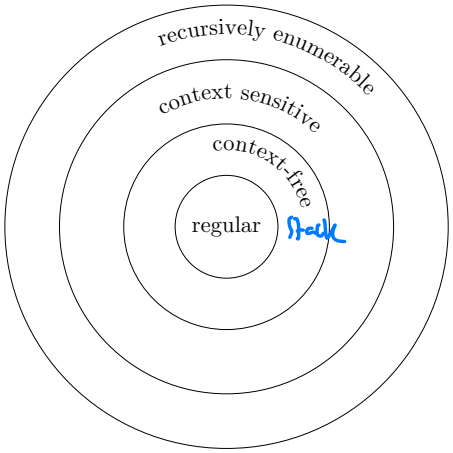
Summary and overview

1 What we learned, and what we know about natural languages?

1.1 Summarizing this class...

1. Recursion is everywhere, including natural numbers (`Nat`), propositional formulas (`Form`), strings, trees ...

2. What amount of computational resources do we need to compute natural languages?



Chomsky Hierarchy: regular ⊆ context-free ⊆ context-sensitive ⊆ unrestricted

Where do natural languages sit?

Phonotactics: subregular, maybe Tier-based strictly local
Syntax: mildly context sensitive
...
Morphology: regular but with ww, which is even beyond context-free

3. Grammars are finite means to infinite use!
   For formal (string) grammars,

Grammar	Language	Automaton
Type-0	Unrestricted/recursively enumerable	Turing machine
Type-1	Context-sensitive	Linear-bounded automaton
Type-2 (context-free rewrite rules)	Context-free	Nondeterministic pushdown automaton
Type-3	Regular	Finite-state automaton

\+ regular expressions for regular languages

\+ Strictly local grammars for strictly local languages

For formal tree grammars,
\+ (bottom-up) finite-state tree grammars which yields context-free languages if we linearize the terminals in a tree from left to right; but it expresses something more about the dependencies internally.

4. In this class, we ask the following essential questions:

   (a) For any given formal grammar, how do we know whether an expression (either string or tree) is generated or not?

   > Usually this question can be approached recursively:
   >
   > - For FSAs: forward, backward
   > - For CFGs: inside, outside
   > - For FSTAs: under, over [a]
   >
   > [a] We didn't talk about over values because it relies on the notion of a tree context. The definition is very similar to outside values!

   (b) We extended this to other questions $f_M(u)$ about a string in the context of a given grammar.

   > - what is the highest weight a real-valued grammar can assign to a string? (max, multiplication)
   > - what is the total probability a probabilistic grammar can assign to a string? (sum, multiplication)
   > - what is the cost a cost grammar can assign to a string? (min, sum)
   > - what are the output strings a transducer can output given an input string? (union, lifted concatenation)

   In answering these questions, we see that there are algebraic commonalities between the two operations we use: the underlying data structure and the underlying function are the SAME as the boolean case.

   So, we abstracted away and proposed a generic version of the **semiring-valued formal grammar**, and the generic version of the semiring-valued recursive functions.

   - We looked at FSAs as a case study, but this applies to all kinds of formal grammars!

   (c) For each language class, what are their mathematical properties?

   > - Regular languages are closed under union, concatenation, and the star operation.
   > - Strictly local languages are closed under suffix substitution.
   > - We can ask the same questions for context-free languages, and all other language classes!

(d) For each language class, how do we know whether a language is in this class? [1]

> Exchangeable subexpressions!
>
> - Regular languages: Myhil-Nerode theorem; finitely many of buckets of prefixes
>
> What about other language classes?

infinite - not regular

(e) How do we test the psychological plausibility of formal grammar?

> Assumptions you need to make:
>
> What kind of grammar is natural language?
>
> Then, you ask,
>
> - What are possible parsing schema?
> - What is the kind of parser[a] used in natural language processing/comprehension?
>
> [a] A parser is defined as the system that given a grammar, the algorithm that can output the analyses of a string.

Assuming CFGs are the grammar used in natural language syntax, we discussed three different possible parsers: bottom-up, top-down, and (arc-eager) left-corner.

Homework 3

Based on the different levels of processing difficulty for branching structures, it seems human parse in a left-corner fashion.

not difficult to parse left-corners

center is difficult

1.2 But the story is not complete yet...

2 classes 900

1. We see in the last class that natural language syntax is beyond context-free, but also should be more restrictive than context-sensitive (because we do not observe $\{a^i \mid i \text{ is a prime number}\}$ in natural languages.)

   Then, what is the grammar formalism that computes this language class?

   - a lot of candidates: multiple context-free grammars (MCFGs), tree adjoining grammars (TAGs), combinatory categorial grammar (CCGs), minimalist grammars (MGs)...

   - What are their mathematical properties?
   - What are some possible parsers?
   - How does a left-corner parser look like in these grammars?

2. Does formal complexity reflect the processing difficulty, and/or learning difficulty of natural language patterns? If not, what does this piece of information tell us?

[1] You can also use the so-called pumping lemma, and usually, a formal language theory class (CS181) will teach you this approach.

Some puzzles

- Formally speaking, $a^n b^n$ or center embedding structures or nesting dependencies are context-free; $a^i b^j c^i d^j$ or cross-serial dependencies are mildly context sensitive.

