Discovering the Rules of language using Bayesian Program Learning

May 4, 2018

Abstract

Both children and linguists confront a similar problem of inference: given utterances produced by speakers, together with aspects of the meaning of those utterances, discover the grammatical principles that relate form to meaning. We study this abstract computational problem within the domain of morphophonology, contributing a computational model that learns phenomena from many natural languages and generalizes in humanlike ways from data used in behavioral studies of artificial grammar learning.

Our work draws on two analogies. The *child-as-linguist analogy* holds that both children and linguists must solve the same abstract inductive reasoning problem, even though the nature of the input data and underlying mental algorithms are surely different in precise detail. Accordingly we isolate the problem of learning morphophonological systems, and show that a single solution to this problem can capture both linguistic analyses from natural languages and infant rule learning of artificial languages. We adopt the framework of "Bayesian Program learning" (BPL) – in which learning is formulated a synthesizing a program which compactly describes the input data. This *learning-as-programming analogy* lets us exploit recent techniques from the field of program synthesis to induce morphophonological rules from data. While *child-as-linguist* poses the computational problem, *learning-as-programming* offers a solution.

1 Introduction

Both children and linguists confront a similar problem of inference: given utterances produced by speakers, together with aspects of the meaning of those utterances, discover the grammatical principles that relate form to meaning. Here, we study this abstract computational problem within the domain of morphophonology, the part of language that builds up words from simpler parts such as stems, prefixes, and suffixes (morphology) and then specifies how the words are realized as sound patterns (phonology). Our contribution is a computational model that learns phenomena from 26 natural languages and generalizes in humanlike ways from data used in infant studies of artificial grammar learning.

Our work draws on two analogies. The *child-as-linguist analogy* holds that both children and linguists must solve the same abstract inductive reasoning problem, even though the nature of the input data, the process of theory construction, and underlying mental algorithms are surely different in precise detail. Accordingly we isolate the problem of learning morphophonological systems, and show that a single solution to this problem can capture both linguistic analyses from natural languages and infant rule learning of artificial languages.

We adopt the framework of "Bayesian Program learning" (BPL) – in which learning is formulated a synthesizing a program which compactly describes the input data. This *learning-as-programming analogy* explains the flexibility and richness of a grammar as a consequence of While *child-as-linguist* poses the computational problem, *learning-as-programming* offers a solution.

We develop a programming language tailored to expressing morphophonological generalizations. For children this programming language is innate and is commonly called "Universal Grammar" (UG). Linguists instead must uncover this programming language by looking at many languages and searching for commonalities. Learning what makes up UG unfolded over evolutionary time for children and is ongoing within linguistics. BPL reinterprets the problem of discovering UG. In BPL, UG is the programming language combined with an inductive bias over programs. Here we estimate this inductive bias empirically from naturally occurring morphophonological systems as a way of learning to learn grammars.

2 Morphophonological phenomena

What is phonology supposed to explain? Familiar phenomena from English help motivate phonological grammars. The plural of *cat* is pronounced kæts ("cats"), while the plural of *dog* is pronounced dɑgz ("dogs"). A phonological rule accounts for the alternation between s and z: z becomes s at the end of words whenever the preceding sound is unvoiced (does not involve vibrating the vocal cords). This generalization holds throughout the entirety of English. The count system of Tibetan presents another puzzle. "1" in Tibetan is pronounced ʔ̊jig, "10" is ʔ̊ju, yet "11" (= 10 + 1) is ʔ̊jugʔ̊jig, rather than ʔ̊juʔ̊jig. The explanation lies in Tibetan phonology, which deletes word initial consonants followed by another consonant (Table 1). In this analysis of Tibetan, "1" has a latent "underlying" representation as gʔ̊jig but is pronounced as ʔ̊jig.

The aim of phonology is to explain pronunciations of words in terms of latent underlying representations (eg, gʔ̊jig for Tibetan "1") and the rules that give observed surface pronunciations from underlying forms (eg, deleting word initial consonants). Context-sensitive rewrite systems are a Turing complete representation that also naturally express phonological rules [1]. We use context-sensitive rewrites as a representation for programs that implement a language's phonology. For example, the rewrite in Table 1, `C`→ ∅/`#_C`, deletes (→ ∅) consonants (`C`) at word boundaries (`#`) when followed by another consonant (`C`). Thus phonological rule learners, whether they be children, linguists, or machines, must solve a program induction problem.

Example data		Phonology
1	ǰig	
4	ši	
5	ŋa	
9	gu	
10	ǰu	
11 (= 10 + 1)	ǰugǰig	*Consonant cluster reduction:*
14 (= 10 + 4)	ǰubši	C→ ∅ / # _ C
15 (= 10 + 5)	ǰuŋa	
19 (= 10 + 9)	ǰurgu	
40 (= 4 + 10)	šibǰu	
50 (= 5 + 10)	ŋabǰu	
90 (= 9 + 10)	gubǰu	

Table 1: Tibetan count system.

2.1 Inducing Programs

We model the program learning problem as Bayesian inference, treating the pronunciations of words as observed data and the phonology as a latent variable whose values are programs. So the model's beliefs about possible programs are:

$$P(\text{program}|\text{words}) \propto P(\text{program}|\text{UG}) \prod_{w \in \text{words}} \sum_{u \in \text{underlying forms}} P(u)P(w|u, \text{program}) \quad (1)$$

Here, Universal Grammar (UG) is an inductive bias over programs. This inductive bias encodes knowledge of the program space (what rules are possible in human language and what rules are impossible), imparting a bias towards more naturally structured rules, like rules written in terms of syllables and consonants.

2.2 The role and importance of universal grammar

We claim any plausible theory of linguistic rule learning needs a universal grammar. Studies of infant grammar learning suggests that babies come equipped with prior knowledge of what language should look like. This prior knowledge constrains the space of allowed generalizations, accounting for the astonishing pace at which children acquire language. This line of reasoning is a "poverty-of-the-stimulus" style argument: if children can induce so much grammar from such meager data, something else must bridge the gap. Because universal grammar also constrains the space of allowed grammars, we would expect that humans do not learn certain generalizations, even when the generalizations seem to hold in the data. Indeed this is the case: this "surfeit-of-the-stimulus" style argument explains phenomena in natural languages (for example, statistical regularities that seem to hold in Turkish are not actually used by speakers) and in infant artificial language learning (for example, infants acquire certain phonological generalizations from just a few examples, while never acquiring other generalizations that are seemingly just a simple). While previous accounts of universal grammar were abstract, our model is concrete and computational. We define universal grammar as a probability mass function that puts zero mass on impossible or unattested generalizations, places more mass on programs licensing the generalizations that people learn quickly and naturally, and then smoothly falls off in probability mass as the programs become less natural.

