Towards a Learning Theory of Grammars

Dan Mitropolsky, Postdoctoral Fellow (MIT)
PhD (Columbia), Joining Tufts as Asst. Professor in CS next Fall

Joint work with Laura Ying Schulz (ETH, MIT), Tomaso Poggio (MIT)

For Statistical Learning Theory, Fall 2025

Motivation: Understanding how LLMs learn

- LLMs seem to have mastered language
- A *new kind* of linguistic intelligence
- **Major question:** how is language processed and learned in *humans?*
- **New question:** how is language processed and learned in *LLMs?*

Motivation: Understanding how LLMs learn

- **Major question:** how is language processed and learned in *humans?*
- **New question:** how is language processed and learned in *LLMs?*

- On the **processing** side, there is work analyzing LLMs at inference
  - e.g. **probing work** — which parts of the network are sensitive to what features in language, by what depth are grammatical relations computed, etc.
- On the **learning** side, most work focused on improving training methods
- **Very little about the *training dynamics of LLMs on language!!!***

Overview of Today's Talk

- Natural languages (English etc.) are **context-free grammars,** a kind of grammar with structured properties
- Context-free grammars = a "structured hypothesis class" like **polynomials, XOR functions,** etc.
  - There exist theories of how these *other* hypothesis classes are learned
- **Theoretical investigation** - what can we say about learning vs. the **substructure** of CFG?
- **Empirical investigation** - study how **small language models** learn small CFGs where we control everything (e.g. "amount of recursion")

Preliminaries

Context-Free Grammars

- Set of rules for generating hierarchical language
- Special **start symbol S**
- Set of **non-terminals A, B, C, … (including S)**
- Set of terminals (“tokens” of the language)
- Set of rules **mapping non-terminals to strings of non-terminals and terminals**

Context-Free Grammars

- Set of rules for generating hierarchical language
- Special **start symbol S**
- Set of **non-terminals A, B, C, … (including S)**
- Set of terminals (“tokens” of the language)
- Set of rules **mapping non-terminals to terminals**

Example

$$S \rightarrow S\ S$$
$$S \rightarrow (\ S\ )$$
$$S \rightarrow \epsilon$$

ϵ represents empty string

Non-terminals: S

Terminals: $($, $)$

What language does this CFG generate?

Context-Free Grammars

- Set of rules for generating hierarchical language
- Special **start symbol S**
- Set of **non-terminals A, B, C, … (including S)**
- Set of terminals (“tokens” of the language)
- Set of rules **mapping non-terminals to terminals**

Example

$$S \rightarrow S\ S$$
$$S \rightarrow (\ S\ )$$
$$S \rightarrow \epsilon$$

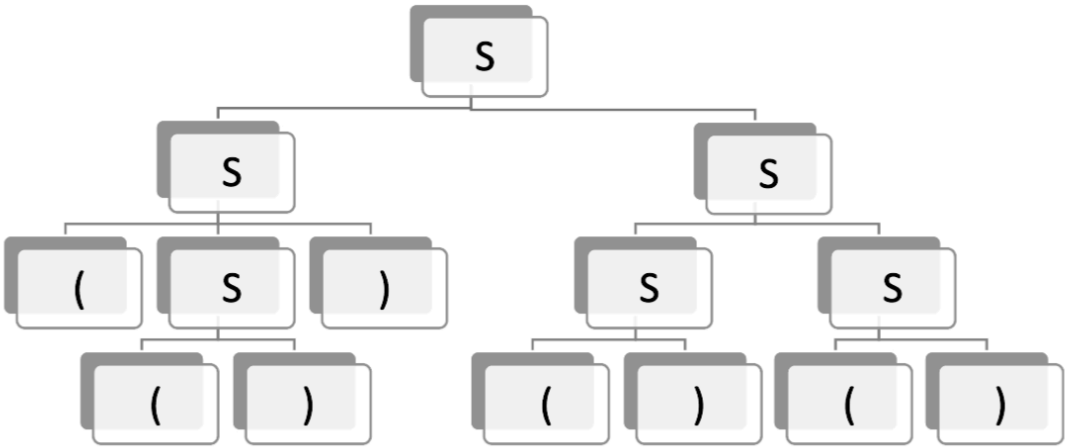
ϵ represents empty string

Non-terminals: S

Terminals: $(,\ )$

THE LANGUAGE OF ALL STRINGS OF BALANCED PARANTHESES

Example from our CFG



(( )) ( ) ( )

What languages are CFGs?

- All programming languages

```
S -> if COND then DO else DO
COND -> x, COND -> y
COND -> ( COND and COND )
COND ->  ( COND or COND )
DO -> return
DO -> S
x, y -> …
```

- Mathematical / arithmetic languages

What languages are CFGs?

- Natural languages!

```
S -> NP  VP
NP -> D  N
D -> the  |  a
N -> cat  |  dog  | robot  | hamster | …
VP -> V  Adv
V -> runs  |  eats  | sleeps  | …
Adv  -> happily  | …
```

Do LMs learn CFGs?

- Issue: a CFG defines a subset of all strings, *no notion of distribution*
- Enter **Probabilistic Context-Free Grammars**
- PCFG is a CFG, but for each non-terminal **X:**
  - Each rule with **X** on left-hand side assigned **probability**
  - Probabilities of the rules sum to 1:
    - $$\sum_{R = X \rightarrow \cdots} \Pr(R) = 1$$

Do LMs learn CFGs?

- Issue: a CFG defines a subset of all strings, *no notion of distribution*
- Enter **Probabilistic Context-Free Grammars**
- PCFG is a CFG, but for each non-terminal **X:**
  - Each rule with **X** on left-hand side assigned **probability**
  - Probabilities of the rules sum to 1:
    - $$\sum_{R = X \rightarrow \cdots} \Pr(R) = 1$$

Example

$S \rightarrow S\ S$ [0.3]

$S \rightarrow (\ S\ )$ [0.3]

$S \rightarrow \epsilon$ [0.4]

What is language modeling?

- Have a distribution *P* over all possible sentences {NON-TERMINALS}*
- Language modeling = **Maximum Likelihood Estimation (MLE)**
- For dataset of sentences $\{s_1, \ldots, s_N\}$ find parameters $\hat{\theta}$ s.t.
- $\hat{\theta} = \arg\max_\theta \mathscr{L}_N(\theta)$
- Where $\mathscr{L}_N(\theta) = Q_\theta(s_1) \cdot Q_\theta(s_2) \cdots Q_\theta(s_N)$

What is language modeling?

