Meta-Learning

Lake 2019 & McCoy et al. 2020

By Joe O'Connor, Abby Bertics, and Ferran Alet

Timeline

Introduction

Lake 2019

Compositional generalization through meta sequence-to-sequence learning

Discussion

Breakout rooms + group

Break

McCoy et al. 2020

Universal linguistic inductive biases via meta-learning

Discussion

Interspersed

Conclusion

Meta-learning: a 2-slide overview

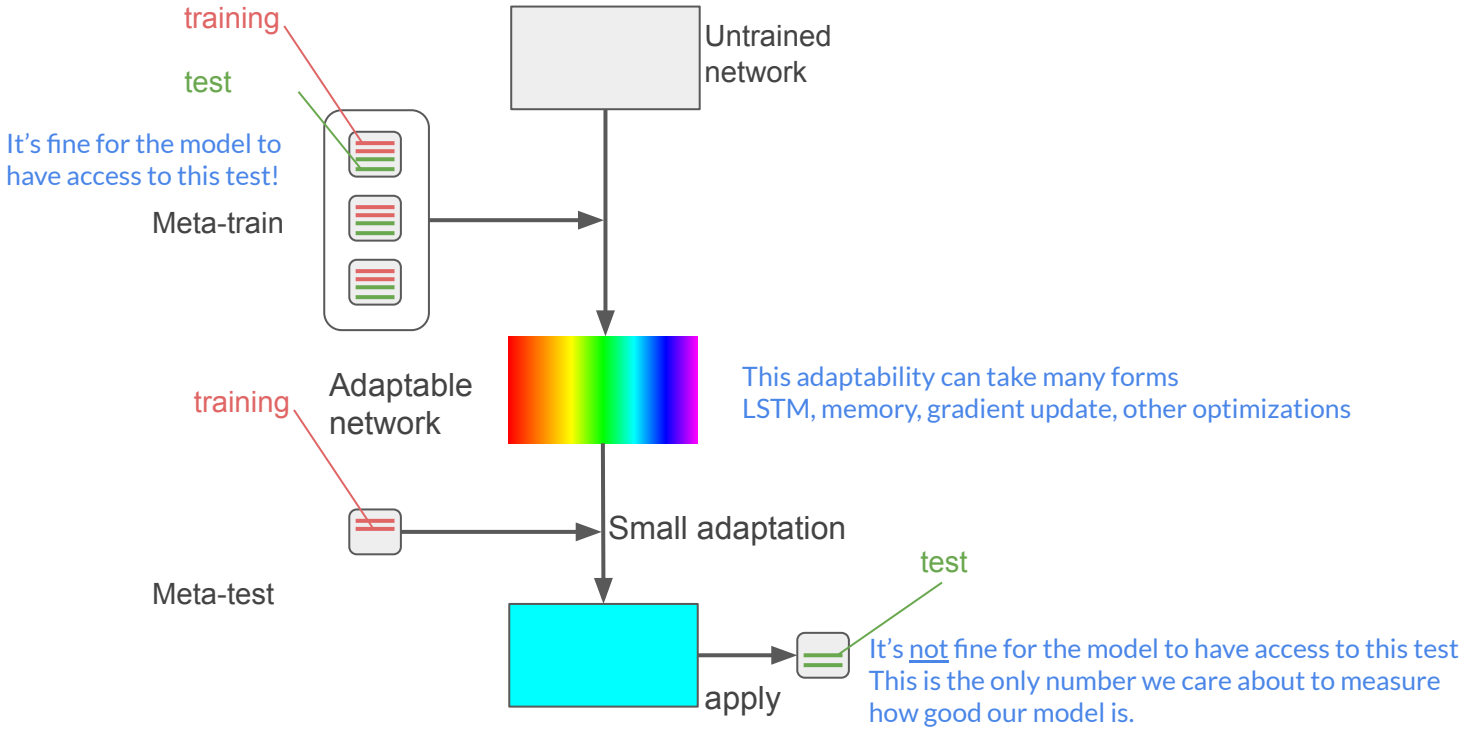
Leveraging related tasks, either in terms of data or computations

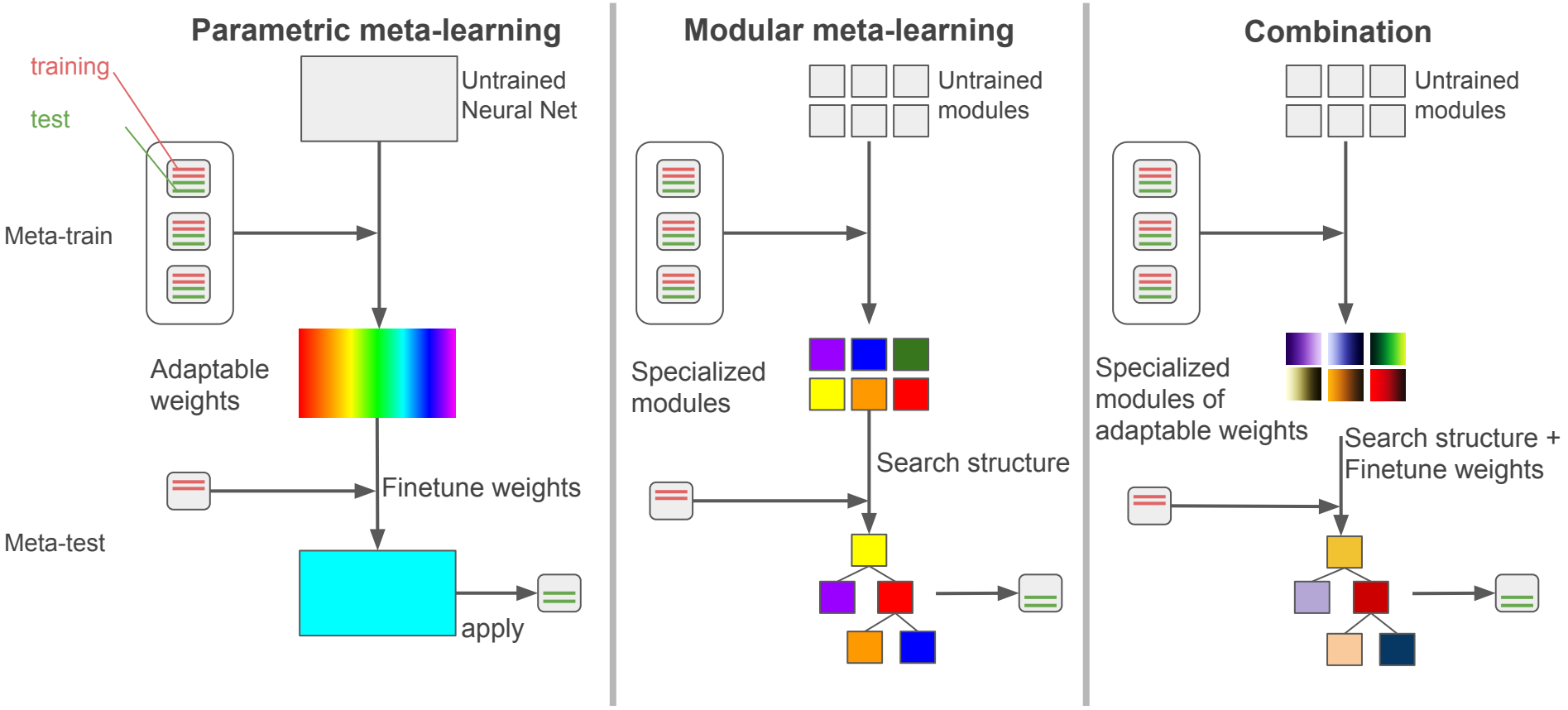
- Learning to learn from few examples (few-shot learning)
- Learning to optimize
- AutoML, architecture search, meta-learning new algorithms
- ...