  But in terms of processing, Dutch (cross-serial dependencies) is easier than German (center-embedding dependencies). [2]

- Formally speaking, total reduplication $\{ww \mid w \in \Sigma^*\}$ is even beyond context-free and requires mildly context-sensitive grammars.

  But empirically, total reduplication is a widely attested phenomenon in morphophonology and easy to learn. Morphophonological patterns lack nesting dependencies.
  Indonesian pluralization: *buku* 'book' ∼ *bukubuku* 'books'

3. If we know what formal grammar is in terms of its structure, how do children ever converge to a specific grammar with no difficulty? What is the learning process?

2 Some loose-ends

2.1 How does everything connect to classical NLP tasks?

1. Statistical learning applies to the probabilistic version of the formal grammar: maximum likelihood estimation applied to estimate the parameters.

   - ... for probabilistic FSAs (or Hidden Markov Models): forward-backward algorithm
   - ... for probabilistic CFGs: inside-outside algorithm

2. HMMs are often used for part of speech tagging (the tag of one word might depend on the tags of adjacent words), speech recognition, speech synthesis, (even bioinformatics data, because they are sequential too!)

   Inside-outside was first introduced as an application of speech recognition and has been widely used for probabilistic parsing.

2.2 How does everything connect to modern NLP advancements?

1. A recent trend of research aims to connect formal grammars with neural networks in order to better understand what the "black box" of ML/AI models is actually learning. Many different approaches have been taken in pursuit of this goal. [3]

   - using rigorous math (spectral learning) to show equivalence/the connection (theoretically speaking)
     - Rabusseau et al (2019) bridges between Weighted Finited Automata and second-order recurrent neural networks with linear activation functions; they are expressively equivalent!
     - Chen et al (2018) Turing-complete-ness and undecidability of (RELU-activated) RNNs .

[2] Bach, E., Brown, C., & Marslen-Wilson, W. (1986). Crossed and nested dependencies in German and Dutch: A psycholinguistic study. *Language and Cognitive Processes*, 1(4), 249-262.
[3] For a more comprehensive overview, see the tutorial here: https://rycolab.io/classes/esslli-23/

  - Perez et al (2021): transformers with architectures based on *attention* are Turing complete.
  - Hao et al (2022) studies the capacities of different attention formalisms.
- using empirical experiments (really just simulations, no human subjects involved) to see what languages ($a^n b^n$, ww^R, ww...) can be learned by different kinds of neural networks
  - Delétang et al (2022)

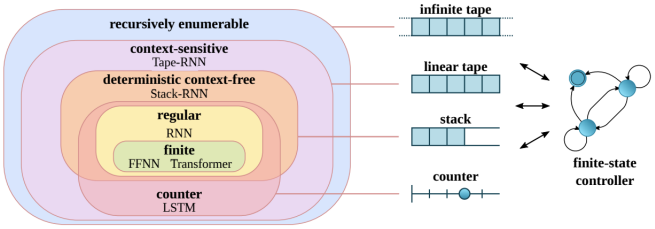


Figure 1: Formal language classes and their correspondence with neural network architectures. *Left*: Our empirical evaluation locates the architectures on the hierarchy of formal language classes. *Right*: Each formal language class is associated with a minimal computational model (automaton) to recognize or generate the language (see Section 3). All automata have a finite-state controller at their core, in addition to increasingly restrictive memory access as we descend the hierarchy.

  - Wang (2023) studied the capacities of RNN Seq2Seq's capacities by training the neural net with reversals, total identity, copying and quadratic copying.
- enriching the structure of neural networks with memory akin to those used in classical formal grammars (stack, tape) for more interpretable neural networks.
  - Hao et al (2018) extended RNNs with stack memory and discovered that the training process disprefers interpretable classic stack-based algorithms.

2. Debates about large language models

- There is a recent debate about what LLMs really tell us about linguistics and human cognition. If this is something that you are curious about:
  (a) Piantadosi (2023). Modern language models refute Chomsky's approach to language
  (b) Katzir (2023). Why large language models are poor theories of human linguistic cognition. A reply to Piantadosi (2023).
  (c) Rawski & Baumont (2023). Modern Language Models Refute Nothing
  (d) Kodner, Payne & Heinz (2023). Why Linguistics Will Thrive in the 21st Century: A Reply to Piantadosi (2023)
- One thing that I would like to mention is if you ever wonder what the monstrous language models with millions, billions and trillions parameters actually learn, (computational) linguistics is an important and useful tool.
  You can ask the following questions, for example....
  - Do language models learn island (in-)sensitivity? What about exceptions?
  - Do language models behave like a human when it comes to different embeddings of branching structures?
  - Do language models really process sentences in a human way?
  - Do language models learn an infinite language or just a finite approximation?

If you go to the recent proceedings of the society for computation in linguistics, you will see a lot of recent attempts along this line, and there are some works cited in the papers mentioned above.

3. Compositionality and recursion are some key tests(?)
Dziri et al (2023) trained chatgpt and gpt4 (transformer-based LLMs) with some questions that use dynamic programming and recursion (`Nat` multiplication is one of them). They found that these models have limitations in compositional reasoning with increased complexity in tasks (hypothesized to be due to error propagation).

3 Concluding remarks

There are a lot of questions to be answered, both in NLP (engineering side) and in computational linguistics (theoretical side). These questions help us understand what intelligence really means in the context of human intelligence, and are essential to build bridges between...

investigating languages and cognition by viewing them as computation

and

designing more organic, transparent, interpretable AI systems