2.3 Learning natural language phonology

Our model learns a diverse set of phonological generalizations found in natural languages. We compiled a multilingual data set of words from standard linguistic textbooks. In these data sets, our model recovers allophonic relationships (e.g., l vs. r in Japanese) and systems that form different inflections of words (e.g., English verb inflections, or the Tibetan count system). Systems like these have received intense study within computational linguistics. But no prior model could explain the diversity of phonological phenomena found even in a textbook. To explain many diverse phenomena we need a rich, flexible representation – programs – coupled to a linguisticly informed inductive bias – universal grammar. Our model learns simple generalizations like English verbal inflections, as well as phenomena like vowel harmony (in the language Kikuria, the word tiga becomes tegera in a different inflection, or ruga becomes rogera), or tonal rules in languages with tone (in the language Kerewe, different inflections cause tones to appear, disappear, or move to different parts of a word).

For these results, the learning-as-programming analogy does the work for us. BPL moves beyond the impoverished representations of prior learners (neural networks as in, or rigid classes of symbolic rules as in) instead represented generalizations as programs. We claim that programs are a sufficiently general representation to support the kinds of rules found in phonology.

3 The Model

4 Artificial Grammar Learning

The fundamental hypothesis underlying AGL research is that artificial grammar learning must engage some shared resource with first language acquisition – and so we can gain insights into first language acquisition by instead studying AGL in controlled experiments.

Our contribution to AGL research is the demonstration that a single BPL model can explain (1) that infants quickly acquire certain simple grammars very easily, and at the same time (2) those same infants go on to learn the very complicated grammar of their first language. We collected grammars from AGL datasets [2, 3, 4] and presented our BPL learner with data drawn from these grammars (Table 2), finding that our model recovers the generative processes that underlies these data sets.

Grammar	Example input to learner	MAP grammar	Natural language analogues
ABA (same/different/same) Marcus et al. 1999.	wofewo lovilo fimufi	$\varnothing \rightarrow \sigma_i$ / #_$\sigma\sigma_i$	Reduplication (eg, Tagalog)
ABB (different/same/same) Marcus et al. 1999.	wofefe lovivi fimumu	$\varnothing \rightarrow \sigma_i$ / σ_i_#	Reduplication (eg, Tagalog)
ABx (different/different/constant)	wofeka lovika fimuka	stem+x	concatenative morphology
AAx (same/same/constant) Gerken 2006.	wowoka loloka fifika	stem + x $\varnothing \rightarrow \sigma_i$ / #_σ_i	reduplication concatenative morphology
AxA (same/constant/same) Gerken 2006.	wokawo lokalo fikafi	x+ stem $\varnothing \rightarrow \sigma_i$ / # _ σ σ_i	Infixing Reduplication
Pig Latin	pɪg→ɪgpe latɪn→atɪle æsk→æske	$\varnothing \rightarrow$C$_i$ / #C$_i$ [ ]* _ # $\varnothing \rightarrow$ e / _ # C$\rightarrow \varnothing$ / # _	**Tim, is there an analog?** **word initial segments moves to word final?**

Table 2: Using BPL to model artificial grammar learning. Learner given 5 examples.

Just like in first language acquisition, these AGL learning setups introduce a trade-off between grammars that are likely and therefore small ("parsimony") and grammars that best fit the data. For example, a learner could infer a grammar that just memorized to the data (perfect fit but poor parsimony) or it could infer a grammar that can generate every possible word (parsimonious but a poor fit). Effective grammar learners must navigate this trade-off.

To analyze this trade-off, we calculated its shape in the form of a **pareto front** [5]. In our setting, a pareto front is the set of all grammars that are not worse than another grammar along the two competing axes of parsimony and fit to data. Intuitively, grammars on the pareto front are ones which an ideal Bayesian or MDL learner prefers, *independent* of how the learner decides to relatively weight the prior and likelihood. Figure 2 diagrams the Pareto fronts for two AGL experiments as the number of example words provided to the learner is varied. What these Pareto fronts show is (1) the set of grammars entertained by the learner, and (2) how the learner weighs these grammars against each other as measured by the prior (parsimony) and the likelihood (fit to the data). We believe that the Pareto front is a useful way of viewing the space of possible grammars, and understanding the different trade-offs between the grammars.