- $\hat{\theta} = \arg\max_\theta \mathscr{L}_N(\theta)$
- Where $\mathscr{L}_N(\theta) = Q_\theta(s_1) \cdot Q_\theta(s_2) \cdots Q_\theta(s_N)$

  By **monotonicity**, can redefine $\mathscr{L}_N(\theta) = \frac{1}{N} \sum_{i=1}^{N} \log(Q_\theta(s_i))$

  Taking $N \to \infty$, $\mathscr{L}_N(\theta) \to \mathscr{L}(\theta) := \mathbb{E}_{s \sim P} [\log Q_\theta(s)]$

What is language modeling?

- **Taking** $N \to \infty$**,** $\mathscr{L}_N(\theta) \;\to\; \mathscr{L}(\theta) := \mathbb{E}_{s\sim P}\,[\log Q_\theta(s)]$

- That is, MLE does empirical maximization of likelihood of true distribution
- Claim: **The above expectation would be maximized by P itself**
  - **What would** $\mathscr{L}(P)$ **equal?**

What is language modeling?

- **Taking** $N \to \infty$**,** $\mathscr{L}_N(\theta) \;\to\; \mathscr{L}(\theta) := \mathbb{E}_{s\sim P}\,[\log Q_\theta(s)]$

- That is, MLE does empirical maximization of likelihood of true distribution
- Claim: **The above expectation would be maximized by P itself**
  - **What would** $\mathscr{L}(P)$ **equal?** $\sum_s P(s)\;\log(P(s))$ **Entropy of P!**

Kullback-Leibler Divergence

- Given distributions P and Q overΣ^* the **Kullback-Leibler (KL) Divergence of Q from P =**
- $$\mathrm{KL}(P \parallel Q) = \sum_{s \in \Sigma^*} P(s) \log \frac{P(s)}{Q(s)}$$
- **Fundamental measure of distance between two distributions**

Alternative View of MLE / Language Modeling

- Proposition
  MLE **same as minimizing KL Divergence between true P and** Q_θ
- Theorem
  $\mathscr{L}(\theta) = -\,\mathrm{KL}(P \parallel Q_\theta) + H(P)$

  where $H(P) = \mathbb{E}_{s\sim P}\,[\log(P(s)]$ is the **entropy** of P

- Theorem implies Prop: $H(P)$ is independent of θ, so max'ing $\mathscr{L}(\theta)$ same as minimizing $\mathrm{KL}(P \parallel Q_\theta)$

Proof

- Proof

$$-\mathcal{L}(\theta) = \sum_{x \in \Sigma^*} P(x)(\log Q_\theta(x))$$

$$= \sum_{x \in \Sigma^*} P(x)(\log P(x) - \log P(x) + \log Q_\theta(x))$$

$$= -\sum_{x \in \Sigma^*} P(x) \log \frac{P(x)}{Q_\theta(x)} + \sum_{x \in \Sigma^*} P(x) \log P(x)$$

$$= -\,\mathrm{KL}(P \parallel Q_\theta) + H(P)$$

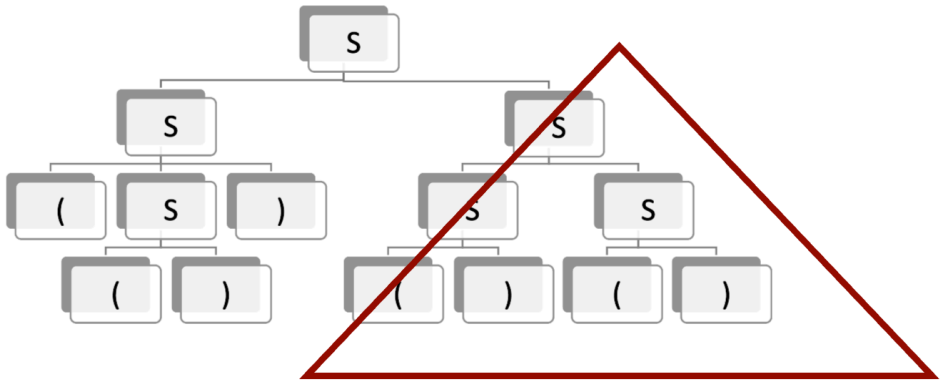
The “Algebra” of CFGs

Back to Context-Free Grammars…

- Other areas that study specific hypothesis classes look at **substructures:**
  - Eg polynomials: monomials that compose them
  - XOR functions: XOR's of smaller XOR functions
- **What is the substructure of CFGs?**
- Intuitively: sub-"grammars"?

Subgrammars

- Intuitively, should correspond to sub-trees:



- Turns out to be quite subtle!

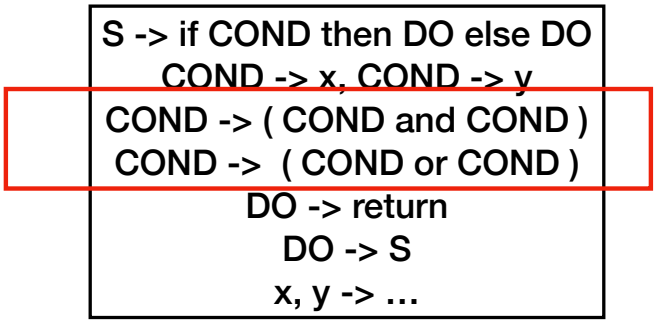
Subgrammars: inner

- **Definition** An **inner subgrammar G'** of **G**
  - Choose non-terminal X of **G** to be start symbol of **G'**
  - **G' =** *closure* of X (all rules X → ⋯, all rules from those non-terminals, etc)
  - Probabilities (for PCFG): those of **G**

  - Observe: strings of **G'** are **sub**strings of **G**
  - Intuition: captures **compositional substructure** of CFGs

Inner subgrammar example

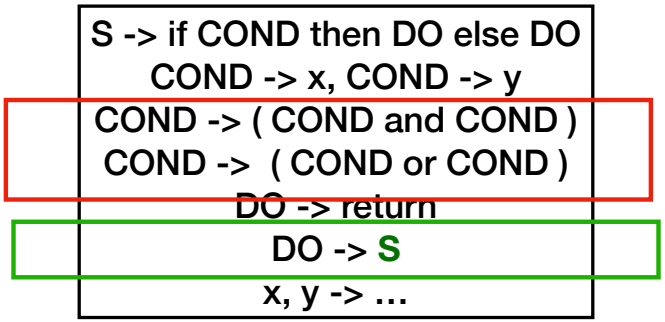
- Our programming language G



- COND as starting symbol forms a subgrammar
- What about from DO?