Two views of meta-learning:

- Mechanistic view: [more useful for 1st paper]
  - Deep Network that reads an entire dataset and then makes predictions for new datapoints
  - Dataset→ datapoint; therefore we now have meta-dataset of datasets
- Probabilistic view: [more useful for 2nd paper]
  - Extract prior from a set of (meta-training) tasks that allows efficient learning of new tasks
  - A new task uses this prior plus small training set to infer most likely parameters

Setting


training
test
It's fine for the model to have access to this test!
Meta-train
Untrained network
Adaptable network
This adaptability can take many forms
LSTM, memory, gradient update, other optimizations
training
Meta-test
Small adaptation
test
apply
It's not fine for the model to have access to this test
This is the only number we care about to measure how good our model is.


Parametric meta-learning
training
test
Untrained
Neural Net
Meta-train
Adaptable
weights
Finetune weights
Meta-test
apply
Modular meta-learning
Untrained
modules
Specialized
modules
Search structure
Combination
Untrained
modules
Specialized
modules of
adaptable weights
Search structure +
Finetune weights

Compositional generalization through meta sequence-to-sequence learning

Lake 2019

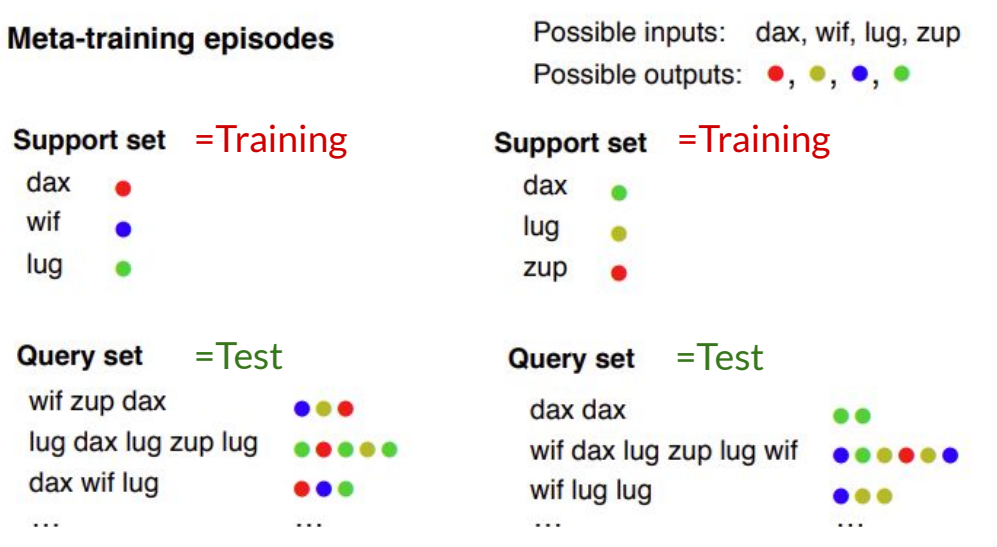
Presented by Ferran Alet and Joe O'Connor

TLDR for Lake

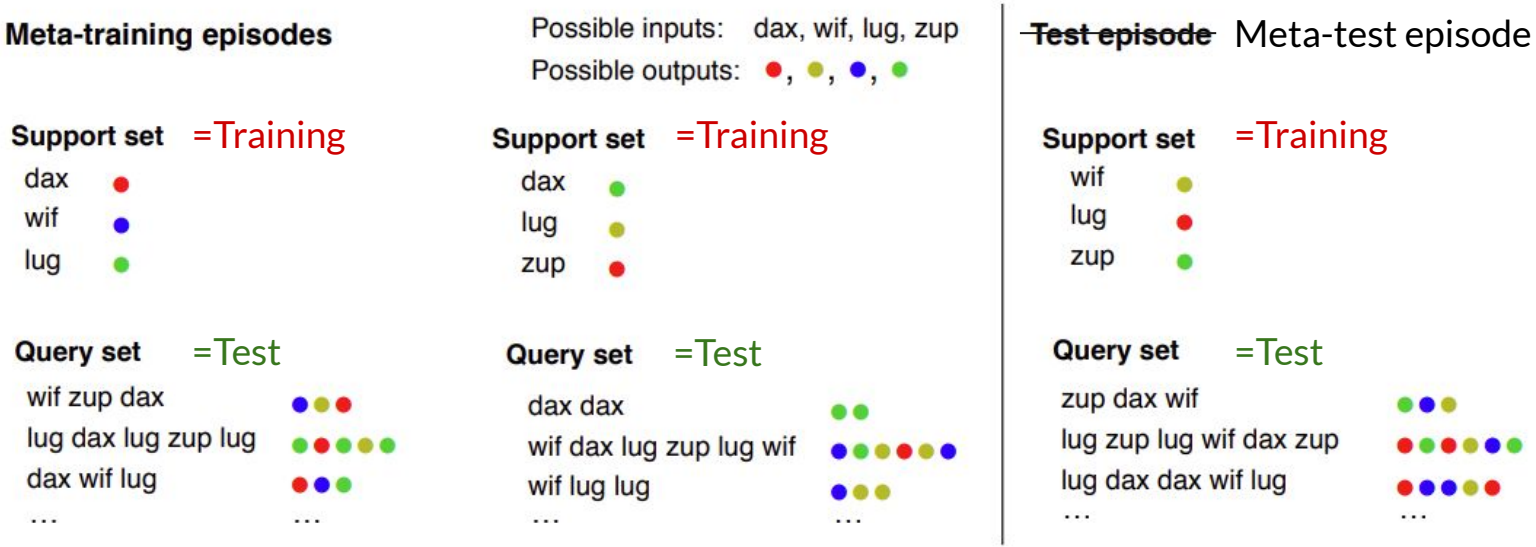
Solving meta-seq2seq: learning to solve sequence-to-sequence tasks from small amounts of data

with memory-augmented neural networks:networks that can probe learned soft dictionaries that encode previous inputs

Dataset 1:


Meta-training episodes
Support set =Training
dax
wif
lug
Query set =Test
wif zup dax
lug dax lug zup lug
dax wif lug
…
Possible inputs: dax, wif, lug, zup
Possible outputs:
Support set =Training
dax
lug
zup
Query set =Test
dax dax
wif dax lug zup lug wif
wif lug lug
…

Dataset 1:



Meta-learning version: 4! Assignments of 4 words to 4 colors

Dataset 2: SCAN; meta-learning augmentations

Table 1: SCAN task for compositional learning with input instructions (left) and their output actions (right) [16].