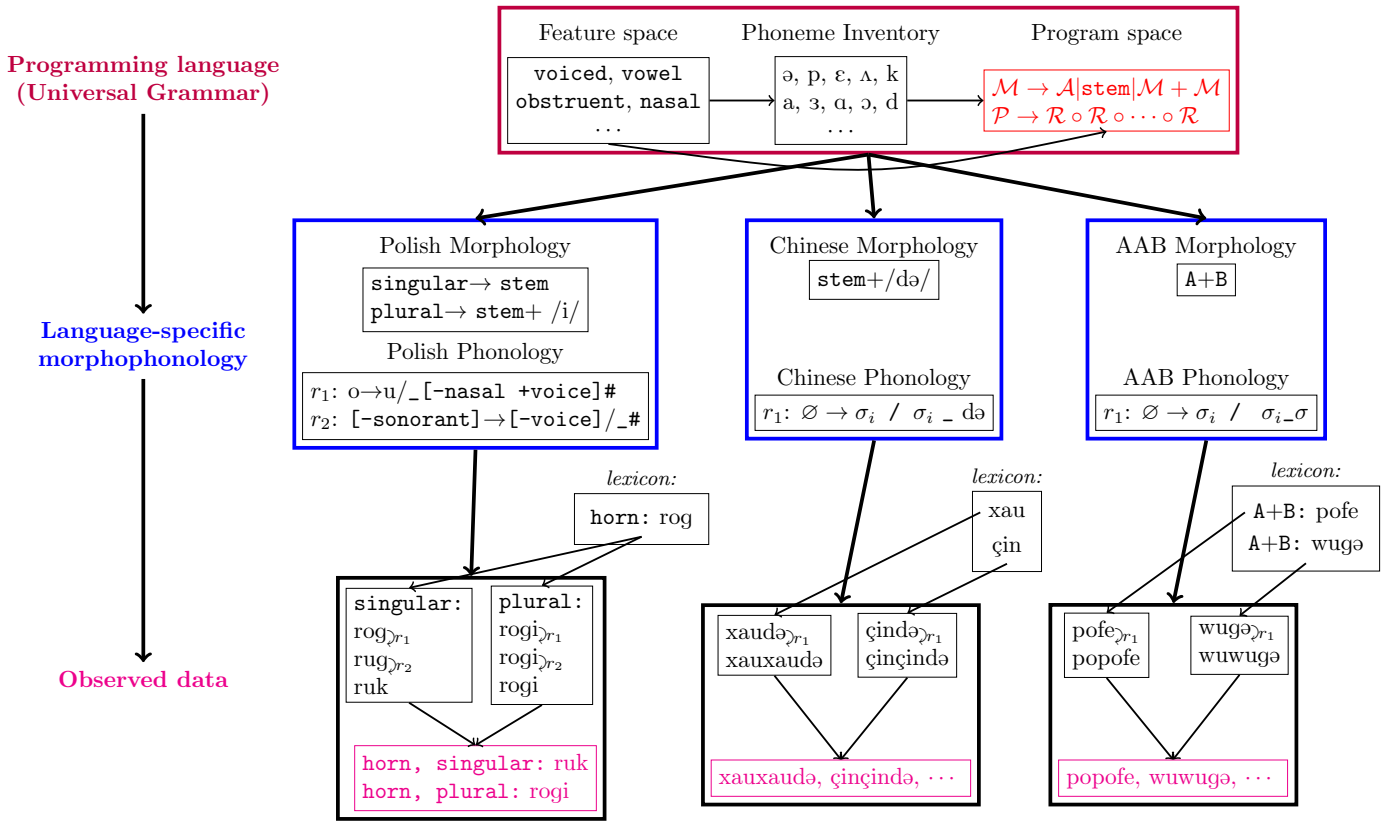

Figure 1: Red: Both the morphology $\mathcal{M}$ and phonology $\mathcal{P}$ are programs either build through concatenation or function composition (Tables 4 & 3). These programs explain how words are built in both natural and artificial grammars (blue). Learners observe pronunciations of words () in order to recover these programs, but with the aid of a strong inductive bias (Universal Grammar, purple)

.

Textbook Tibetan counting problem:

Number	Surface form
10	dʒu
1	dʒig
11	dʒugdʒig
4	ʃi
40	ʃibdʒu
9	gu
19	dʒurgu
5	ŋa

Our model's solution:

Induced phonological rule: C→ ∅/#_C

(Reduce word initial consonant clusters)

Induced underlying representations:

1	gdʒig
4	bʃi
5	ŋa
9	rgu
10	bdʒu

Figure 3: Example textbook phonology exercise (left) and the solution our model discovers for it (right). To solve this problem the model must jointly infer unobserved underlying representations for each of the Tibetan numbers, along with a phonological rule that explains their surface pronunciations. The same code that calculated the Pareto fronts (Figure 2) produces this analysis of the Tibetan count system.

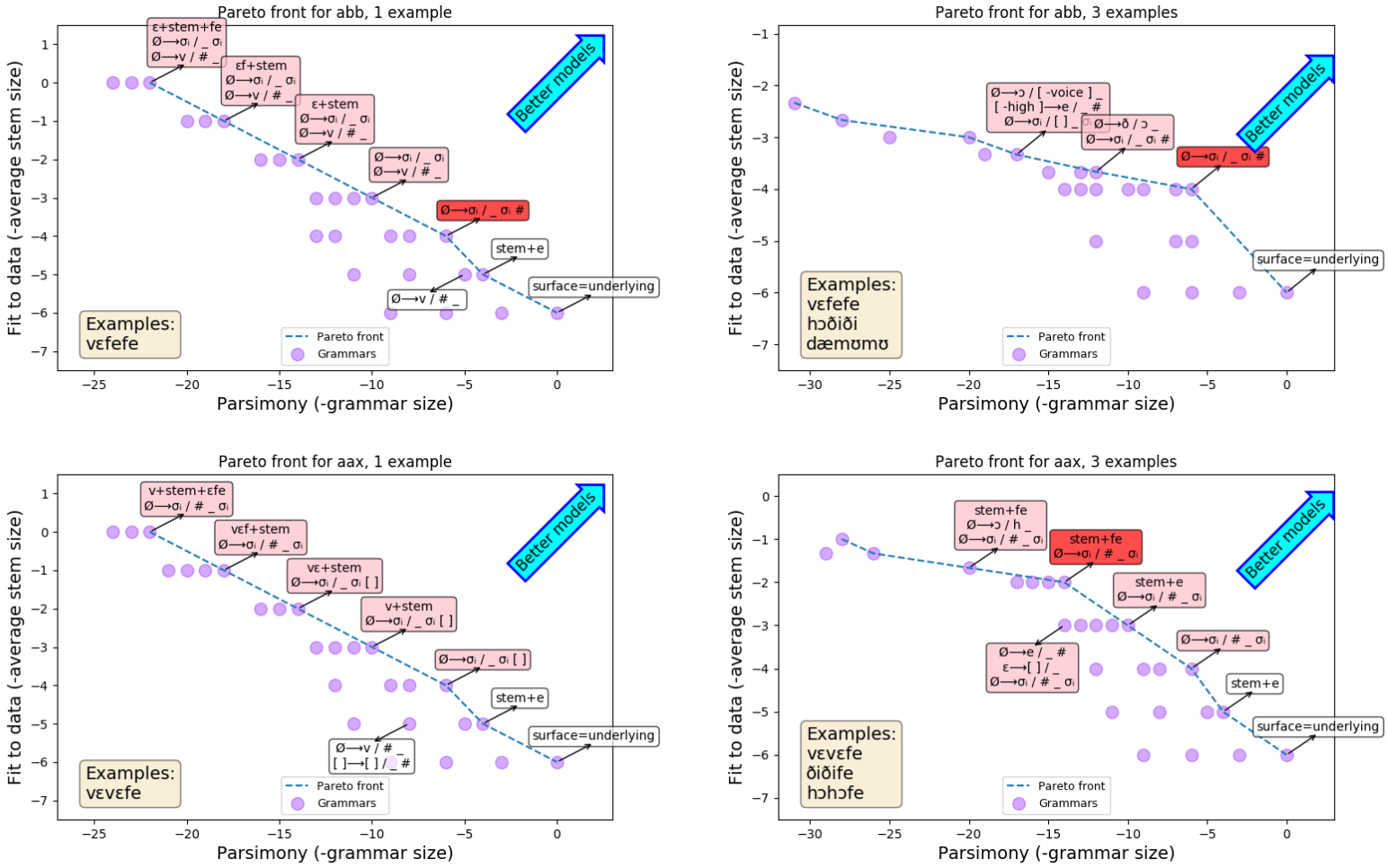

Figure 2: Pareto fronts for ABB (Marcus 1999; top two) & AAX (Gerken 2006; bottom two) learning problems fo either one example word (left column) or three example words (right column). Red shade: ground truth grammar. Pink shade: shares structure with ground truth grammar. White shade: incorrect generalizations. As the number of examples increases, the Pareto fronts develop a sharp kink around the ground truth grammar, which indicates a stronger preference for the correct grammar. With one example the kinks can still exists but are less pronounced.