Inner subgrammar example

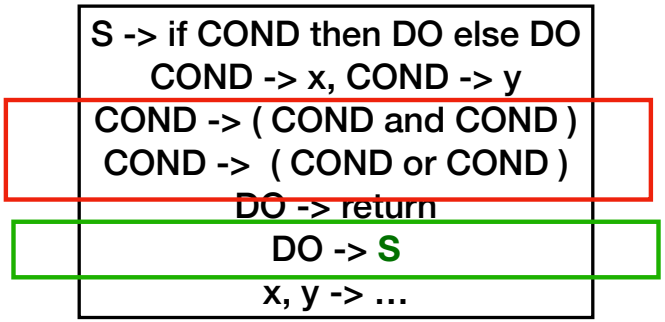
- Our programming language G



- COND as starting symbol forms a subgrammar
- What about from DO?
  - DO generates S, so subgrammar from DO = all of the G

Inner subgrammar example

- Our programming language G



Proper subgrammar

(not all of G)

- COND as starting symbol forms a subgrammar
- What about from DO?
  - DO generates S, so subgrammar from DO = all of the G

Decomposition of CFG into inner sub grammars

- **Formally, why inner sub grammars capture "composition/hierarchical structure" of CFGs:**

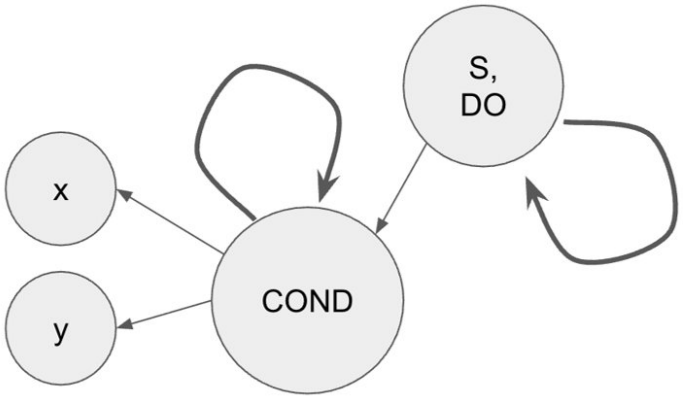
- **Theorem**
  Every (P) CFG can be uniquely decomposed into a hierarchy of its inner sub grammars.

  This hierarchical structure can be represented as a directed acyclic graph (DAG) with self-loops.

A first hypothesis about LM learning

- Each node labelled with non-nonterminal(s) generating that subgrammar
- Self-loop = subgrammar can generate itself

S -> if COND then DO else DO
COND -> x, COND -> y
COND -> (COND and COND)
COND -> (COND or COND)
DO -> return
DO -> S
x, y -> …



Subgrammars: outer

- **Definition** An outer **subgrammar G'** of **G**
  - Choose at least one rule $S \rightarrow \cdots$ of **G**
  - For each non-terminal X on RHS, choose at least one rule $X \rightarrow \cdots$ of G, etc.
  - Probabilities of G': **renormalized probabilities of G (per non-terminal)**

  - Observe: all strings generated by G' are strings of G (*not substrings*)
  - Captures notion of *reduced / simplified version of the grammar*

Recap!

- Language modeling is MLE over sentences, **same as minimizing KL-divergence between true language distribution and model's**
- PCFGs, a kind of language, have substructures:
  - Inner sub grammars (compositional substructure, "subtrees")
  - Outer sub grammars (simplified version of the grammar)
- **We can finally study the connections between the two!**

Recap!

- Language modeling is MLE over sentences, **same as minimizing KL-divergence between true language distribution and model's**
- PCFGs, a kind of language, have substructures:
  - Inner sub grammars (compositional substructure, "subtrees")
  - Outer sub grammars (simplified version of the grammar)
- **We can finally study the connections between the two!**
- **Conjecture: Children learn simpler language first** (e.g. outer subgrammars), LLMs will too?

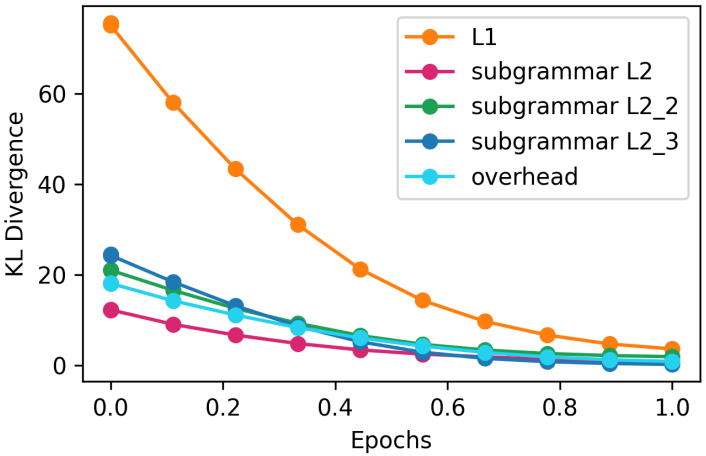
Language Modeling vis-a-vis Subgrammars

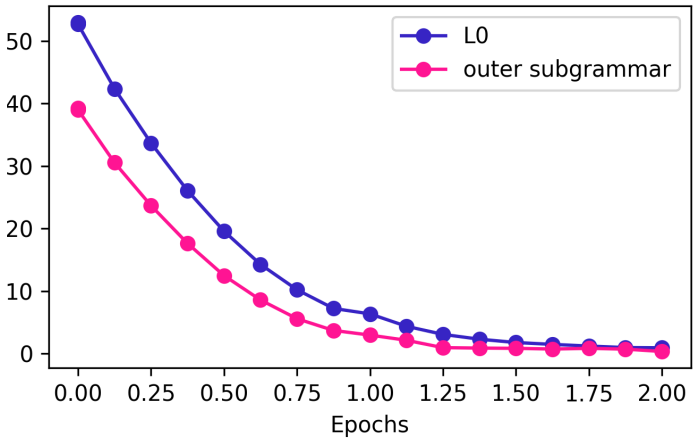
Empirical testbed: TLMs and PCFGs

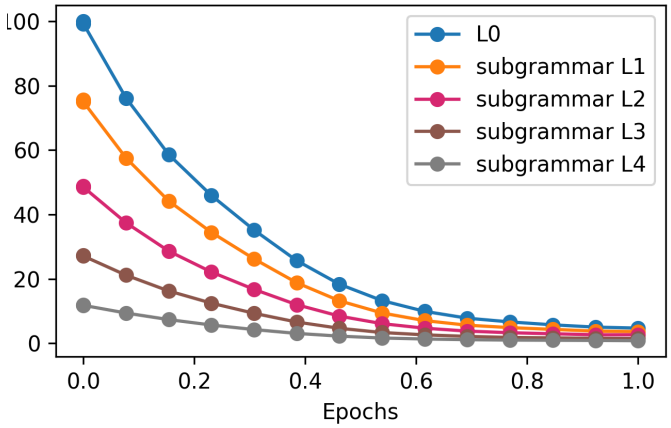
- **Conjecture: Children learn simpler language first** (e.g. outer subgrammars), LLMs will too?
- **A good moment to introduce our empirical framework…**
- Train **tiny** models (tiny transformers) on PCFGs we define ourselves
  - Can control how many subgrammars, depth of them (e.g. PCFG with 1 proper inner subgrammar, two, three, or a “tower” of nested sub grammars)

What did we find?