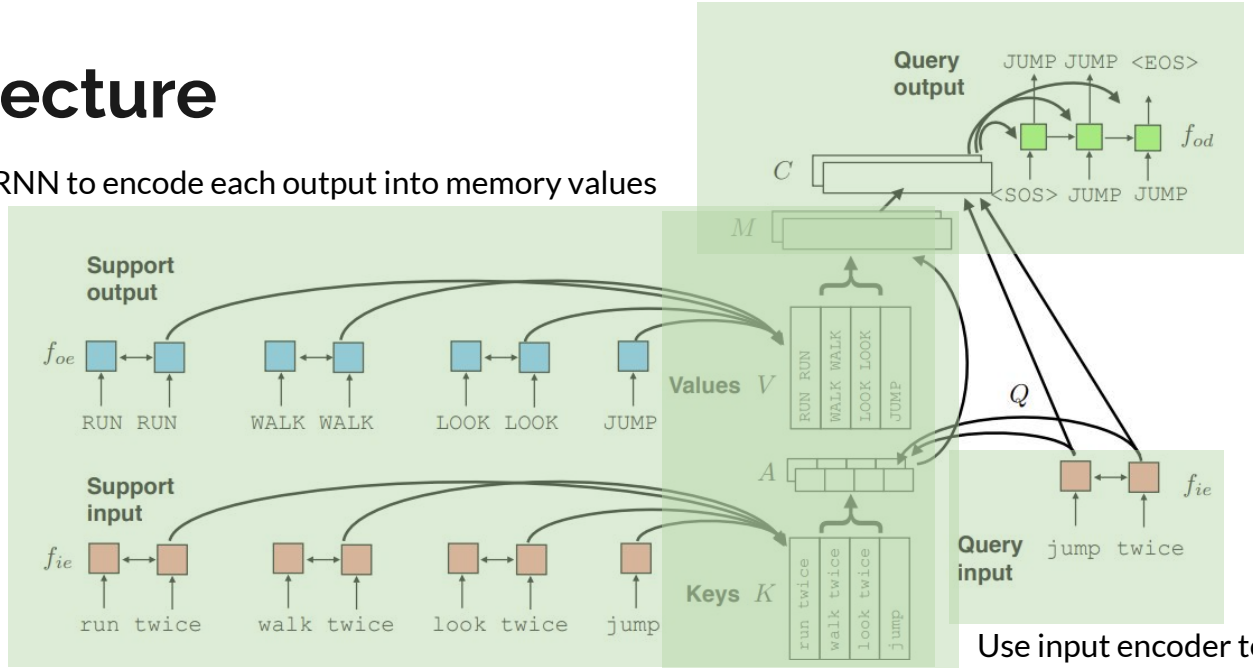
jump	⇒	JUMP
jump left	⇒	LTURN JUMP
jump around right	⇒	RTURN JUMP RTURN JUMP RTURN JUMP RTURN JUMP
turn left twice	⇒	LTURN LTURN
jump thrice	⇒	JUMP JUMP JUMP
jump opposite left and walk thrice	⇒	LTURN LTURN JUMP WALK WALK WALK
jump opposite left after walk around left	⇒	LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN LTURN JUMP

We meta-train on 4!-1=23 variations of SCAN by mapping (‘jump’, ‘run’, ‘walk’, ‘look’) a permutation of the correct meanings (JUMP, RUN, WALK, LOOK) and test on the unseen identity permutation

- Is this cheating a bit? → Would we have similar (meta-)data on real tasks?

Architecture

Use different RNN to encode each output into memory values



Encode RNN to encode each input into memory keys

Use decoder from retrieved context to decode output

- Decoder has attention to context at every step

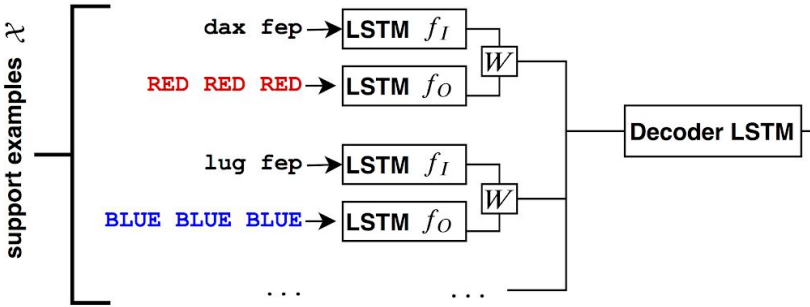
Use input encoder to create key to probe memory

Memory as soft dictionary

- Use queries and keys to get attention over slots
- Use attention to get weighted-average value for every key

Program Synthesis Approach to SCAN
(Nye, Solar-Lezama, Tenenbaum, Lake)

Given examples...



our system infers a program...

Synthesized grammar G:

```
G =
dax -> RED
lug -> BLUE
zup -> YELLOW
wif -> GREEN
u2 fep -> [u2] [u2] [u2]
x2 kiki x1 -> [x1] [x2]
u1 blicket u2 -> [u1] [u2] [u1]
u1 x1 -> [u1] [x1]
```

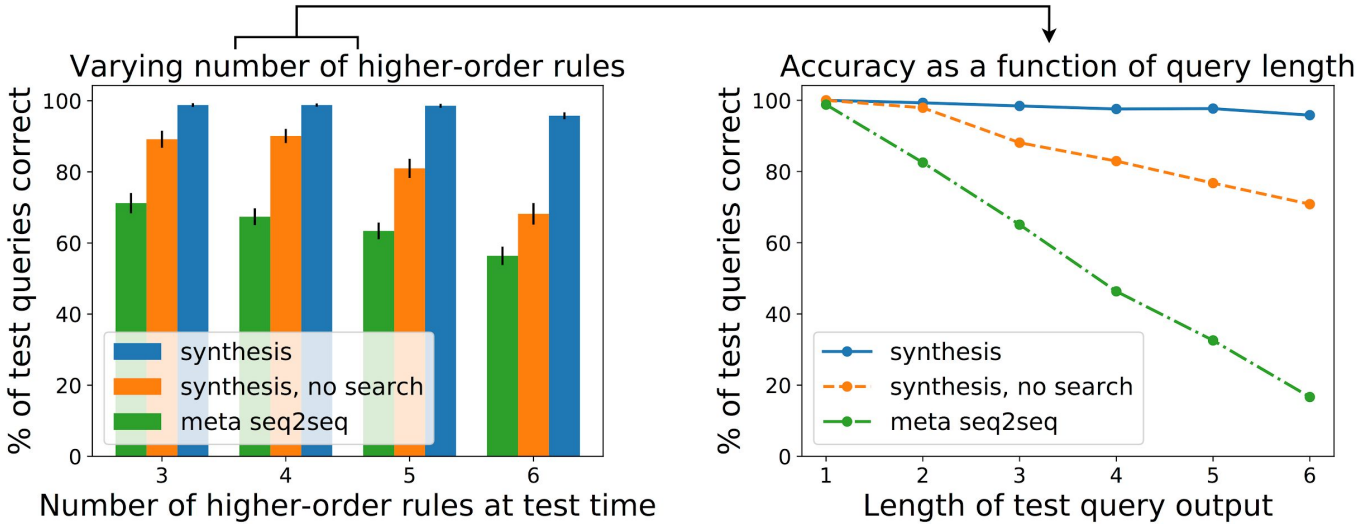
which can be applied to held-out examples:

```
G.apply(`zup fep`)

= [zup][zup][zup]

=
```

