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, metamodell
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 bastardizing its hyphen for a licence 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 practitioners.
• 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

5. The empiricist strikes back
Starting in the late 1970s, in the research community centered 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

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 a series of digital corpus of the dead street journal in 1994 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 1960s and 1970s 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 psychology and the social sciences had diluted theoretical and computational theory suffered a, perhaps connected, loss of energy and sense of progress. Of course, in some sub-disciplines they never completely disappeared. The starting point for all analyses is the raw string of symbols (often called the string or the input). What underlies mature human linguistic performance, and what processes are involved in human language acquisition?

10. The even newer landscape

Some aspects of the relativist programme have been making a bit of a comeback. Of course, in some rule-disciplines they never completely disappeared.

One reason for this is very concrete:
- The nature of language required to explain word order in Romance languages (or maybe just the right-hand side to be in the same direction):
  - Right-hand side: start state is NT 1
  - Left-hand side: start state is NT 1

For example, the following (right-hand) grammar defines the language $s^*$:

11. Reasons to love trees

There's another area where data scarcity is relevant too:
- What is the nature of a particular human language?
- Are the processes are involved in human language acquisition? What is distinctive about human languages in general?
- Are noisy channel architecture systems will struggle to handle most of the world's languages?
- What underlies mature human linguistic performance, and what processes are involved in human language acquisition?

12. The nature of human language

See the two basic perspectives:
- Artificial
- Psychological

The original structuralist question: What is the nature of a particular human language? What underlies mature human linguistic performance?

Or, what is distinctive about human languages in general?

What is the nature of a particular human language?

The current pretty broad consensus is that natural languages are somewhere between type 2 and 4.

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 of his actual goal.

- He has a model of human language processing.
- Not what he wanted to characterise what people know about their language.
- Not whether they know about their 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 human procedure.

But 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:
- Type 0: Regular languages, finite-state automata, regular grammars.
- Type 1: Context-free languages, pushdown automata, context-free grammars.
- Type 2: Context-sensitive languages, language level automata, context-sensitive grammars.
- Type 3: Linearly enumerable languages, Turing machines, generated rewriting systems.

Each mechanism or 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 3.

15. Rewriting systems

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

A rewriting system consists of:

- Terminals
- Non-terminals

Non-terminals: Names for constituents in a language

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

- A left-hand side: non-empty sequence of any number of terminals and non-terminals.
- A right-hand side: literally empty sequence of any number of terminals and non-terminals.

The only symbols:
- Universally:
  - $\Rightarrow$
  - $\Rightarrow^*$
  - $\Rightarrow^+$

Distinguished:
- The starting rule for all analyses:
  - $S \rightarrow s^*$

16. Rewriting, cont’d

Regular or context-free grammars (CFGs) 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.

A CFG is described by a 5-tuple:

- $S \rightarrow a B$ or $S \rightarrow a$

For example, the following (right-hand) grammar defines the language $s^*$:

This should be familiar from Saxon’s rule.

- Regular grammars are equivalent to finite state machines.

And the following context-free grammar defines $s^*$:

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 FSA, a string is in the language defined by a finite-state automaton $A$ if there is a non-empty sequence $s^*$ of states $q_i$ such that:
- The first state is $A$’s 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.
- Either $s^*$ is of length two or more and there is at least one transition between every adjacent pair of states in $s^*$, with at least such that the concatenation of all these 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 $G$ if and only if you can get to the string by:

1. Writing down $G$’s distinguished symbol
2. 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
3. Repeating this until there are no non-terminals left
4. The resulting tableau is called a derivation.

19. CFG interpretation, cont’d

Node admissability:

- Parsing is a language defined by a grammar $G$ if there is at least one labelled tree such that:
  1. The leaf nodes of the tree, in order, correspond to the string.
  2. The root of the tree is labelled with $G$’s distinguished symbol.
  3. For every non-leaf node $n$ in the tree:
     - whose left-hand-child is the label of the node
     - whose right-hand-child is the label 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:

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 $x = 4$ and $y = 1$
- We call that kind of semantic computation a compositional one.

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 $x = 4$ and $y = 1$
- We call that kind of semantic computation a compositional one.

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 have been made to generalize

- 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.