- **Conjecture: Children learn simpler language first** (e.g. outer subgrammars), LLMs will too?







- **3 top level subgrammars** **Outer subgrammar** **4 nested inner subgrammars**
- Appears to learn *all subgrammars in parallel, in all cases!!!*

LM Loss vs Subgrammars

- Conjecture — maybe LMs learn subgrammars in parallel because loss obeys a recursive formula over the sub grammars?
- Conj There exists a simple formula *f* such that, for PCFG **G** with (inner/outer) subgrammars $G_1, \ldots, G_k$:

  $\mathrm{KL}(G \mid Q_\theta) = f(\mathrm{KL}(G_1 \mid Q_\theta), \cdots, \mathrm{KL}(G_k \mid Q_\theta))$

- **Problem: what exactly is** $\mathrm{KL}(G_i \mid Q_\theta)$
  - G_i are grammars of "substrings" of G, so how to view Q_θ as a model of them?

Warm-up (if there's time)

- Consider Grammar P given by $S \to \alpha A \beta$, where A is a (proper) subgrammar

$$
\begin{aligned}
D_{\mathrm{KL}}(P \parallel Q) &= \sum_{a \in \Sigma^*} P_G(\alpha a \beta) \log \frac{P_G(\alpha a \beta)}{Q_\theta(\alpha a \beta)} \\
&= \sum_{a \in \Sigma^*} P_G(\alpha a \beta)[\log P_G(\alpha|\epsilon) + \log P_G(a|\alpha) + \log P_G(\beta|a\alpha) - \log Q_\theta(\alpha|\epsilon) \\
&\qquad - \log Q_\theta(a|\alpha) - \log Q_\theta(\beta|a\alpha)] \\
&= \frac{\log P_G(\alpha|\epsilon)}{\log Q_\theta(\alpha|\epsilon)} + \sum_a P_A(a) \frac{\log P_A(a)}{\log Q_\theta(a|\alpha)} + \sum_a P(a) \frac{\log P_G(\beta|\alpha a)}{\log Q_\theta(\beta|\alpha a)}
\end{aligned}
$$

Warm-up (if there's time)

- Consider Grammar P given by $S \to \alpha A \beta$, where A is a (proper) subgrammar

$$
\begin{aligned}
D_{\mathrm{KL}}(P \parallel Q) &= \sum_{a \in \Sigma^*} P_G(\alpha a \beta) \log \frac{P_G(\alpha a \beta)}{Q_\theta(\alpha a \beta)} \\
&= \sum_{a \in \Sigma^*} P_G(\alpha a \beta)[\log P_G(\alpha|\epsilon) + \log P_G(a|\alpha) + \log P_G(\beta|a\alpha) - \log Q_\theta(\alpha|\epsilon) \\
&\qquad - \log Q_\theta(a|\alpha) - \log Q_\theta(\beta|a\alpha)] \\
&= \frac{\log P_G(\alpha|\epsilon)}{\log Q_\theta(\alpha|\epsilon)} + \sum_a P_A(a) \frac{\log P_A(a)}{\log Q_\theta(a|\alpha)} + \sum_a P(a) \frac{\log P_G(\beta|\alpha a)}{\log Q_\theta(\beta|\alpha a)}
\end{aligned}
$$

"Morally" $D_{\mathrm{KL}}(P \parallel Q) = D_{\mathrm{KL}}(P \parallel Q)_\alpha + D_{\mathrm{KL}}(P \parallel Q)_A + D_{\mathrm{KL}}(P \parallel Q)_\beta$

KL sub-divergence

- For PCFG G and subgrammar A, the **KL subdivergence w.r.t. A** is the *expectation* of the KL-divergence of **A** and **Q as a model of A, given a prefix (from G) that commences A**

  Formally:

$$\mathrm{KL}(G \mid Q)_A = \mathbb{E}_{x \sim S \mid x=sa,\, a \in A} \left[\mathrm{KL}(A \parallel Q(\cdot \mid s)) \right]$$

$$= \sum_{s \in \Sigma^*} P_G(s \mid \epsilon) P_G(A \mid s) \sum_{a \in \Sigma^*} \mathrm{KL}(A \parallel Q(\cdot \mid s))$$

Theorem: Recursive formula for KL

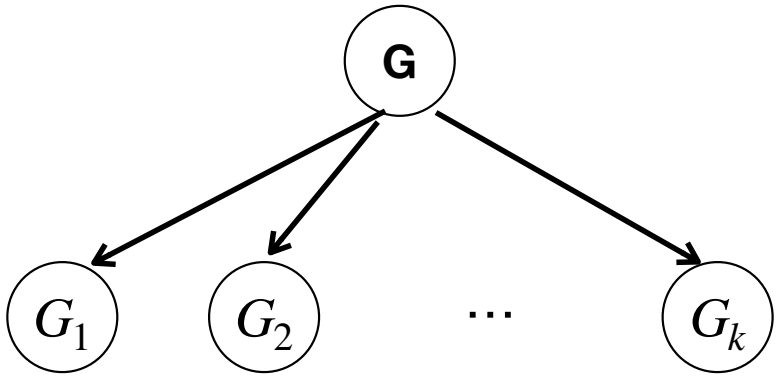
- WLOG, any PCFG G can be written such that S expands only to non-terminals
    - For any terminal on RHS of S, replace with new non-terminal …
- **Theorem.** Let G be PCFG as above with top-level subgrammars $G_1, \ldots, G_k$:

$$\mathrm{KL}(G \mid Q_\theta) = \sum_{i=1}^{k} \mathrm{KL}(G \mid Q_\theta)_{G_i}$$

KL (Loss) expands over TOP level sub grammars

- **Theorem.** Let G with top level subgrammars $G_1, \ldots, G_k$:

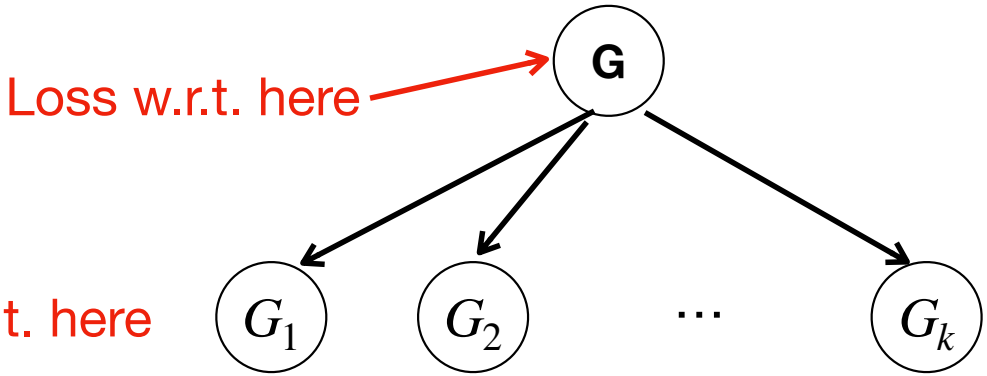
$$\mathrm{KL}(G \mid Q_\theta) = \sum_{i=1}^{k} \mathrm{KL}(G \mid Q_\theta)_{G_i}$$



KL (Loss) expands over TOP level sub grammars

- **Theorem.** Let G with sub grammars $G_1, \ldots, G_k$:

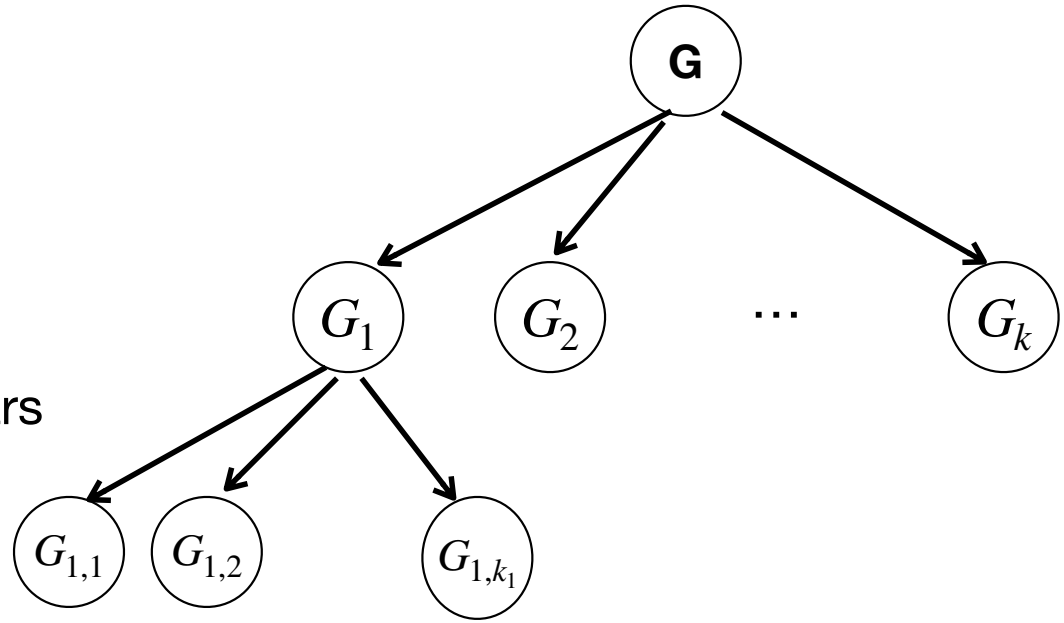
$$\mathrm{KL}(G \mid Q_\theta) = \sum_{i=1}^{k} \mathrm{KL}(G \mid Q_\theta)_{G_i}$$



KL (Loss) expands over TOP level sub grammars

- **Theorem.** Let G with sub grammars $G_1, \ldots, G_k$:

$$\mathrm{KL}(G \mid Q_\theta) = \sum_{i=1}^{k} \mathrm{KL}(G \mid Q_\theta)_{G_i}$$



- But now suppose a subgrammar decomposes into its *own* sub grammars
- Recursive formula applies recursively!

Corollary: KL decomposed recursively

- **Corollary** Let G be PCFG with “leaf” subgrammars (in DAG decomposition into inner sub grammars) $Z_1, \ldots, Z_k$, then

$$\mathrm{KL}(G \mid Q_\theta) = \sum_{i=1}^{k} \mathrm{KL}(G \mid Q_\theta)_{Z_i}$$

Finally - self loops?

- We’ve ignored self-loops in the DAG decomposition
- We’ll account for them, giving more precise recursive formula for KL of a PCFG

Theorem with Expected Recursion

- **Definition** (for self-looping grammar) Let the ***recursion*** R be the number of times S occurs in a derivation of G (not counting the initial S that gets expanded).

- **Theorem** Let G have **proper** subgrammars $G_1, \ldots, G_k$ (top-level, or leaves)

$$\mathrm{KL}(G \mid Q_\theta) = \frac{\sum_{i=1}^{k} \mathrm{KL}(G \mid Q_\theta)_{G_i}}{1 - \mathbb{E}[R]}$$

  where negative KL indicates unbounded divergence.

Theorem with Expected Recursion

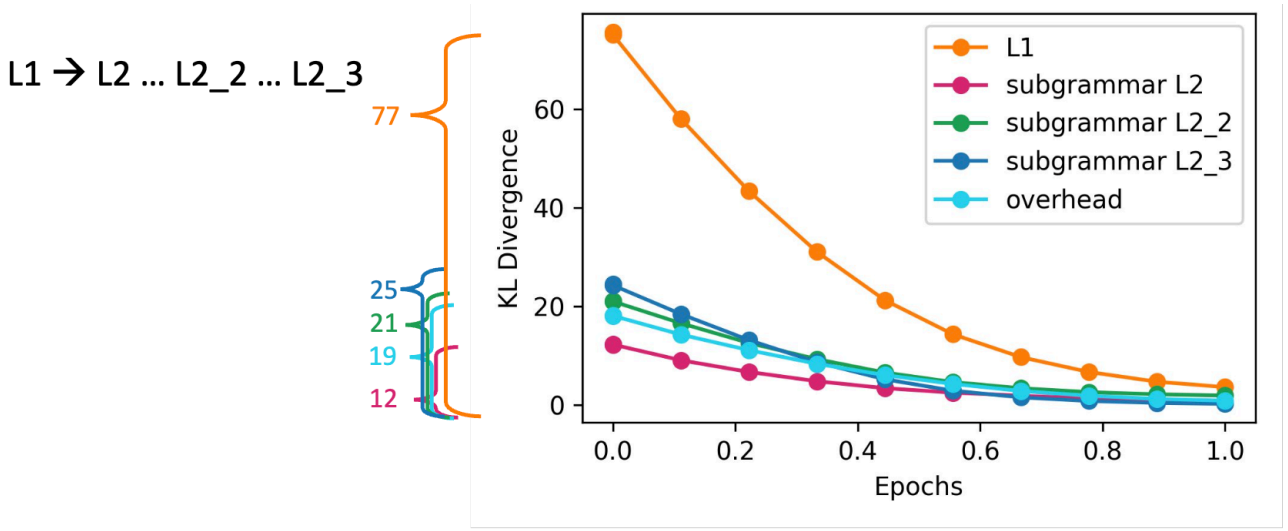
- **Definition** (for self-looping grammar) Let the ***recursion*** R be the number of times S occurs in a derivation of G (not counting the initial S that gets expanded).
- **Theorem** Let G have **proper** subgrammars $G_1, \ldots, G_k$ (top-level, or leaves)