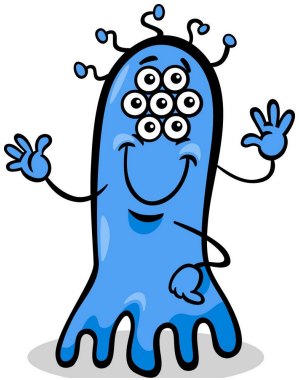
Programs naturally scale to longer outputs



Experiment 1: Mutual exclusivity

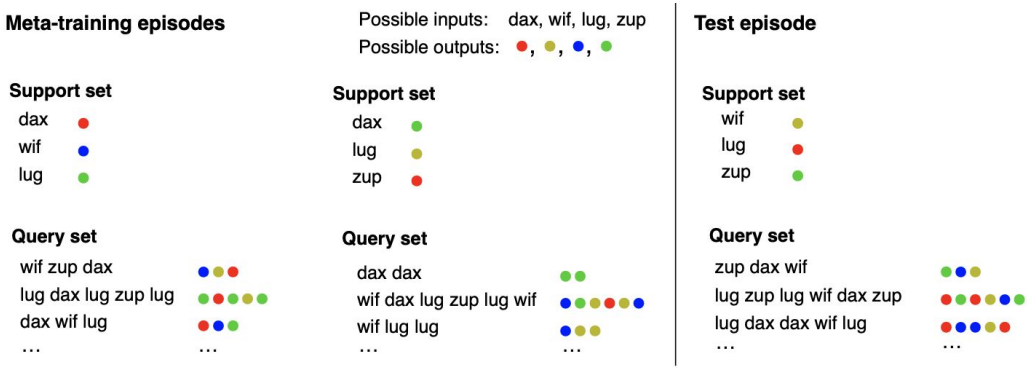
- **Motivation:** children use mutual exclusivity to help learn the meaning of new words, and adults use ME to resolve ambiguity in laboratory tasks on artificial language
- E.g., Which one is the *dax*?

Hm… well this one is definitely a cup...



… and I've never seen anything like this before

Setup & results



- Training
  - Each episode is random permutation of mapping from inputs to outputs
  - Three mappings given in support set, must recover the fourth from the query set
- Testing
  - Meta seq2seq achieves 100% accuracy
  - Can acquire new mappings without updating parameters
  - Can reason about the absence of symbols in memory

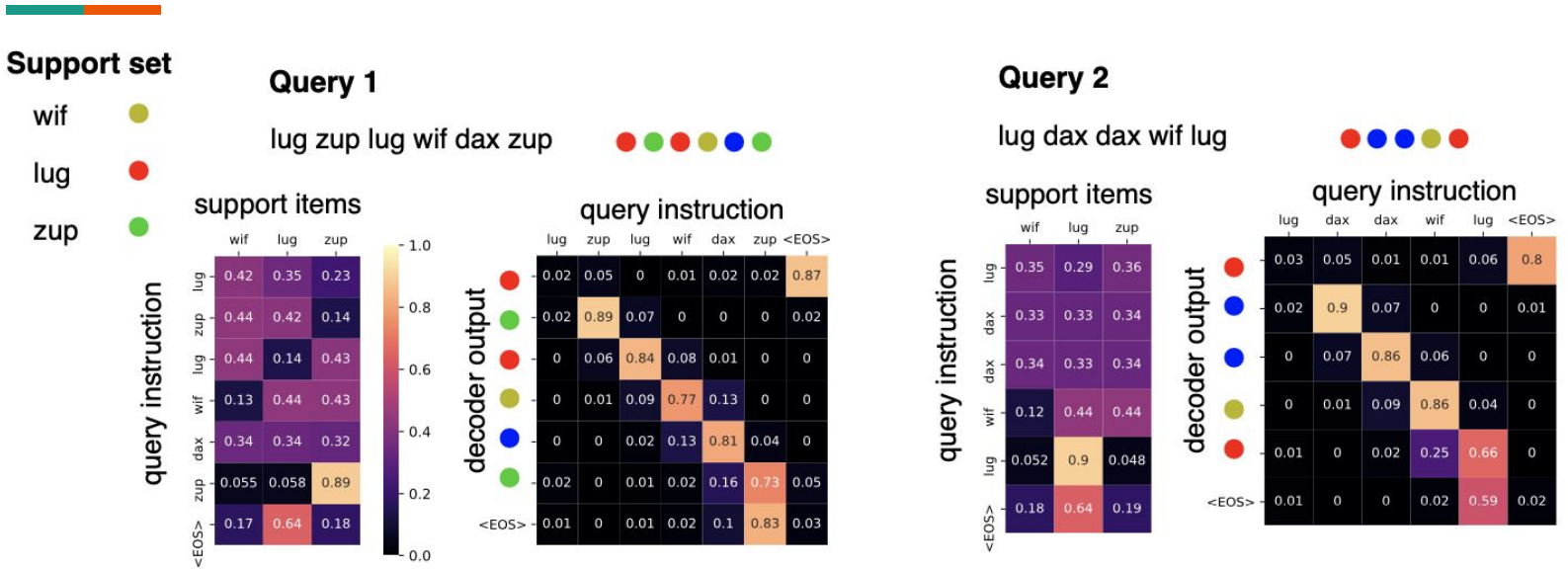


Figure A.1: During a test episode of the ME task, the support set (top left) and two queries are shown. The ME inference is that "dax" maps to "blue." The key-value memory attention A for each query is shown in the left matrix, with rows as encoder steps and columns as support items. The decoder attention for each query is shown in the right matrix, with rows as the decoder steps and columns as encoder steps. <EOS> marks the end-of-sequence.

Experiment 2: Adding a new primitive through permutation meta-training

- Want to check whether a model can use a new primitive compositionally
- E.g., if you know how to *doomscroll*, then you know how to *anxiously doomscroll for hours while drinking wine on a Tuesday night in November*

jump	⇒	JUMP
jump left	⇒	LTURN JUMP
jump around right	⇒	RTURN JUMP RTURN JUMP RTURN JUMP RTURN JUMP
turn left twice	⇒	LTURN LTURN
jump thrice	⇒	JUMP JUMP JUMP
jump opposite left and walk thrice	⇒	LTURN LTURN JUMP WALK WALK WALK
jump opposite left after walk around left	⇒	LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN LTURN JUMP

Table 1: SCAN task for compositional learning with input instructions (left) and their output actions (right) [16].

