist question
  - What is the nature of a particular human language?
  - Considered as a set of utterances/sentences

  Or, what is distinctive about human languages in general?

Psychological
: The linguistic component of cognitive psychology
  - What underlies mature human linguistic performance?
  - How does that arise throughout child development?

13. From structuralism to generative grammar

Building on, and extending, formal language theory, Chomsky (re)defined the scientific study of language

Warning! His rationalist rhetoric ("what do people know about their language") and his technical vocabulary ("generative grammar") encourages a misunderstanding

- That his goal is a model of human language processing

But his actual goal is to characterise *what* people know about (their) language

- Not *how* they know about language
- Or what processes are involved in human *use* of language

Generative grammar is 'generative' because it defines a language as the set of strings *generated* by a specified formal procedure

- Not because it describes the process by which *people* generate speech or text

14. The Chomsky hierarchy, again

Chomsky identified four classes of (formal) languages

- Together with 4 classes of mechanism capable of generating them
- And four variants on rewriting rules (more on this later) ditto

type 3
: regular languages; finite-state automata; regular grammars

type 2
: context-free languages; pushdown automata; context-free grammars

type 1
: context-sensitive languages; linear bounded automata; context-sensitive grammars

type 0
: recursively-enumerable languages; Turing machines; generalised rewriting systems

Each mechanism and grammar type is capable of generating any language of its type, or a higher type

- But *not* a lower type

The current pretty broad consensus is that natural languages are somewhere between type 2 and type 1

- Closer to 2 than 1, insofar as that makes sense

15. Rewriting systems

The grammar types in the Chomsky hierarchy can all expressed as constraints on a formalisation of the idea of **rewriting systems**

A rewriting system consists of

Terminals
: or **terminal symbols**: words (for now)

Non-terminals
: or **non-terminal symbols**: Names for constituents in a language

Rules
: or **productions**, each of which is a pair of

  a left-hand side
  : a non-empty sequence of any number of terminals and non-terminals

  a right-hand side
  : a (possibly empty) sequence of any number of terminals and non-terminals

Distinguished symbol
: One of the non-terminals
  - The starting point for all analyses
  - Usually **S**

16. Rewriting, cont'd

Regular and **Context-free** grammars (**CFG**s) restrict the left-hand side of rules to a single non-terminal symbol

Regular grammars further restrict the *right*-hand side to be a pair of a non-terminal and a terminal, or a single terminal

- All the pairs have to be in the same direction:

  right-linear
  : Every pair is **t**,**NT**

  left-linear
  : Every pair is **NT**,**t**

For example, the following (right-linear) grammar defines the language a^*b^*:

```
S → aS
S → B
B → bB
B → b
```

This should look familiar from lecture 2:

- Regular grammars are equivalent to Finite State machines

And the following context-free grammar defines a^nb^n:

```
S → a S b
S →
```

Note the common convention that non-terminals start with upper-case letters, terminal with lower-case

17. Reasons to love trees 1: Defining a language

What is the language defined by a mechanism?

For FSAs, a string is in the language defined by a finite-state automaton **F** iff there is a non-empty sequence **s*** of states of **F** such that

- The first state in **s*** is a start state

- The last state in **s*** is a terminal state
- Either
    - **s*** is of length one and the string is empty
        - That is, **s*** consists entirely of a state which is both a start and an end state
    - **s*** is of length two or more and there is at least one transition between every adjacent pair of states in **s*** with a label such that the concatenation of all those labels is the string

18. Define a language with a grammar

There are two (main) ways to interpret a grammar as defining a language

- The details vary slightly depending on the type
- What follows is for CFGs

A string is in the language defined by a context-free grammar **G** iff

Rewriting
You can get to the string by

- writing down **G**'s distinguished symbol
- writing down a new line by choosing a non-terminal from the line above, choosing a rule from **G** with that symbol as its left-hand side, then re-writing the line above with the chosen symbol replaced by the right-hand side of the chosen rule
- repeating this until there are no non-terminals left
- The resulting tableau is called a **derivation**

19. CFG interpretation, cont'd

Node admissability
A string is in a language defined by a grammar **G** iff there is at least one labelled tree such that

- The leaf nodes of the tree, in order, correspond to the string
- The root of the tree is labelled with **G**'s distinguished symbol
- For every *non*-leaf node, there is a rule in **G**
    - whose left-hand side is the label of the node
    - whose right-hand side corresponds to the labels of the node's children, in order

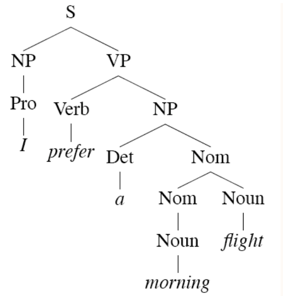
20. Trees as proofs

Under *either* interpretation, we can use a **parse tree** (strictly speaking an *ordered* tree) to illustrate the way in which a string belongs to the language defined by a CFG

For example, using the grammar and lexicon for the ATIS domain given in section 10.2 of Jurafsky & Martin (3rd edition)

- For the sentence
- I prefer a morning flight

The parse tree that 'proves' that this is in the language is



21. Generativity

As with FSAs and FSTs (see Lectures 2,3), you can view rewrite rules as either analysis or synthesis machines

- Generate strings in the language
- Reject strings not in the language (recognition)
- Show *how* strings are in the language (parsing)

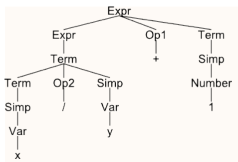
22. Reasons to love trees 2: Scaffolding for semantics

Compilers and interpreters for programming languages use parse trees to implement a **compositional** semantics

Consider a simple grammar for Python arithmetic expressions:

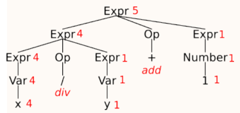
```
Expr → Expr Op1 Term | Term
Term → Term Op2 Simp | Simp
Simp → Var | Number | '(' Expr ')'
Op1 → '+' | '-'
Op2 → '*' | '/'
```

Assuming some more work to **tokenise** the input, this will give us an analysis for the string "`x/y+1`"



We can compute a value for this expression using the tree

- If we associate the appropriate computation with each production
    - (skipping a lot of details here, some of which will be covered in a few weeks)
    - And assuming that `x` is `4` and `y` is `1`



We call that kind of semantic computation a **compositional** one

> "The meaning of the whole [constituent] is a function of the meaning of the parts [children]"

Logic-based approaches to natural language semantics often adopt this approach as well

23. Reasons to love trees 3: tackling the sparse data problem

In a number of different areas

- including machine translation, question-answering and text-to-speech synthesis

attempts are being made to generalise

- *from* languages for which we *do* have lots of data
- *to* languages where we don't

By abstracting linguistic structures (trees and/or dependency graphs) to a sufficient extent

We won't get to this in this course.