$$\mathrm{KL}(G \mid Q_\theta) = \frac{\sum_{i=1}^{k} \mathrm{KL}(G \mid Q_\theta)_{G_i}}{1 - \mathbb{E}[R]}$$

Consider this… if expected recurrence not < 1, then PCFG generation never terminates (S keeps generating new S), so divergence "multiplied" infinitely

Revisiting loss decomposition

- Remember PCFG with 3 top level subgrammars…

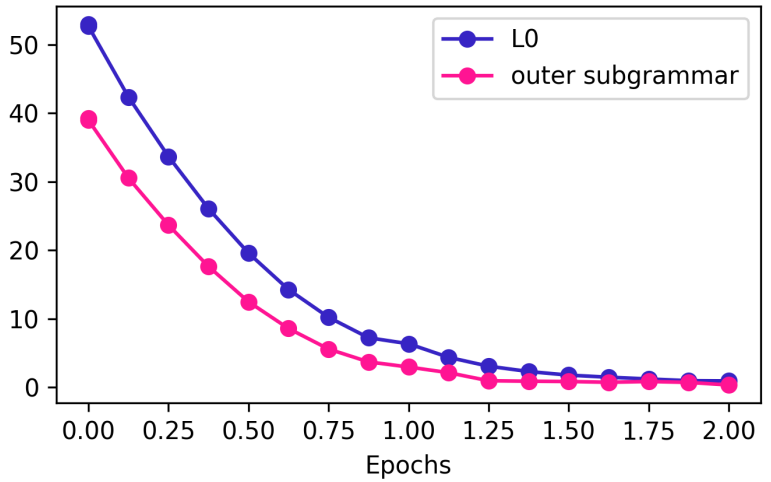


KL is exact sum over sub-divergence at any time step in training

A word on outer subgrammars

- A similar **linear recursion** theorem holds for *outer* sub grammars — see paper if interested!

  (Predicts that loss will train as on the right)



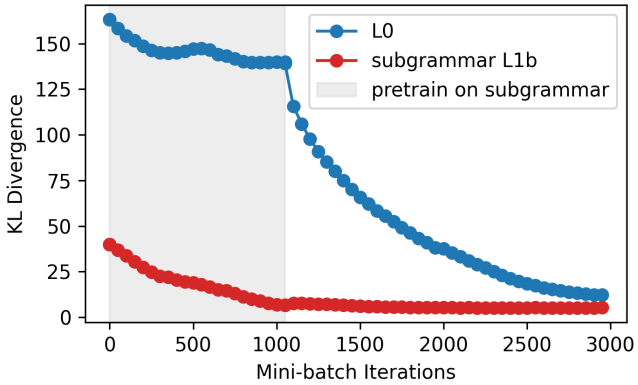
Experiments using Tiny LMs and CFGs

Exploiting SLMs and small CFGs: empirical investigations

- That's most of the theory for the day
- But we introduced this *experimental* framework too: train SLMs on synthetic, controlled CFGs (we decide the substructure)
- What else can we ask?
  - 1) Exploit subgrammar structure to improve performance?
  - 2) How is compositionality represented in weight space?
  - 3) Do LMs really "know" the compositional (subgrammar) structure?

Pre-training LMs on subgrammars

- ***Idea:*** First train LM on *just* subgrammar, then full grammar. What changes?

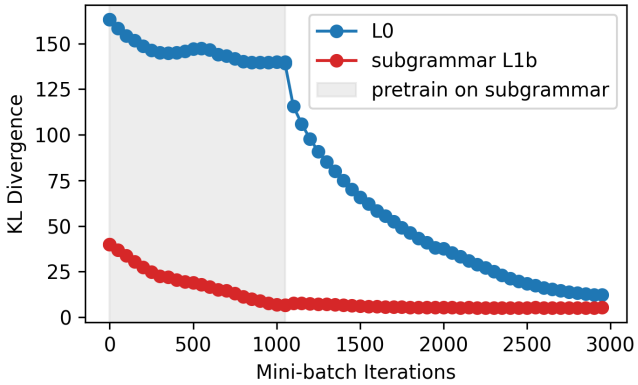


  Above: PCFG where $S \rightarrow AB$, training first on A only, then whole grammar.

- Observations: does not forget subgrammar during full training

Pre-training LMs on subgrammars

- ***Idea:*** First train LM on *just* subgrammar, then full grammar. What changes?

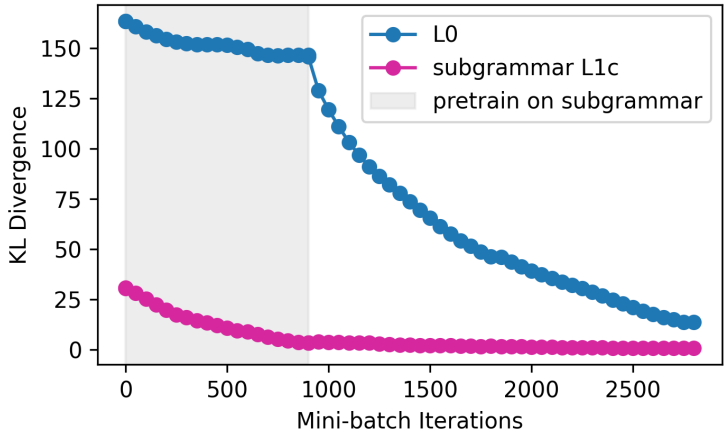


  Above: PCFG where $S \rightarrow AB$, training first on A only, then whole grammar.

- Observations: does not forget subgrammar during full training
  Not surprising because A is a "prefix" of all strings of full grammar?