Setup

- **Standard seq2seq training**
  - Exposed to *jump* in isolation as well as every primitive and composed instructions for the other actions
    - ~13,000 instructions
  - E.g., taught how to *jump*, *walk twice*, *look around right*, but not *look around right and jump twice*
- **Standard seq2seq testing**
  - Evaluated on all ~7,000 composed instructions that contain *jump*
- **Meta seq2seq training**
  - Each episode is generated by sampling a random mapping from primitive instructions to primitive actions
    - Never see the “correct” mapping
  - 20 support instructions and 20 query instructions per episode
- **Meta seq2seq testing**
  - Support set is correct mapping from primitive instructions to primitive actions
  - Evaluated on all composed *jump* instructions
- **Meta seq2seq ablations:** one with no support loss, one with no decoder attention

Results

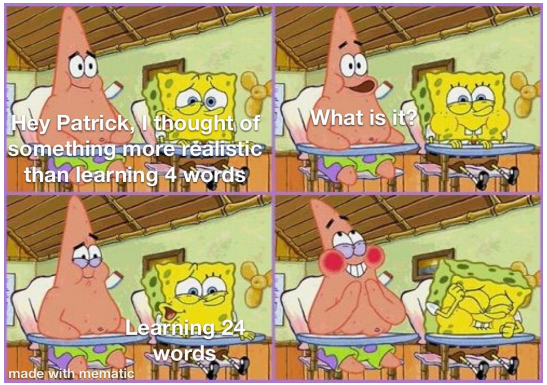
- **Claim:** network learns how to compose
- **Claim:** network learns to store and retrieve variables from memory with arbitrary assignments
  - (as long as it has seen the whole input space and whole outputs space)

Model	standard training	permutation meta-training
meta seq2seq learning	—	**99.95%**
-without support loss	—	5.43%
-without decoder attention	—	10.32%
standard seq2seq	0.03%	—
syntactic attention [30]	78.4%	—

Table 2: Test accuracy on the SCAN "add jump" task across different training paradigms.

Experiment 3: Adding a new primitive through augmentation meta-training

- Hey that last thing was pretty cool, but the model only had to learn 4 words
- Let's do something much more realistic and make it learn… 24 words
    - Add *Primitive1*, *Primitive2*, ..., *Primitive20* and *Action1*, *Action2*, …, *Action20*



Setup

- **Standard seq2seq training**
  - Exactly analogous to the previous experiment but with the extra primitives/actions
- **Standard seq2seq testing**
  - Exactly the same as the previous experiment (no extra primitives/actions)
- **Meta seq2seq training**
  - Each episode is generated by sampling 4 primitive instructions (out of all 24) and sampling 4 primitive actions (out of all 24), with the mappings also randomly defined
  - Never see *jump* mapped to JUMP
- **Meta seq2seq testing**
  - Exactly the same as the previous experiment (no extra primitives/actions)
- **Meta seq2seq ablations:** same as previous experiment

Results

- Interesting that when the task got more “complex” it also got… easier
- No support loss does better than before because of increased pressure to use the memory

Model	standard training	permutation meta-training	augmentation meta-training
meta seq2seq learning	—	**99.95%**	**98.71%**
-without support loss	—	5.43%	**99.48%**
-without decoder attention	—	10.32%	9.29%
standard seq2seq	0.03%	—	12.26%
syntactic attention [30]	78.4%	—	—

Table 2: Test accuracy on the SCAN “add jump” task across different training paradigms.

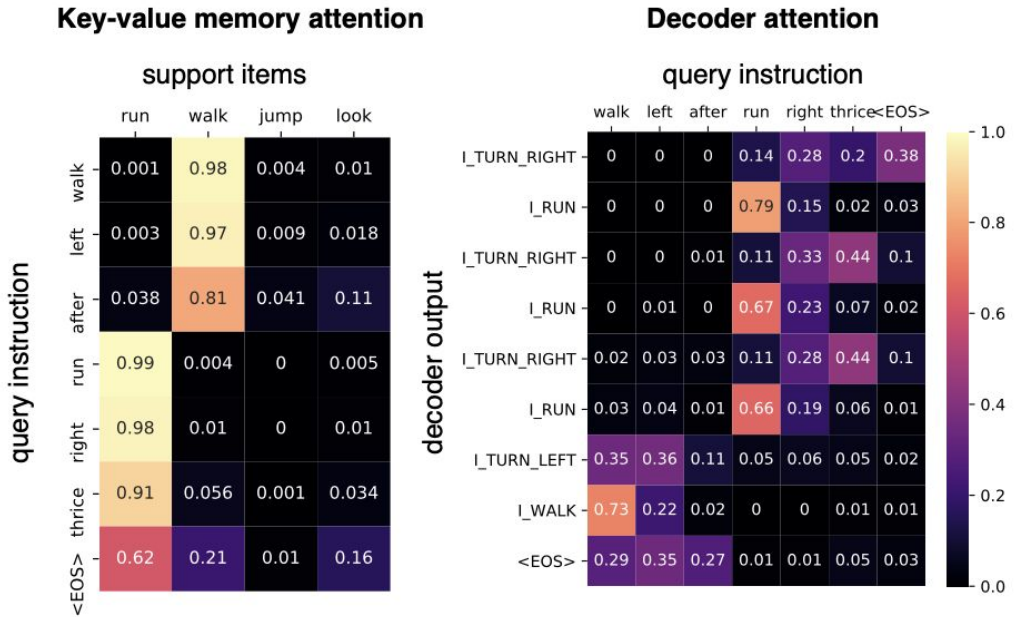


Figure A.2: Attention in meta seq2seq learning on the SCAN task. During test, the network is evaluated on the query “walk left after run right thrice.” <EOS> marks the end-of-sequence.

Experiment 4: Combining familiar concepts

- **My interpretation:** if you know how to do *X*, *Y*, and *YZ*, and you know that *X* and *Z* are used in essentially the same way, you should know how to do *YX*
- E.g., if you know how to *jump right*, *jump left*, and *jump around left*, then you should be able to use the relationship between *left* and *right* to figure out how to *jump around right*

Setup & results

Model	around right
meta seq2seq learning	**99.96%**
standard seq2seq	0.0%
syntactic attention [30]	28.9%

- **Standard seq2seq training**
  - All instructions except those including *around right*
- **Standard seq2seq testing**
  - All instructions that include *around right*
- **Meta seq2seq training**
  - Include *forward* and *backward* primitives and FORWARD and BACKWARD actions
  - Each episode is generated by sampling a random mapping of two direction primitives to two direction actions
  - Never see *right* map to RTURN
- **Meta seq2seq testing**
  - Support set is mapping from *turn left* and *turn right* to their correct meanings
  - Evaluated on all instructions that include *around right*

Experiment 5: Generalizing to longer instructions

- Now that we've proved beyond a shadow of a doubt that the model is capable of mastering compositional skills and variable manipulation, it should have no problem figuring out the meaning of sequences with a few more required actions, right?