4.1 Beyond artificial grammars: Applying BPL to natural language

A successful model of artificial grammar learning should also explain at least some aspects of first language acquisition. As a first step in this direction, we take the same model that we applied to these AGL learning problems and use it to solve phonology textbook exercises. Each of these textbook exercises is an inductive reasoning problem where the learner must explain surface pronunciations in terms of a latent generative model (a program). Figure 3 shows our model solving a representative textbook problem.

TIM: Should I talk about UG? We needed it for Tibetan. We probably don't have space.

References

[1] N. Chomsky and M. Halle. *The sound pattern of English*. Studies in language. Harper & Row, 1968.

[2] LouAnn Gerken. Infants use rational decision criteria for choosing among models of their input. *Cognition*, 115(2):362–366, 2010.

[3] Gary F Marcus, Sugumaran Vijayan, S Bandi Rao, and Peter M Vishton. Rule learning by seven-month-old infants. *Science*, 283(5398):77–80, 1999.

[4] Michael C Frank and Joshua B Tenenbaum. Three ideal observer models for rule learning in simple languages. *Cognition*, 120(3):360–371, 2011.

[5] Christopher A Mattson and Achille Messac. Pareto frontier based concept selection under uncertainty, with visualization. *Optimization and Engineering*, 6(1):85–115, 2005.

[6] Adam Albright and Bruce Hayes. Rules vs. analogy in english past tenses: A computational/experimental study. *Cognition*, 90:119–161, 2003.

[7] Sharon Goldwater and Mark Johnson. Learning ot constraint rankings using a maximum entropy model.

[8] Ryan Cotterell, Nanyun Peng, and Jason Eisner. Modeling word forms using latent underlying morphs and phonology. *Transactions of the Association for Computational Linguistics*, 3:433–447, 2015.

[9] Roni Katzir Ezer Rasin, Iddo Berger. Learning rule-based morpho-phonology. *Submitted.*, December 2015.

[10] LouAnn Gerken. Decisions, decisions: Infant language learning when multiple generalizations are possible. *Cognition*, 98(3):B67–B74, 2006.

[11] Kirk H Smith. Grammatical intrusions in the recall of structured letter pairs: Mediated transfer or position learning? *Journal of Experimental Psychology*, 72(4):580, 1966.

[12] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[13] Michael Becker, Nihan Ketrez, and Andrew Nevins. The surfeit of the stimulus: Analytic biases filter lexical statistics in turkish laryngeal alternations. *Language*, 87(1):84–125, 2011.

5 Appendix

Grammar rule	English description
$\mathcal{P} \rightarrow \mathcal{R} \circ \mathcal{R} \circ \cdots \circ \mathcal{R}$	Phonology $\mathcal{P}$ is a sequence of compositions of rewrites $\mathcal{R}$
$\mathcal{R} \rightarrow \mathcal{F} \longrightarrow \mathcal{C} / \mathcal{T} \_ \mathcal{T}$	Rewrite focus $\mathcal{F}$ to structural change $\mathcal{C}$ between triggers $\mathcal{T}$
$\mathcal{T} \rightarrow \#\mathcal{T}' \vert \mathcal{T}'$	Triggers optionally match end of string
$\mathcal{T}' \rightarrow \epsilon \vert \mathcal{X}\mathcal{T}' \vert \mathcal{X}^*\mathcal{T}'$	Triggers are sequences of matrices $\mathcal{X}$ possibly with Kleene star
$\mathcal{X} \rightarrow a \vert t \vert s \vert \cdots$	Matrices can be constant phonemes
$\mathcal{X} \rightarrow [\pm\mathcal{G} \pm\mathcal{G} \cdots \pm\mathcal{G}]$	Matrices check features $\mathcal{G}$
$\mathcal{X} \rightarrow \sigma$	Matrices can be whole syllables
$\mathcal{G} \rightarrow \text{voice} \vert \text{nasal} \vert \cdots$	Standard phonological features
$\mathcal{F} \rightarrow \mathcal{X}$	Focus can be a feature matrix
$\mathcal{F} \rightarrow \varnothing$	Focus can be empty (insertion rule)
$\mathcal{F} \rightarrow \mathbb{Z}$	Focus can be one of the triggers (copies it)
$\mathcal{C} \rightarrow \mathcal{X}$	Structural change can be a feature matrix
$\mathcal{C} \rightarrow \varnothing$	Deletion rule

Table 3: Grammar over phonologies. The non-terminal $\mathcal{P}$ is the start symbol for the grammar.

Grammar rule	English description
$\mathcal{W} \rightarrow (\mathcal{M})\mathcal{M}(\mathcal{M})$ $\mathcal{M} \rightarrow$ sequence of phonemes	Morphology builds words $\mathcal{W}$ by concatenating morphemes $\mathcal{M}$. Prefix/Suffix optional (in parentheses)

Table 4: Grammar over morphologies. The non-terminal $\mathcal{W}$ is the start symbol for the grammar.

5.1 Inference

The solver takes as input:

- A set of inflected surface forms, $\{x_n\}$. Here the variable x_n refers to all of the inflections of a word. So an example set of 3 inflected surface forms might be $\{(\text{fish}, \text{fished}), (\text{walk}, \text{walked}), (\text{eat}, \text{ate})\}$.
- A bound on the number of rules to consider synthesizing, B_R.

It produces as output:

- A set of underlying forms, $\{u_n\}$.
- A morphology consisting of a prefix ($\{p_i\}$) and a suffix ($\{s_i\}$) for each inflection
- A sequence of rules, $\{r_j\}_{j=1}^{B_R}$

It is *guaranteed* to discover the rules, morphology, and underlying forms minimizing:

$$\text{solver}(\{x_n\}, B_R) = \underset{\{r_j\}_{j=1}^{B_R}, \{u_n\}, \{p_i\}, \{s_i\}}{\arg\min} \sum_j \text{cost}(r_j) + \sum_i (\text{cost}(p_i) + \text{cost}(s_i)) + \sum_n \text{cost}(u_n) \tag{2}$$

In exchange for this strong guarantee, however, it has only a very weak guarantee on the amount of time it will take to discover the solution: in the worst case, it will take time exponential size of the problem. In practice it takes close to a day for it to do exact search over 3 rules. For easy problems, like the AGL experiments, I can get away with 4 rules if I wait for a few hours.

One can also ask the solver for the top K solutions minimizing the cost function, which I will write:

$$\text{solver}_K(\{x_n\}, B_R) = \text{the } K \text{ minimum cost solutions.} \tag{3}$$

One can also provide additional constraints to the solver, for example telling it what the underlying forms are, or telling it what some of the rules are. I will write these constraints by adding an extra argument to the solver that looks like conditioning in Bayesian inference. For example,

$$\text{solver}_K(\{x_n\}, B_R | u_1 = \text{fish}) = \text{the } K \text{ minimum cost solutions where } u_1 \text{ has to be fish.} \tag{4}$$

So now the game is to leverage this solver as a "black box" inside of an incomplete search algorithm. A key subroutine in my attempts at building such a search algorithm is `solveIncrementalChange`, which takes as input an existing set of rules ($\{r_j^e\}_{j=1}^{B_R}$) along with a **search radius** (R), and solves for a solution that changes at most **search radius** rules.

$$\text{solveIncrementalChange}_K(\{x_n\}, \{r_j^e\}_{j=1}^{B_R}, R) = \text{solver}_K(\{x_n\}, B_R + 1 | r_j \neq r_j^e \text{ at most } R \text{ times}) \tag{5}$$

So `solveIncrementalChange` can add a rule (see $B_R + 1$). I didn't build in any explicit mechanism for reordering rules – if you set the search radius R to 2, then it could effectively swap 2 rules.

In addition to synthesizing large programs, we also have to deal with large data sets. Armando already came up with a nice trick for this called CEGIS[1]. The idea behind CEGIS is that you synthesize a program from one example, and then you check to see if that program is inconsistent with any other examples. If not then you are done. Otherwise, you have found a **counterexample**. You add the counterexample to your training data set and repeat.

Combining `solveIncrementalChange` with CEGIS gives the skeleton of the inference algorithm that I have been experimenting with, called `incrementalCEGIS`. See the pseudocode at the top of the next page:

[1] counterexample guided inductive synthesis

`incrementalCEGIS`:
Input: Matrix of inflected forms for N lexemes, X
Hyperparameter: Batch size B
Output: underlying forms, $\{u_n\}$; prefixes $\{p_i\}$; suffixes $\{s_i\}$; rules $\{r_j\}$
// Initialize training set T with B examples
Set T = inflections for B lexemes in X
// Find a solution for examples, using as many rules as you need (∞)
Set $\{r_j\}_{j=1}^{B_R}, \{p_i\}, \{s_i\} = \text{solver}(T, \infty)$
repeat
  // Find the set of counterexamples C inconsistent with our current guess as to what the solution is
  Set $C = \{x | x \in X$ and x inconsistent with $\{r_j\}_{j=1}^{B_R}, \{p_i\}, \{s_i\}\}$
  // If there are no counterexamples as we are done
  if $C = \varnothing$ **then return** $\{r_j\}_{j=1}^{B_R}, \{p_i\}, \{s_i\}$
  // Add a batch of counterexamples to the training set
  $T \leftarrow T \cup \{C_j\}_{j=1}^{B}$
  // Start out with a search radius of 1 and increase it until we find a way of accommodating another example
  for $R = 1, 2, 3, 4, ...$ **do**
    Set synthesisResult= solveIncrementalChange$(T, \{r_j\}, R)$
    if synthesisResult$\neq \perp$ **then**
      Set $\{r_j\}_{j=1}^{B_R}, \{p_i\}, \{s_i\}$ =synthesisResult
      break out of loop over R // Successfully updated both our solution and the training set

One important feature of this approach is that it does not reuse previous inferences about the morphology or the underlying forms. I think we want to add this. For example, we might allow it to change up to R underlying forms, or disallow changes in the morphology once the training set is big enough.
Variations of **incrementalCEGIS** that I have tried:

Maintaining a population of solutions. Instead of taking the top 1 solution every time you synthesize, try taking the top K. There is some trade-off here between the cost of solving for more solutions and the benefit of not needing as many counterexamples. I have not done any systematic experiments, but my general sense is that this trade-off favored $K = 1$ in practice (so not maintaining a population of solutions).

Letting the solver insert the new rule anywhere vs constraining it to either append or prepended the new rule. Sketch uses up a lot of memory whenever I let it insert the rule anywhere – not sure why this is.

Giving sketch more problems to solve but making each problem smaller. Here what I do is have an outer loop that iterates over all of the different ways you could edit an existing set of rules, and then invoke the solver for each of these. So for example I invoke the solver telling it to change only the first rule, then I invoke it telling it to change only the second rule, etc, finally telling it to keep all the rules the same but add a new rule.

5.2 Search heuristics

5.2.1 Bounding the size of the rules

Sketch only works with finite program spaces, because it unrolls anything that loops or is recursive up to some depth bound. A necessary consequence of this is that we cannot learn rules with an arbitrary number of feature matrices. I bound the number of feature matrices to be 2 on each side.

5.2.2 "Freezing" rules and underlying forms

incrementalCEGIS constructs a sequence of models that explain more and more of the data. If in this sequence we see that a certain rule always occurs, or that a certain surface form always has a certain underlying representation, then we might guess that rule or underlying representation is really part of the correct solution and that we shouldn't go back and try to revise that rule or underlying form. Concretely what this means is that I keep track of how many times a rule or underlying form is found by **incrementalCEGIS**; once it has occurred at least 5 times I "freeze" the rule or underlying form and do not allow it to change in the future.
For example, after a few rounds of **incrementalCEGIS** we might find that we always need to devoice word final segments. If this is the case then we will stop trying to change the rule that devoices word final segments.

5.2.3 Guessing underlying forms

incrementalCEGIS is a generic way of learning large programs that explain large data sets, and does not encode any domain specific knowledge about phonology. Empirically, the search problem is *much* easier if you either know what the underlying forms are or have some constraints on what the underlying forms are. If I introspect on my own reasoning process while solving these problems, I usually guess at most a few plausible underlying forms for each lexeme. I encoded a simple heuristic for guessing parts of underlying representations: given prefixes/suffixes $\{p_i\}$ & $\{s_i\}$ and observed inflections for a single lexeme $\{x_i\}$, this heuristic

guesses that the underlying form begins with the longest shared prefix of the surface forms with the prefixes chopped off, and that it ends with the longest shared suffix of the surface forms with the suffixes chopped off. For example, if we have surface forms (test, testing, tests), and have prefixes (t, t, t) and suffixes (t, ting, ts), this heuristic constrains the underlying form to start with es and end with es.

6 Related work

Minimum generalization learner: [6]. The interesting similarity here is that we both learned the structure of the rules. But it's restriction to the supervised case hampers its ability to really capture what a grammar is about (a generative model learned without supervision) and it misses out on lots of cool phenomena.
Maximum entropy learner: [7]. Whether you take an OT perspective or a generative perspective, the structure of the rules (or constraints) has to come from somewhere. How do you learn that structure? This is a program induction problem.
Unsupervised phonological rule learners: [8] learns weighted fst's while [9] learns context-sensitive rewrites. Here we show that these results and more can be reformulated as a program induction problem, and that doing so buys you (1) connections to probabilistic programming and Bayesian program learning; (2) theoretical guarantees; (3) a bunch more phenomena modeled, like supervised cases (pig latin), UG, and infant studies.
References from a discussion with Adam Albright:
LISA (an acronym standing for something) babbling
Alex Cristia: Can infants learn phonology in the lab?
David Stampe: Spontaneous Devoicing. Evidence of UG
Chapter 8 of SPE [1]: What is the right evaluation metric for rules?
Collin Wilson: paletization generalizations as more evidence of UG in phonology
Degree of articulatory effort as an inductive bias
Rules shouldn't be too disruptive of the underlying form. Should also minimize confuseability. Final devoicing is a good example of this. PMAP. "weak contrast"

7 Experiments

7.1 Phonological Rule Induction

Several metrics of success here: (1) how well does it compress the data? (2) how well does it predict held out forms? (3) how natural or interpretable are the resulting solutions?
Here are cases where I think it learns something cool:

Language	Example data	Underlying forms	Phonology	Phenomena
Gen	sra agble drɛ hlɛ	l underlying	l→r/`[+coronal] _`	l∼r alternation

Language	Example data		Phonology	Morphology	Phenomena

Tibetan	1	ǰig	`[-nasal]`→ ∅ / `# _`
			or,
			`C`→ ∅ / `# _ C`
				given in problem	Consonant cluster reduction

	4	ši			
	5	ŋa			
	9	gu			
	10	ǰu			
	11 (= 10 + 1)	ǰugǰig			
	14 (= 10 + 4)	ǰubši			
	15 (= 10 + 5)	ǰuŋa			
	19 (= 10 + 9)	ǰurgu			
	40 (= 4 + 10)	šibǰu			
	50 (= 5 + 10)	ŋabǰu			
	90 (= 9 + 10)	gubǰu			

Language	Example data				Phonology	Morphology	Phenomen
Kerewe	kubala	kubalana	kubalila	kubalilana	`[]`→`[-H]`/`[+H] [] _` `[]`→`[+H]`/`[+H] [] _ []`	ku+stem+a ku+stem+ana ku+stem+ila ku+stem+ilana kutú+stem+a kukí+stem+a kutú+stem+ila kukítú+stem+ila	Tone spr Rule ord
	kugaya	kugayana	kugayila	kugayilana			
	kubála	kubálána	kubálíla	kubálílana			
	kutúbála	kukíbála	kutúbálila	kukítúbalila			
	kutúgáya	kukígáya	kutúgáyila	kukítúgayila			
	kutúbála	kukíbála	kutúbálila	kukítúbalila			

Language	Example data		Phonology	Morphology	Phenomena
Polish	dom	domi	o→u/_[-nasal +voice]#	stem	Final devoicing
	kot	koti	[-sonorant]→[-voice]/_#	stem+i	Rule ordering
	lut	lodi			
	vus	vozi			
	wuk	wugi			
	ruk	rogi			
	bur	bori			
	šum	šumi			

Language	Example data			Phonology	Morphology	Phenomena
Makonde	amáŋga	amíle	áma	[]→[-stress] / _ [ ]* [+stress]	stem+áŋga	Stress patterns
	akáŋga	akíle	áka	[+middle -stress]→a/ _ [ ]	stem+íle	Neutralizing rule
	aváŋga	avíle	óva		stem+a	
	amáŋga	amíle	óma			
	utáŋga	utíle	úta			
	aváŋga	avíle	éva			
	taváŋga	tavíle	táva			
	uŋgáŋga	uŋgíle	úŋga			
	patáŋga	patíle	póta			

Language	Example data		Phonology	Morphology	Phenomena
Kikuria	rema	remera	[+high]→[+middle]/ _ [-low]* [+middle]	stem+a	Vowel harmony
	roma	romera		stem+era	
	tiga	tegera			
	ruga	rogera			
	hoora	hoorera			
	siika	seekera			
	huuta	hootera			
	suraaŋga	suraaŋgera			

Language	Example data				Phonology	Morphology	Phenomena
Hungarian	a:g^{y}	a:g^{y}ban	a:k^{y}to:l	a:g^{y}nak	V→[+mid +tense +front]/ [+front] [ ]* _	stem	Vowel harmony
	ö:r	ö:rben	ö:rtö:l	ö:rnek	[ ]→[+voice]/ _ [+bilabial]	stem+ban	Voicing assimilation
	ku:t	ku:dban	ku:tto:l	ku:tnak	[-sonorant]→[-voice]/ _ [-voice]	stem+to:l	Rule ordering
	re:s	re:zben	re:stö:l	re:snek		stem+nak	
	rab	rabban	rapto:l	rabnak			
	vi:z	vi:zben	vi:stö:l	vi:znek			
	fal	falban	falto:l	falnak			
	test	tezdben	testtö:l	testnek			

These problems are solved:

- Kikurai: fricatives alternate with stops after nasals
- Greek: Velar stops alternate with palletized versions before front vowels
- Farsi: Trills alternate with flaps
- Osage: coronal stops become dentals before central vowels
- Amharic: alternation between ə & ɛ
- Gen: l/r alternation
- Kishambaa: voiced/unvoiced nasals alternate
- Thai: stops are unreleased word finally
- Palauan: a word initial neutralizing rule
- Quechua: Velar becomes uvular when followed by uvular (spreading)
- Lhasa Tibetan: no contrast between velar/uvular, or voiced/voiceless stops or fricatives
- Axininca Campa: stops become glides
- Kikuyu: infinitive prefix can surface as either k or ɣ
- Korean: vowel harmony, aspiration only surfaces in certain contexts
- Hungarian: vowel harmony and voicing assimilation
- Kikuria: vowel harmony
- Farsi: a deletion rule explains singular/plural
- Tibetan: initial consonant cluster reduction explains counting system
- Makonde: stress patterns; unstressed vowels are neutralized
- North Saami: 3 neutralization rules explain nominative sg essive
- Samoan: 2 deletion rules that explain words that sound the same in one inflection being different in another
- Russian: devoicing of word final obstruent
- English: verbal inflections (voicing assimilation, epenthesis)
- Finnish: nominative/partive explained by vowel raising and vowel harmony
- Kerewe: Interacting tone rules
- Polish: vowel alternations interacting with devoicing of word final obstruent in singular/plural
- Ancient Greek: voicing assimilation, deaspiration, deletion rule. Order matters.
- Serbo-Croatian: predictable stress, devoicing, neutralizing, epenthesis
- Catalan: Spritization, devoicing, several different kinds of cluster reduction
- Korean: place assimilation and nasalization
- Latin: assimilation, devoicing, different kinds of cluster reduction, vowel harmony

These problems it solves most of:

- Somali: Spritization, copying rules

12 textbook problems are not solved, along with two other problems that involve tier–based representations. 3 of the above problems are not actually exercises but are instead examples that I included because they are neat problems. This means that we solve $\frac{28}{40} = 70\%$ of the textbook currently, using a slightly liberal interpretation of what it means to "solve a problem" (e.g., our solution for Latin is slightly different from the textbook answer, but to me it looks fine and is at least as compressive).

7.1.1 An approximation to UG: Program fragments

Within-rule reuse of structure:

Tibetan: C→ ∅/#_C

English: [-sonorant]→[-voice]/[-voice] _

Kikuria: [+high]→ [+middle] / _ [-low]* [+middle]

Within-language reuse of structure:

Axininca Campa:
p→w / [ ] _
k→y / [ ] _

English:
∅ →ə / [+sib] _ [+sib]
∅ →ə / [+cor +stop] _ [+cor +stop]

Cross-language reuse of structure:

Russian/Polish: [-sonorant]→[-voice]/_#

Samoan/Tibetan:
C→ ∅/_#
C→ ∅/#_C

Ancient Greek/Hungarian:
[-sonorant]→[-voice -aspirated] / _ [-voice]
[-sonorant]→[-voice] / _ [-voice]

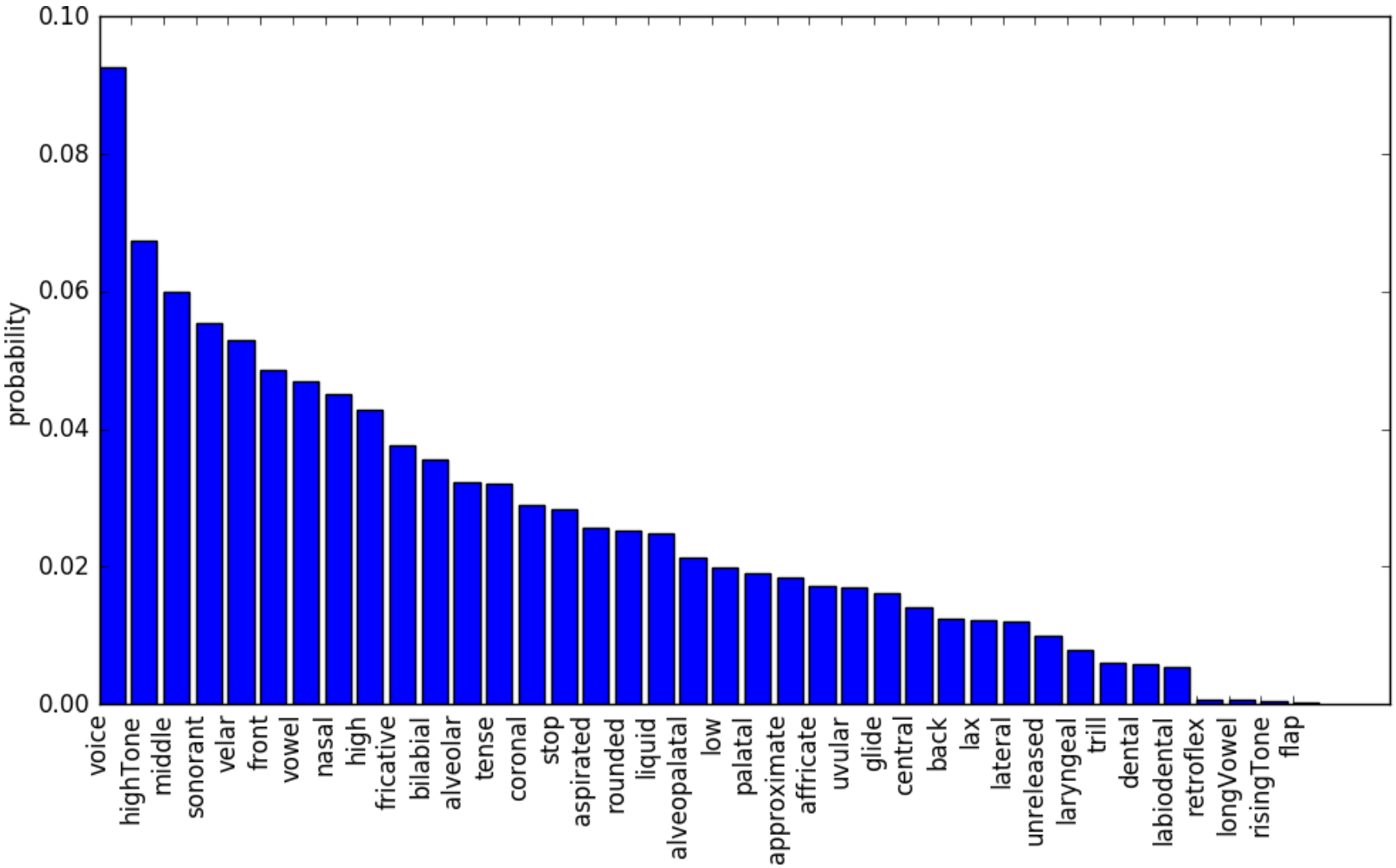

Figure 4: Which features are frequently used in correct rules? Voicing, sonorant, nasality are very frequent, and are also near the top of the feature geometry hierarchy. Feature geometry was not given to the system, but you can see at least this coarse level of structure emerge from the data. Vowel features also frequent due to vowel harmony occurring in lots of languages.

7.2 Artificial Phonologies

It would be cool to model all of the grammars in [4]. Right now we can model AAB/ABA patterns like in Gary Marcus's famous infant experiments. Other infant studies with artificial grammars: [10]. MNPQ problem: [11].

Analogs in real languages: agreement morphology (eg gender in Spanish - there is a constraint that two things have to be the same); reduplication (this could literally be modeled using the same ABA system); obligatory contour principle (opposite of harmony - two things have to be different, just like the ssd/sds distinction).

For any of these problems, there is a trade-off between how complicated of a program you are willing to use and how badly you want to compress the data. Here, compressing the data means representing the surface forms with short underlying representations. This trade-off is of course present in the phonology problem sets as well. But the problem sets are complicated and have lots of

data, precluding any in-depth analysis of the trade-off except quantitatively. For these synthetic grammar learning problems, we can dig in and see exactly what the trade-off looks like for different amounts of data. See figure **??** for an analysis of this trade-off along with examples of what the programs look like.

8 Applications

The point of this is to motivate why you would want to solve exactly this problem even though it doesn't do anything more immediately useful than solve your phonology homework. Practical applications that could come from this line of work:

- Low-resource language acquisition: `http://sapience.dec.ens.fr/bootphon/` for some languages there is very little data and very few speakers
- Mathematica for Phonology: a lot of the time physicists don't do integrals; they just punch them into Mathematica. This is a tool that provably finds the "best" grammar for some data, sort of like how Mathematica finds the correct integral of some expression. And just like some integrals are beyond the reach of computer algebra systems, some data sets might be beyond the reach of this hypothetical tool. Analogy to the work in [12]. According to Tim it would be more practical for the field for knowledge is to have a system that can work out the consequences of a given rule set, maybe pointing out places where it's inconsistent with the data or suggesting underlying forms. These alternative uses could also be accommodated.
- A probe for UG: the prior over phonology is a parameter of the algorithm which you could in principle estimate from the data or compare different settings of the prior to see how well they compress the data. Or, come up with the right constraints so that the system doesn't generalize in the same places where humans don't generalize (surfeit of the stimulus: [13]). You can approach universal grammar from two directions: what do I need to build then so that it will learn the same things that humans learn? And what do I need to take out in order to stop it from learning things that humans never learn?

9 Discussion

Deep learnability issue: what is the actual computational complexity of learning these grammars? The actual hardness is an empirical question which you can upper bound by coming up with better heuristics for inducing grammars. Attempts at *lower bounding* the complexity of learning has generally been way too conservative. We want to know the actual hardness of the learning problem, and constructing a practical program learner shows that it might not be too hard with the right UG.
We must emphasize that there is **no general-purpose learner**. So we need UG. This tool lets us dig in and study it. There's always going to be a trade-off between how much you build in and how much data/computation is needed to recover the correct generalizations. Our compromise in the space of this trade-off is to:

- Not play in the space of arbitrary turing machines but instead search through a finite program space (bounded program size runtime and memory consumption; program structure given by a grammar)
- Make contentfull proposals about what the structure of universal grammar might be, and test them empirically on many languages

Comparison with deep learning: In a sense this project is like the opposite of deep learning. It's about the kind of learning that yields explanations of the causal processes behind data sets; where the learner comes equipped with explicit background knowledge (UG) that we can tweak finely and systematically.
One similarity though is that some of the phenomena we model, like infant learning of synthetic grammars, are the sort of things that humans pick up unconsciously. A lot of deep learning might be analogous to humans rote learning some motor or perceptual task.

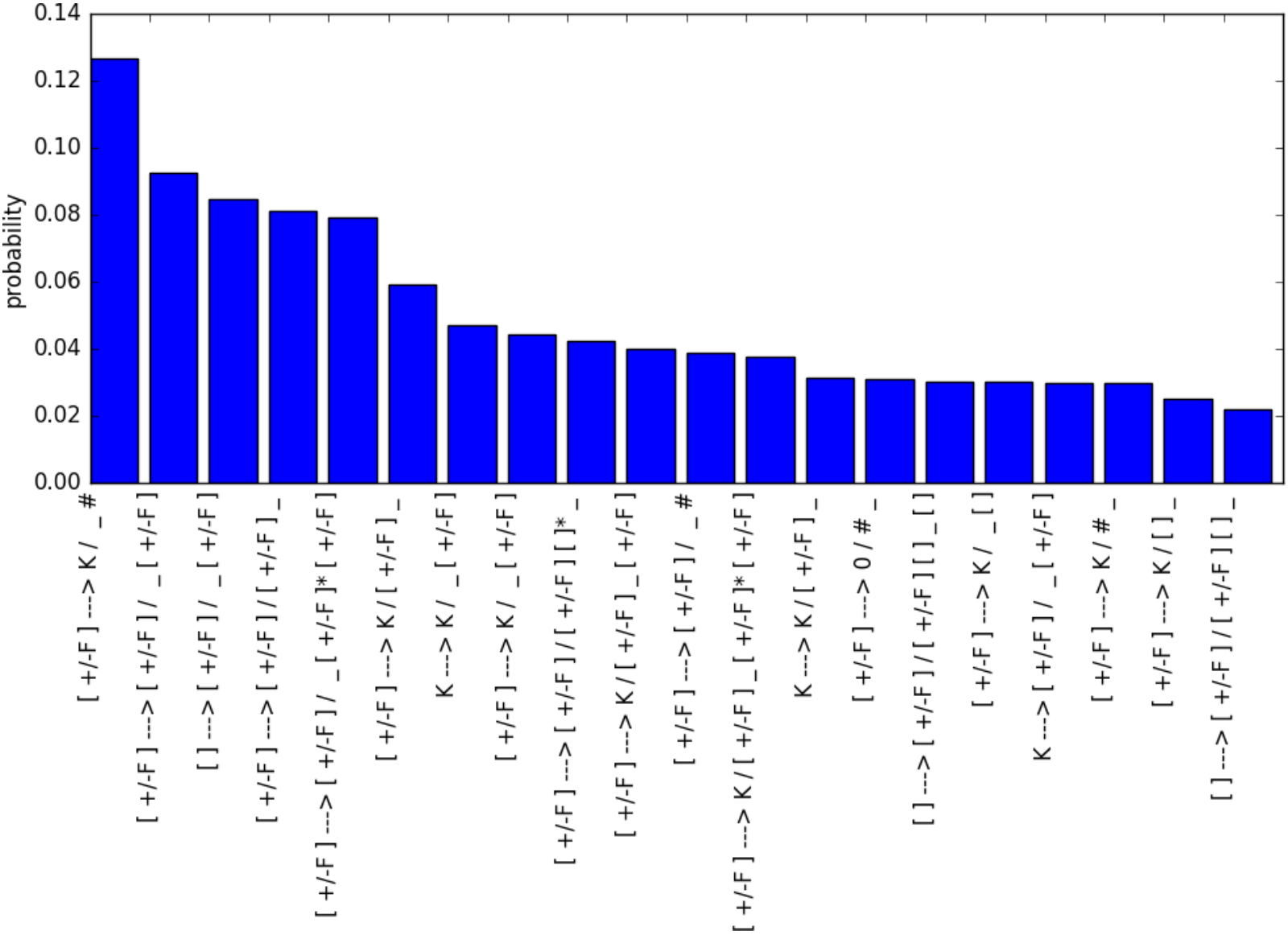

Figure 5: What is the general structure of rules that occur in phonology, ignoring the particular constants that occur in the rule? Here K stands in for an arbitrary phoneme; ±features stands in for an arbitrary feature matrix. From this plot you can glean that (1) neutralizing rules are very common (top two most frequent rules); (2) harmony/assimilation rules are also frequent; (4) deletion rules tend to occur at word boundaries. There are about 5×10^4 possible syntactic structures for rules, but only 15 are actually attested (relative frequency> 1). Here I think that the sparsity of the problem sets is hurting us: these results come from 26 problems, so the fact that you see only 15 general schemas isn't too crazy if you don't believe in strong innate constraints on the structure of rules. But looking at this data you also see a generalization that holds across all of the rules learned: the structures tend to be pretty small, so you never see a rule that checks feature values of more than two adjacent segments.