Pre-training LMs on subgrammars

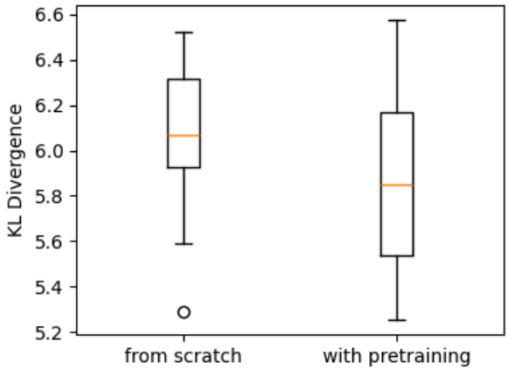
- PCFG where $S \rightarrow AB$, training first on **B** only, then whole grammar.



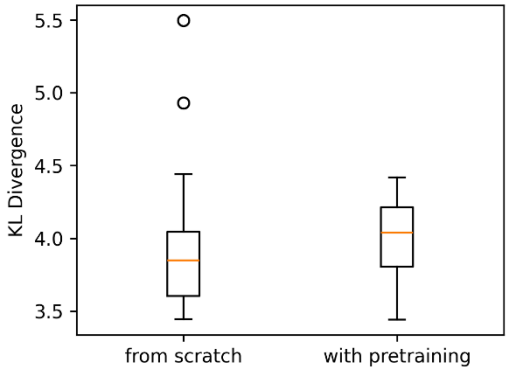
- Further experiments showed, retains subgrammar in *any* position within grammar.

Pre-training LMs on subgrammars

- Does it help with final performance?
- **Answer: yes, in very small models (relative to complexity of the language)**



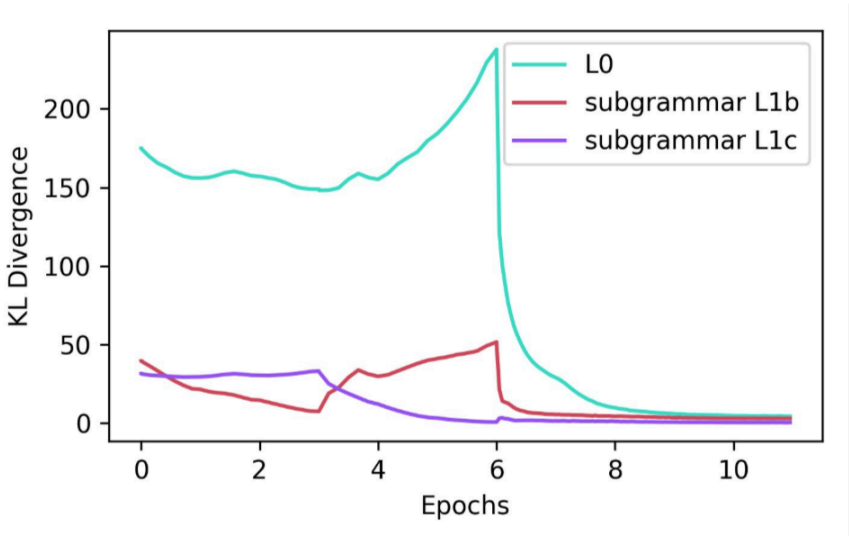
(a) Two-layer Transformer



(b) Four-layer Transformer

Pre-training with sub grammars via “curriculum learning”

- For G with (top level) subgrammars A, B, consider curriculum learning algorithm:
  1) train on A only
  2) train on B only
  3) train on full grammar
- Issue: forgets A (or B) when training on other (*no* reinforcement)
- **Not** faster training algorithm overall (vs directly training on full grammar)
- **But** observe “phantom memory” of previous sub grammars in slope…



Study internal model activations

- ***Idea***: maybe a model first trained on subgrammar, then full grammar represents sub vs. full grammar differently
- Compare activations (outputs of neurons) when feeding in subgrammar strings, full grammar strings
    - Issue: representations in model highly non-linear, dot-products very hard to interpret
- **Centered Kernel Alignment (CKA) idea:**
  1) For 30 random initializations, train with or without subgrammar training
  2) Take dot-products of subgrammar activations strings, full-grammar strings
  3) Take dot-products of those dot-products (similarity) between initializations

Study internal model activations

- **Centered Kernel Alignment (CKA) idea:**
  1) For 30 random initializations, train with or without subgrammar training
  2) Take dot-products of subgrammar activations strings, full-grammar strings
  3) Take dot-products of those dot-products (similarity) between initializations

  Will models trained on subgrammar "pair up" subgrammar strings vs full grammar strings more similarly (between each other)?

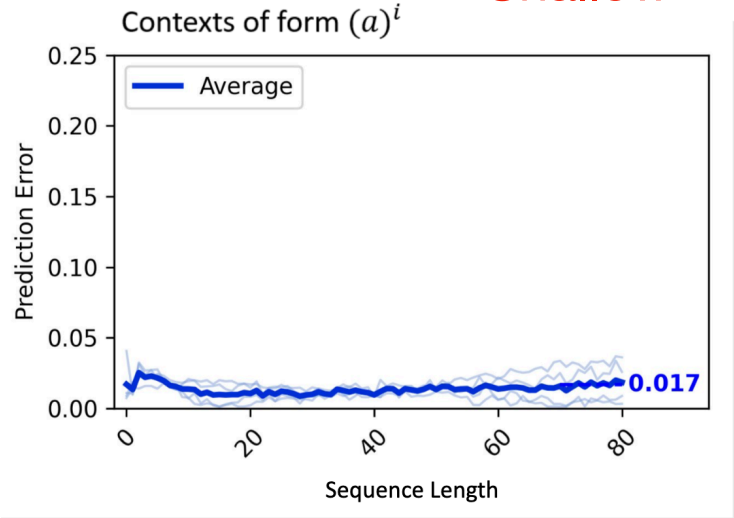
Yes, definitively:

	Two-layer Transformer				Four-layer Transformer	
	Pretraining 10 epochs		Pretraining 20 epochs		Pretraining 10 epochs	
	Attention	MLP	Attention	MLP	Attention	MLP
Full grammar sequences						
From Scratch	0.258	0.535	0.249	0.535	0.249	0.469
With Pretraining	0.281	0.534	0.303	0.511	0.323	0.491
Percentage change (%)	*+8.9*	*-0.2*	*+21.7*	*-4.7*	*+8.3*	*+1.0*
Subgrammar sequences						
From Scratch	0.298	0.561	0.288	0.558	0.295	0.513
With Pretraining	0.339	0.566	0.348	0.544	0.347	0.525
Percentage change (%)	*+13.8*	*-0.1*	*+20.8*	*-2.6*	*+10.7*	*+1.9*
Subgrammar pretraining only	0.288	0.558	0.288	0.558	0.295	0.523