Setup

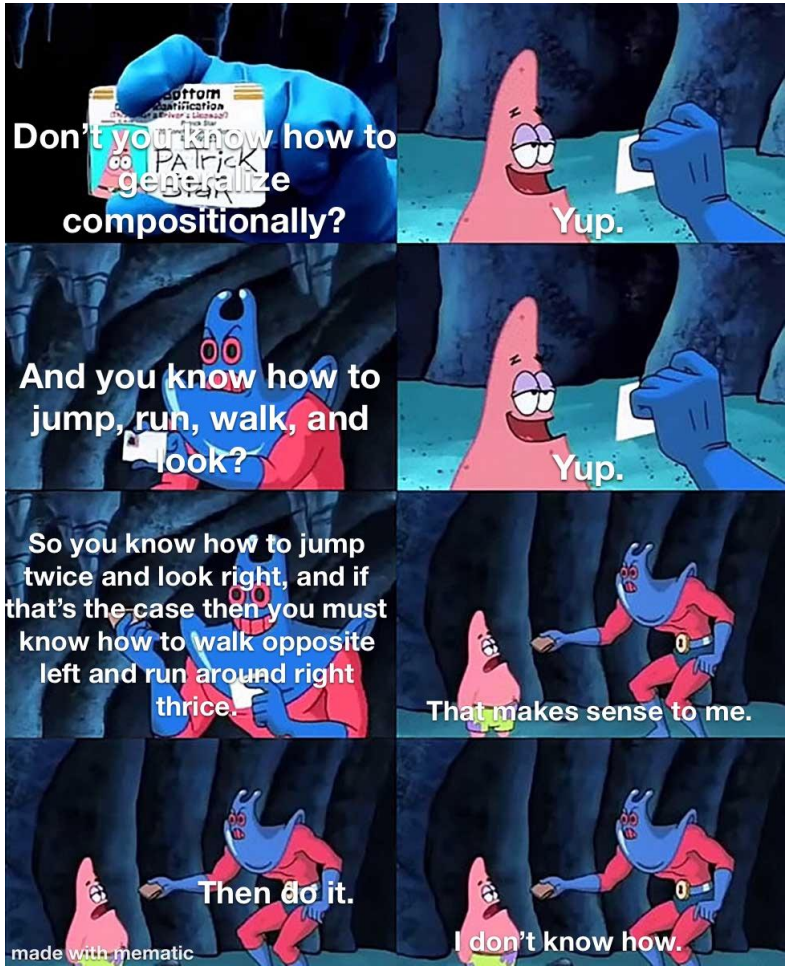
- **Standard seq2seq training**
  - All instructions that require 22 or fewer actions (~17,000)
- **Standard seq2seq testing**
  - All instructions that require 24-28 actions (~4,000)
  - E.g., have seen *jump around right twice* as well as *look opposite right thrice*, but now needs to *jump around right twice and look opposite right thrice*
- **Meta seq2seq training**
  - Support items are instructions with less than 12 actions and query items are instructions with 12-22 actions
  - Each episode has 100 support items and 20 query items
  - The extra primitives and actions are also included
- **Meta seq2seq testing**
  - Support of 100 instruction/action sequences with at most 22 actions
  - Evaluated on all instructions that require 24-28 actions

Results

- How can we explain this?

Model	around right	length
meta seq2seq learning	**99.96%**	16.64%
standard seq2seq	0.0%	7.71%
syntactic attention [30]	28.9%	15.2%

Table 3: Test accuracy on the SCAN "around right" and "length" tasks.



Me: mom can we stop and get some compositional generalization
Mom: no we have compositional generalization at home
The compositional generalization at home:

Model	around right	length
meta seq2seq learning	**99.96%**	16.64%
standard seq2seq	0.0%	7.71%
syntactic attention [30]	28.9%	15.2%

Meta seq2seq discussion questions

- Lake acknowledges the model's ability to use "variables" is not *exactly* the kind of thing classicists insist is necessary and unattainable via connectionist models, but how close is it? Would some extra symbolic machinery get it the rest of the way there, as he suggests it would?
- In the test stage of the mutual exclusivity experiment, the model gets a support set of three mappings and must learn the fourth mapping. Assuming the query set was such that the mappings where still uniquely determined, what if it got two and had to learn two? One and three? Zero and four?
- Is this meta-learning approach cheating a bit? → Would we have similar (meta-)data on real tasks?
- What would happen if we fed the support set and the query into a fine-tuned GPT-3?
- How robust are these methods to exceptions?

Break

Universal linguistic inductive biases via meta-learning

McCoy et al. 2020

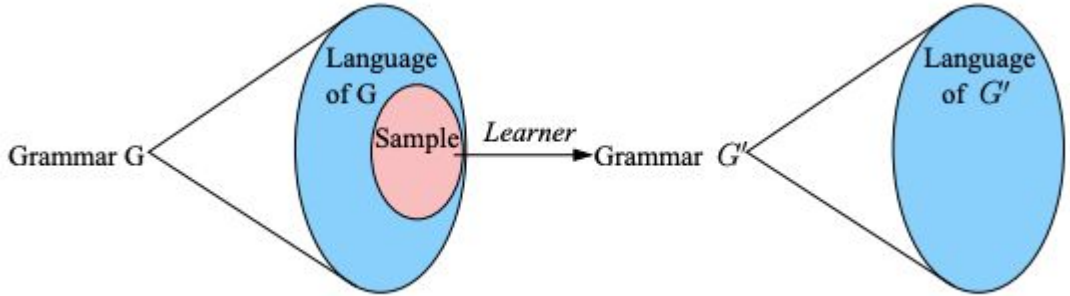
Presented by Abby Bertics

General Paper Claim

- Introduce framework to give particular linguistic inductive biases to a neural network model

Motivation:

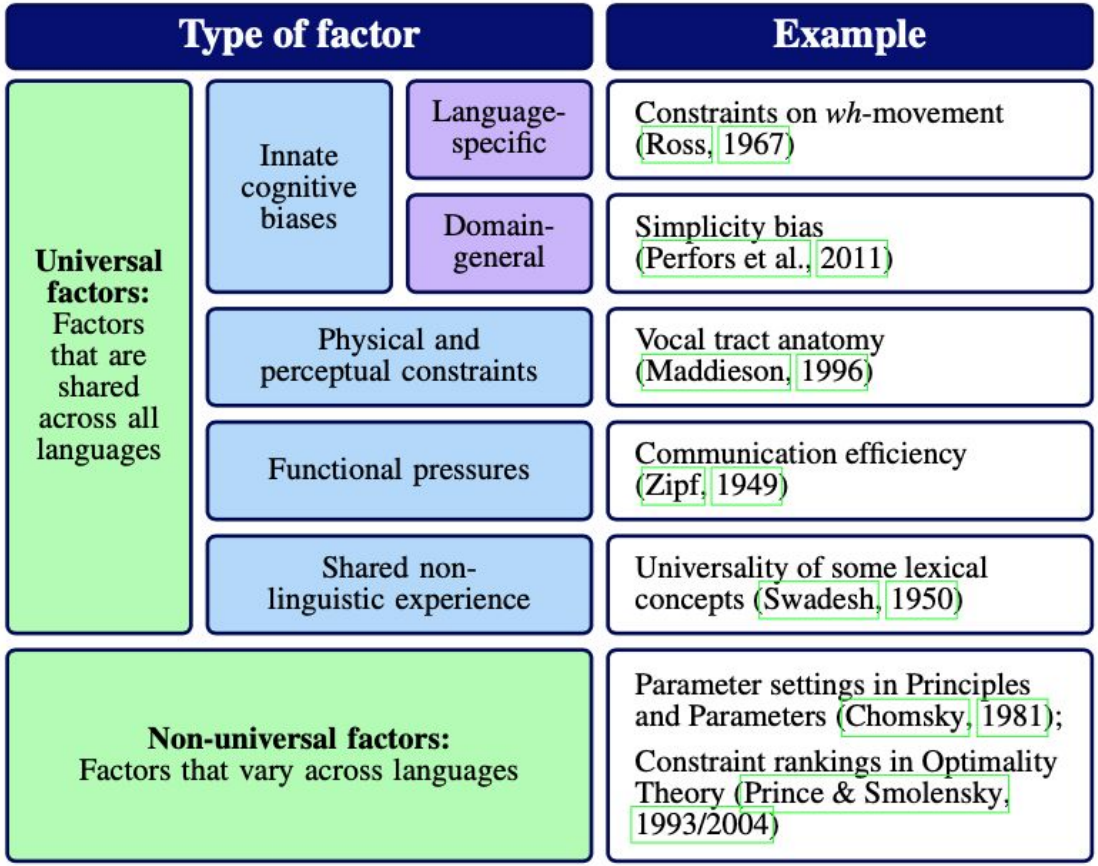
- Near impossibility of language acquisition
  - Poverty of the Stimulus / Data Sparsity Problem
  - Data + *inductive biases*



Quick Question

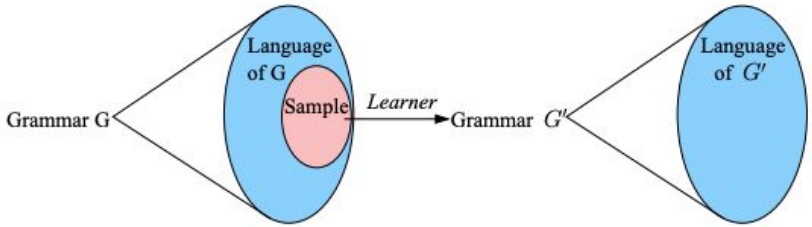
`Flood the chat`

What biases might be useful and/or necessary for a language learner?


Type of factor
Example
Universal factors: Factors that are shared across all languages
Innate cognitive biases
Language-specific
Constraints on *wh*-movement (Ross, 1967)
Domain-general
Simplicity bias (Perfors et al., 2011)
Physical and perceptual constraints
Vocal tract anatomy (Maddieson, 1996)
Functional pressures
Communication efficiency (Zipf, 1949)
Shared non-linguistic experience
Universality of some lexical concepts (Swadesh, 1950)
Non-universal factors: Factors that vary across languages
Parameter settings in Principles and Parameters (Chomsky, 1981);
Constraint rankings in Optimality Theory (Prince & Smolensky, 1993/2004)

Less Quick Questions

1. Which learners are sure to discover a grammar G′ such that the language of G is the same as the language of G′?
2. Which learners can do this for samples drawn from any language which belong to some class of languages?
3. What kind of sample does the learner need to succeed in this way?



The Role of Inductive Biases

- Patterns found in natural language are not arbitrary
  - Grammars which generate these patterns are ultimately constrained in some fashion
- *Universal Grammar*

- Inductive biases constrain the hypothesis search space

Solution: Meta-Learning!

“meta-learning is a very powerful approach for endowing artificial systems with useful inductive biases”

Human learning: Given biases, learn (any) language.
Meta-Learning: Given possible languages, learn biases.

Shifting need for structure from model to data.

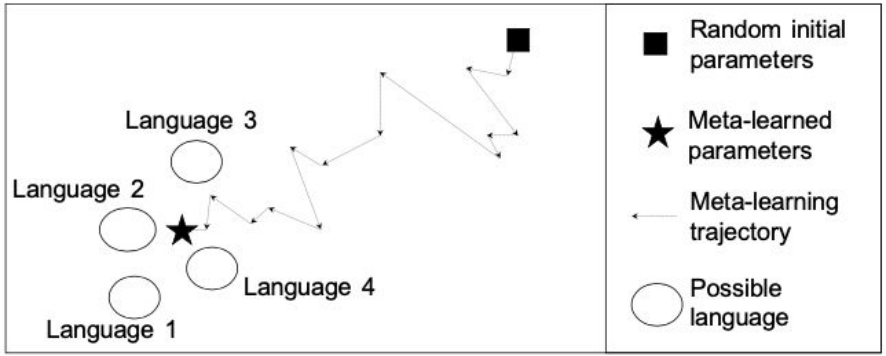
Overview: Model-Agnostic Meta-Learning

Standard Training:

- Minimize error within a single training language

Meta-Training:

- perform well on unseen examples after few steps of training



(Slightly) More Formally

Given: $L = \{L_0, L_1, ..., L_n\}$, $p(L)$, M_0

At step i:

- Select language $\mathbf{L_i}$ from distribution $p(L)$
- Standard goal:
  - learn L_i given initial parameters M_0
  - output: trained model M_i
- Meta-Goal:
  - tweak M_0 using M_i's loss on unseen test examples
  - tweak M_0 s.t. It is easier to learn L_i the next time around

Case Study: Syl·lab·i·fi·ca·tion

- Optimality Theory (Prince and Smolensky 1993/2004): range of possible grammars is determined solely by the ranking of an a priori finite set of constraints

- Mapping from input to output
- Mapping determined by set of constraints
- Constraints universal, ranking is not

Note: phonotactics is an “easy” problem in the realm of language. No phonological systems extend beyond the regular boundary in the Chomsky hierarchy (aka they can all be described by FSAs)

Further Simplifications

Universal factors

i. A set of 4 violable constraints:
   - ONSET: Every syllable should begin with a consonant.
   - NOCODA: No syllable should end with a consonant.
   - NOINSERTION: No sounds should be inserted.
   - NODELETION: No sounds should be deleted.

ii. Mechanisms for mapping inputs to outputs based on a ranking of the constraints.

Non-universal factors

i. Constraint ranking (drawn randomly from the 8 unique input-output mappings)

ii. Set of 2, 3, or 4 consonants (drawn randomly from 20 possible consonants)

iii. Set of 2, 3, or 4 vowels (drawn randomly from 10 possible vowels)

iv. Consonant used when insertion of a consonant is needed (drawn randomly from the language's consonants)

v. Vowel used when insertion of a vowel is needed (drawn randomly from the language's vowels)

Example language

Constraint ranking:
NOCODA ≫ NODELETION ≫ NOINSERTION ≫ ONSET
Consonants: z x n; **Vowels:** e u
Consonant for insertion: z
Vowel for insertion: e

Example input-output pairs:

euzun	→	.e.u.zu.ne.
un	→	.u.ne.
xxxne	→	.xe.xe.xe.ne.
nezu	→	.ne.zu.
eznx	→	.e.ze.ne.xe.
zuxue	→	.zu.xu.e.

Any Qualms?

Approach Overview

1. Define the space of learning problems (L)
2. Meta-training
3. Verification that inductive bias was acquired

1. Defining the space of learning problems

Inductive biases that we test for		
1. A bias for languages with a consistent constraint ranking.	2. A bias for languages that use a consistent set of constraints.	3. A bias for applying observed patterns to novel lengths.
4. A bias for grouping sounds into two universal phoneme classes, namely consonants and vowels.	5. A bias for ONSET and NOCODA over similar constraints (NOONSET, CODA).	6. A bias for assuming that certain mappings imply certain other mappings (*implicational universals*).

- Translate biases into space of possible languages.

2. Meta-Training

- M_0 and M_i are parameters of seq2seq neural network (encoder-decoder)

- *Meta-training set:* 20,000 languages
  - Each language: 100 train and test examples (100-shot learning)
- Every 100 steps, test on 500 held-out languages.
  - Terminate after 10 evaluations w/out improvement
- *Meta-test set:* 1,000 held-out languages

2. Meta-Training Results

98.8% accuracy with meta-learned initial parameters

vs.

6.5% accuracy with a randomly-initialized model

Gut check

Is this cheating?

3. Verifying that inductive bias was acquired

Inductive biases that we test for		
1. A bias for languages with a consistent constraint ranking.	2. A bias for languages that use a consistent set of constraints.	3. A bias for applying observed patterns to novel lengths.
4. A bias for grouping sounds into two universal phoneme classes, namely consonants and vowels.	5. A bias for ONSET and NOCODA over similar constraints (NOONSET, CODA).	6. A bias for assuming that certain mappings imply certain other mappings (*implicational universals*).

- Ease of learning
- Poverty of the stimulus

Ease of Learning

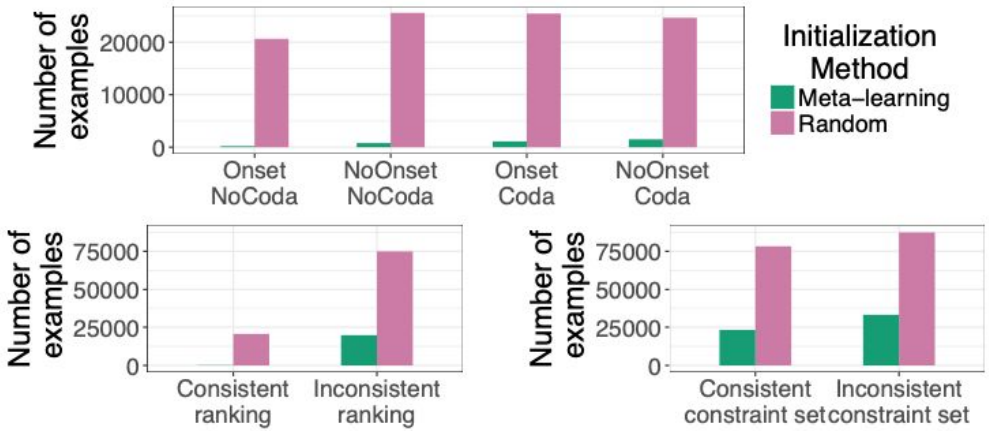


Figure 4: The number of examples needed to learn a language to 95% accuracy (lower is better). Each bar is an average of 80 to 100 languages. Meta-learning improves performance on all conditions.

Surprise: Favors languages consistent training data

Data generated with:

- Onset
- NoCoda

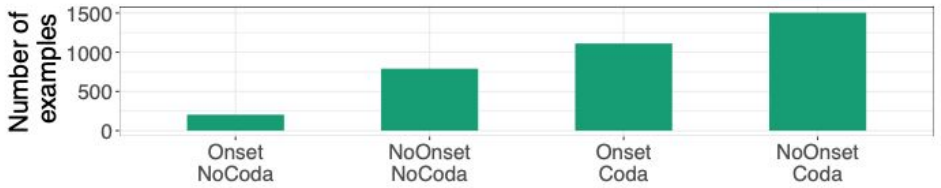
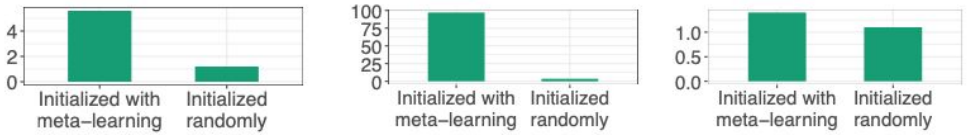


Figure 5: Data from the top panel of Figure 4 re-plotted at a different scale: The number of examples needed by the model initialized with meta-learning to learn languages with different sets of constraints.

Bias hath been bestowed



(a) Ratios comparing languages with an incorrect set of constraints to those with ONSET and NOCODA.

(b) Ratios comparing languages with an inconsistent constraint ranking to those with a consistent ranking.

(c) Ratios comparing languages with an inconsistent set of constraints to those with a consistent set.

Figure 6: Each subplot shows the ratio of the average number of examples need to learn a language with a property inconsistent with typology to the average number needed to learn a language with the analogous consistent property; higher is better. In each case, the model initialized with meta-learning favors the typologically consistent language type more strongly than does the randomly initialized model. The ratios derive from the results shown in Figure 4.

Poverty of the Stimulus

- All new phonemes
- Length 5
- Implicational universals

Results

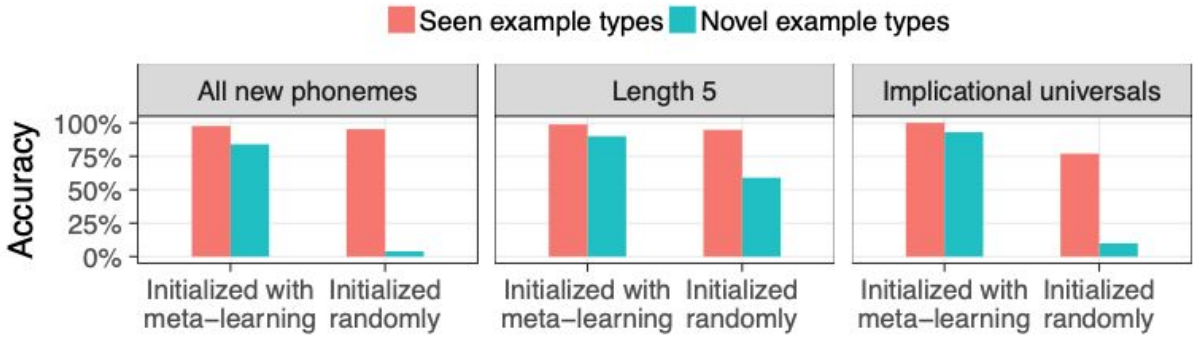


Figure 7: Results on poverty-of-the-stimulus experiments. Both models perform well on the example categories they have seen before, but the performance of the randomly-initialized model plummets when it is tested on novel types of examples; the meta-trained model exhibits less of a performance drop in these cases.

(Their) Conclusions

- meta-learning can impart universal inductive biases specified by the modeler

- this technique could be applied to naturally-occurring linguistic data for which we do not know the underlying data-generating process, to lend insight into the inductive biases that shaped this data
    - meta-learning can disentangle universal inductive biases from non-universal factors

Meta Questions

- What kind of “meta-bias” might this meta-learning framework have?

- Is there one domain-general learning algorithm for language? Or is it more modular?
- I.e. Will a learning algorithm that works well for phonology work well for syntax?

Final Questions

Is this cheating?

What would it mean to not cheat?

Fun idea. Thoughts?

“Properties of the learning mechanism explain patterns found in natural language.” (Heinz 2007)

How about the inverse:
Patterns found in natural language explain properties of the learning mechanism.

A few fun, related papers

Meta-Learning of Compositional Distributions in Humans and Machines (Kumar et al. 2020)

No Free Lunch in Linguistics or Machine Learning (Rawski and Heinz 2019)

- In response to: Generative linguistics and neural networks at 60 (Pater 2019)

Inductive Learning of Phonotactic Patterns (Heinz 2007)