Day 8: Intro to Natural Language Processing

- Reasons why you'd want a computer to understand English. Many of these turned out to be bad ideas or otherwise problematic:
  - Voice Control (Alexa, etc.)
  - Write programs easily
  - Automatic Translation
  - Machines can pick news for you (not actually a good idea)
- NLP is required for passing the Turing Test
  - Lots of tricks to do this:
    - Ex: ELIZA, which just used a long list of patterns to convincingly pretend to be a therapist
      - Shockingly effective
- Why isn't pattern matching a good NLP solution?
  - You can get 90% of the way there, but the last 10% requires very advanced AI
  - Also, this isn't actually understanding
  - Doesn't scale well (lots of patterns, collisions, etc.)
- Machine Translation
  - Big Cold War Project
  - The idea was to use a Russian/English dictionary and syntactic rules to reorder the words
  - Works at small scales, but not in general. Ex:
    - The box is in the pen refers to a pig pen
    - The pen is in the box refers to a writing impliment

- Natural Language DB Interface
  - Allows non-technical executives to interact with a database using a very limited langauge
  - Kinda worked, but learning SQL was easier
- To actually talk to a computer, you need Speech Recognition → NLP → Text to Speech

Linguistics of NLP

- Morphological: syllables, pre/suffixes, inflections, etc.
- Lexical: word meanings and word categories
- Syntax: structure of the language
- Semantics: meaning of the language
- Discourse: links multiple utterances together

Key NLP Components

- Morphological Analysis: pre/suffix segmentation and punctuation segmentation
- Lexical Analysis: categorization and meaning of words
- Syntactic Analysis: linear sequence of words to more complex structure
- Semantic Analysis: syntactic analysis to stored knowledge representation
- Discourse Integration: process and track context across multiple sentences
- Pragmatics: resolve speech and understand what to do

- Syntactic Parsing: at the hard of NLP
  - Many historic issues:
    - Algorithmic Efficiency
    - Is this plausibly the way humans do it (psychological plausibility)
    - Top down or bottom up?

        - Integrating with other forms of knowledge?
        - Does the parser give you all possible parsings or just the best one?
    - Parsing is very interdisciplinary:
        - Linguistics
        - Theoretical Computer Science
        - Computational Linguistics (focused on parsing)
        - AI/NLP: how to use knowledge
    - Grammar: the start of syntax parsing
        - Means the overall knowledge of a language (don't really understand?)
        - Chomsky's Universal Grammar: all possible constraints for a given language
        - Phase Structure Grammar: Main element of syntactic parsing, accounts for hierarchical/sequential structure
    - Autonomy of Syntax: grammatical correctness can be determined independently from logical correctness.
- Generative Systems:
    - Assuming that there are infinite sentences in a language, but that cognition is finite, there must be a finite system of grammar than can generate an infinite number of meaningful utterances
    - Groundbreaking paper: Chomsky's 1957 *Three Models for Language Description*. He showed three types of systems, proving that the first two were inadequate for human language:
        - Finite State Machines
        - Phrase Structure rammars
        - Transformational Grammars
- The Chomsky Hierarchy: Languages form a hierarchy of types/computation models
    - Regular Languages/Finite State Machines: `aaaaaaab`, `abababab`

- Σ is a finite set of all possible tokens in a language. $\Sigma*$ is the infinite set of all possible combinations of tokens. A language is a subset of $\Sigma*$.
- FSMs consist of a graph, with each node being a state and each edge being a token from Σ. Valid strings are strings that can start at a start node and end at and end node, while only taking edges corresponding to the current token.
- Doesn't scale well
- Insufficient for English, because you can insert arbitrary thing between connected words, ex: `Sarah called [the person [she was meeting [for lunch [at the mall]]]] up`.

- Phrase Structure/Context-Free/Push Down Automata: `ab`, `aabb`, `aaabbb`, `aaaabbbb`
  - Σ is still finite set of all possible tokens in a language. $\Sigma*$ is still the infinite set of all possible combinations of tokens. A language is a subset of $\Sigma*$.
  - A set of "non-terminals" N
  - Rules in the form of $N \to N \times \Sigma$
  - A starting symbol s which is a non-terminal. The s is generally always the same, and can expand to any valid sentence.
  - A string (of terminals) is in the language if you can derive it from s according to the rules.
  - Still insufficient for English though:
    - Can't handle conjunctions (`John likes turkey and Bill chicken`)
    - Crossed serial dependencies (`John and Bill like turkey and chicken, respectively`)
    - Duplication phenomena (`Corporate interest rates? Schmorporate interest rates!`)
- Context-Sensitive/Linear Bounded Automata: `he shot himself` (coordinated `he` and `himself`), `aaaccccbbb`

        - Recursively Enumerable Languages/Turing Machines: strings only of prime length
- You might be asked to draw a parse tree on the midterm
- Ambiguity is a big issue, and is a fundamental limit of language
- Three main operations with a grammar:
    - Generate random/all sequences in a language
        - All will take a very long time or possibly forever
    - Recognize: determine if a sequence of symbols is a valid sentence in the language, returns true or false
    - Parse: determine one/some/all valid parse trees for a sequence of symbols
        - Depth-first search often used here (top-down parsing)
            - Eventually works, but needs a lot of pruning
        - Alternative: Chart Parser (bottom up)
            - Runs in $O(n^3)$ time, very effective in practice
            - Sometimes generates too many trees to chose the best one
            - Norvig proved that memoization makes a top-down parser equally efficient
            - A given cell is the lexical categories created by words row index through column index:

0	NP NOM JOHN	S	S			S
	1	VP V SAW	VP			VP
		2	NP NOM MARY			NP
			3	PREP WITH		PP
				4	DET A	NP
					5	N BALL
						6