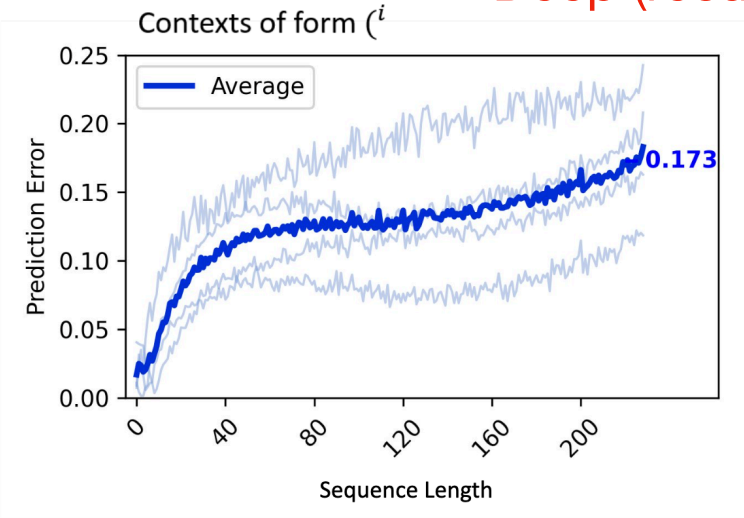
Do LMs “know” substructure?

- Let G be PCFG of balanced parentheses (with terminal “a” instead of epsilon)
- For LM trained to perfect training accuracy, probe on **longer and longer contexts**

Shallow



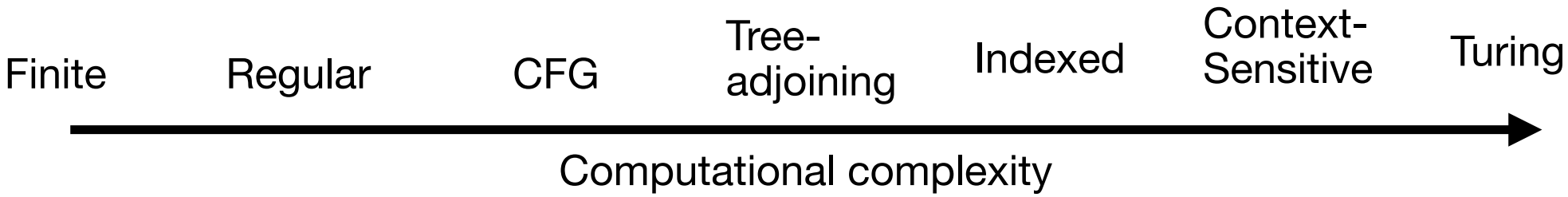
Deep (recursing)



Just a flavor of the kinds of experimentation one can do!

- Other than loss additivity, these *empirical* phenomena lack theoretical basis
  - Look for more subgrammar structure in mathematics of LM training

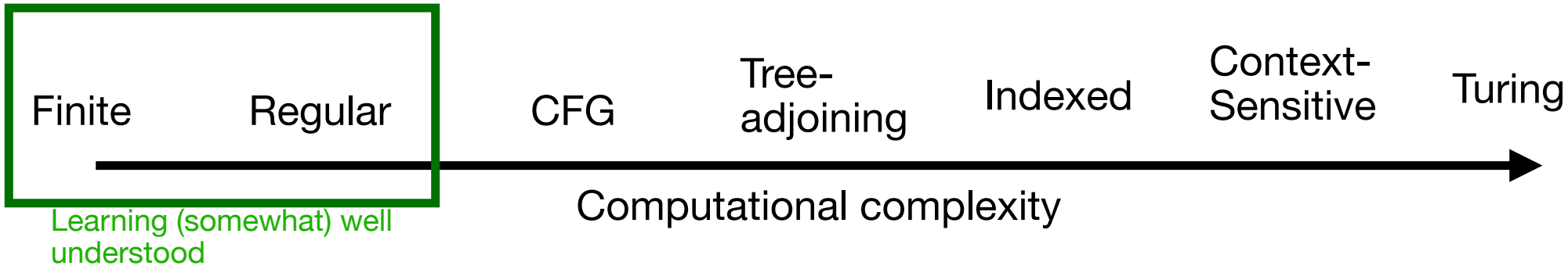
- **Another direction:** CFGs are a formalism at a particular level of the complexity hierarchy of formal languages:



Just a flavor of the kinds of experimentation one can do!

- Other than loss additivity, these *empirical* phenomena lack theoretical basis
  - Look for more subgrammar structure in mathematics of LM training

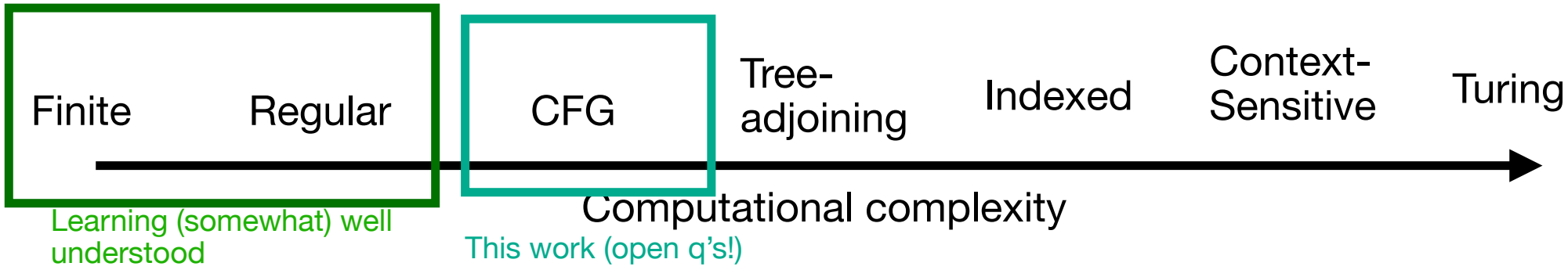
- **Another direction:** CFGs are a formalism at a particular level of the complexity hierarchy of formal languages:



Just a flavor of the kinds of experimentation one can do!

- Other than loss additivity, these *empirical* phenomena lack theoretical basis
  - Look for more subgrammar structure in mathematics of LM training

- **Another direction:** CFGs are a formalism at a particular level of the complexity hierarchy of formal languages:



Just a flavor of the kinds of experimentation one can do!

- Other than loss additivity, these *empirical* phenomena lack theoretical basis
  - Look for more subgrammar structure in mathematics of LM training

- **Another direction:** CFGs are a formalism at a particular level of the complexity hierarchy of formal languages: