A Benchmark for the Machine Learning of Regular Languages

Jeffrey Heinz

Hubert Curien Laboratory
Jean Monnet University
07.11.2023

This Talk

1. Introduce MLRegTest, which contains training and test sets for 1800 regular languages spanning 16 subclasses.
2. 108,000 experiments on recurrent neural networks using MLRegTest.

Main conclusions

1. Lots of variation!
2. NNs perform better on randomly generated test sets than test sets designed to contain pairs of strings $x \in L$, $y \notin L$ and `string_edit_distance`(x,y)=1.
3. Formal properties of regular languages – such as logical or algebraic properties – better account for learning difficulty than the size of the minimal DFA or its syntactic monoid.

Acknowledgments

People

- Sam van der Poel (Georgia Tech PhD student)
- Dakotah Lambert (PhD Ling 2022)
- Kalina Kostyszyn (Ling, current PhD)
- Paul Fodor (Professor, Stony Brook)
- Chihiro Shibata (Professor, Hosei University)

- Derek Andersen (CompLing MA 2021)
- Joanne Chau (CompLing MA 2020)
- Tiantian Gao (PhD CS 2019)
- Emily Peterson (Ling, CompLing MA 2020)
- Cody St. Clair (Ling MA 2020)
- Rahul Verma (MS CS 2018)

Acknowledgments, continued

Computing Resources

Language class verification, data set creation, and neural network training and evaluation were completed on the Stony Brook SeaWulf HPC cluster maintained by Research Computing and Cyberinfrastructure, and the Institute for Advanced Computational Science at Stony Brook University and made possible by NSF grant #1531492.

iACS
INSTITUTE FOR ADVANCED
COMPUTATIONAL SCIENCE

Part I

Motivation

Why Artificial Formal Languages for ML?

1. Classifying sequences is useful in many fields (software engineering, bioinformatics, nlp)
2. Evaluating ML systems on how well they can learn **known** classifiers allows finer examination of the capabilities of ML systems, which can build confidence when they are applied to the learning of **unknown** classifiers
3. This approach has a rich history in several traditions: computational learning theory, grammatical inference, neural networks, and more.

Why 1800 Regular Languages?

1. Regular languages have many characterizations: regular expressions, finite-state acceptors, monadic second order logic with successor or precedence.
2. The languages in the 16 classes used better represent different corners of the space of regular languages compared to earlier benchmarks (Reber, 1967; Tomita, 1982; Bhattamishra *et al.*, 2020).
3. These classes are also understood along logical and algebraic dimensions (McNaughton and Papert, 1971; Pin, 2021).
4. The logical characterizations have been argued to have cognitive interpretations (Rogers and Pullum, 2011; Jäger and Rogers, 2012; Rogers *et al.*, 2013)

Why 1800 Regular Languages?

1. Regular languages have many characterizations: regular expressions, finite-state acceptors, monadic second order logic with successor or precedence.
2. The languages in the 16 classes used better represent different corners of the space of regular languages compared to earlier benchmarks

   (Reber, 1967; Tomita, 1982; Bhattamishra *et al.*, 2020).
3. These classes are also understood along logical and algebraic dimensions

   (McNaughton and Papert, 1971; Pin, 2021).
4. The logical characterizations have been argued to have cognitive interpretations

   (Rogers and Pullum, 2011; Jäger and Rogers, 2012; Rogers *et al.*, 2013)

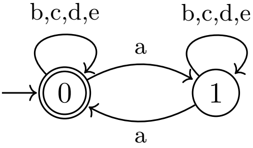
We acknowledge they may still be insufficiently representative ...

Part II

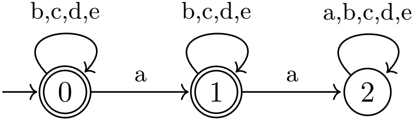
Subregular Hierarchies

Some finite-state acceptors

"Odd numbers of *a* are forbidden."

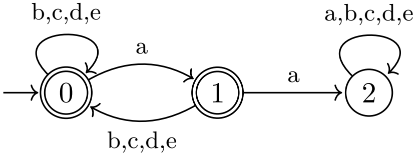


"The subsequence *aa* is forbidden."

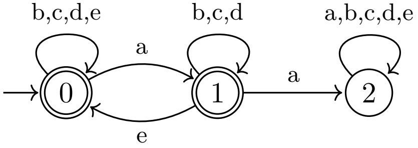


Some finite-state acceptors, continued

"The substring *aa* is forbidden."

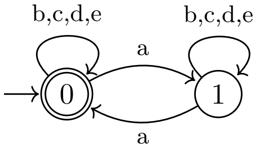


"The substring *aa* on the $\{a\ e\}$ tier is forbidden."



Counting modulo n

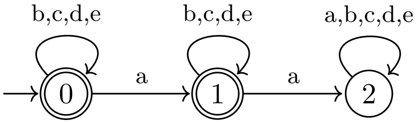
"Odd numbers of a are forbidden."



- Algebraically, such languages are *periodic*.
- We call the class of purely periodic languages languages with a prime-numbered cycle $\mathbb{Z}_p$ (after the algebraic cyclic group).
- Regular languages with a periodic component require MSO logic.

Containing Subsequences or Not

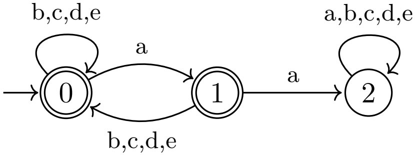
"The subsequence *aa* is forbidden."



- Subsequences are defined as symbols in a *precedence* order (not necessarily successive).
- The length of a subsequence is the *factor width*.
- Languages defined by forbidding subsequences are Strictly Piecewise (SP).
- Their complements are co-Strictly Piecewise (coSP).
- The Boolean closure of SP languages are Piecewise Testable (PT), characterizable with Propositional Logic.
- The Star-Free (SF) languages are First Order definable with Precedence.

Containing Substrings or Not

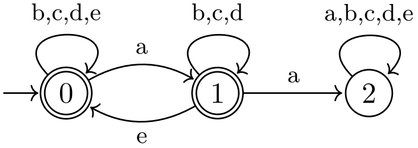
"The substring *aa* is forbidden."



- Substrings are defined as *successive* symbols.
- The length of a substring is the *factor width*.
- Languages defined by forbidding substrings are Strictly Local (SL), characterizable as Conjunctions of Negative Literals.
- Their complements are co-Strictly Local (coSL) (Disjunctions of Positive Literals).
- The Boolean closure of SL languages are Locally Testable (LT), characterizable with Propositional Logic.
- Locally Threshold Testable (LTT) languages are First Order definable with Successor.

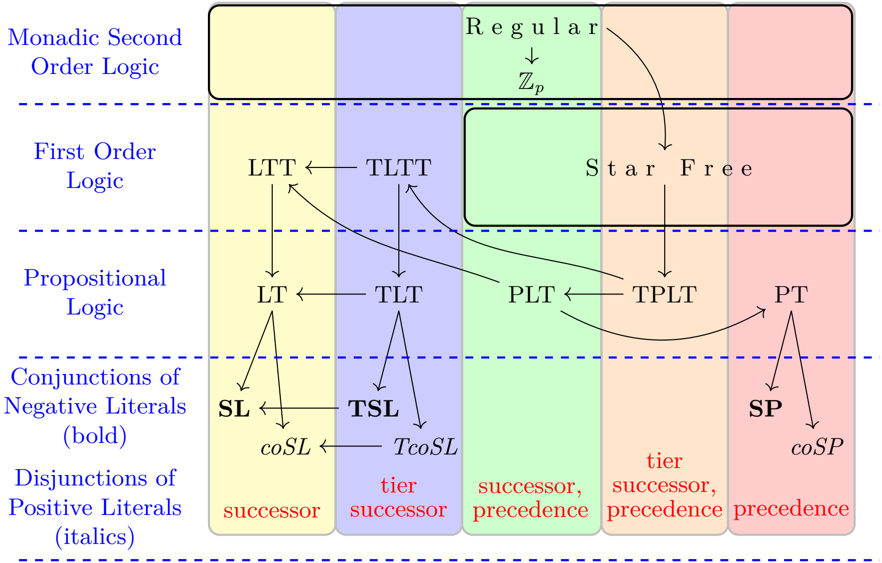
Containing Substrings on Tiers or Not

"The substring *aa* on the $\{a\ e\}$ tier is forbidden."



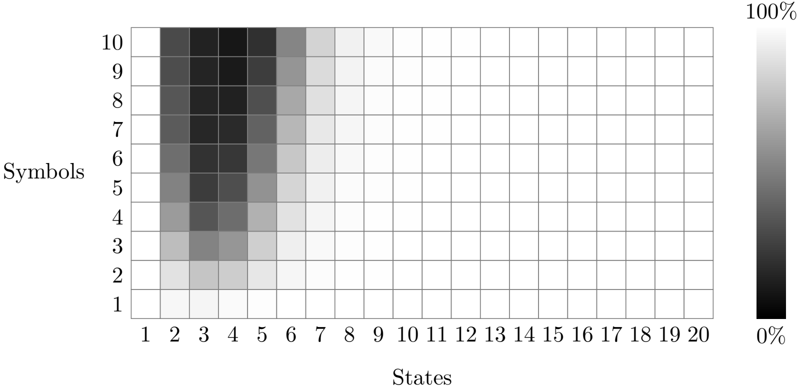
- A *tier* T is a subset of Σ.
- The tier-projection of $w \in \Sigma^*$ is the longest string $u \in T^*$ such that u is a subsequence of w. Remove the non-tier symbols from w to get u.
- Tier-projection lifts to languages and classes to obtain new ones.
- Every Local class is properly contained within a Tier-based superclass (of the same logical type but now under the tier-successor relation).
- Projecting Piecewise classes to tiers does not change their expressivity. (Piecewise classes are closed under tier-projection.)

Summary


Monadic Second Order Logic
First Order Logic
Propositional Logic
Conjunctions of Negative Literals (bold)
Disjunctions of Positive Literals (italics)
Regular
$\mathbb{Z}_p$
Star Free
LTT
TLTT
LT
TLT
PLT
TPLT
PT
SL
TSL
SP
coSL
TcoSL
coSP
successor
tier successor
successor, precedence
tier successor, precedence
precedence

Cf. random construction

Random construction of DFA **may not** lead to diversity.

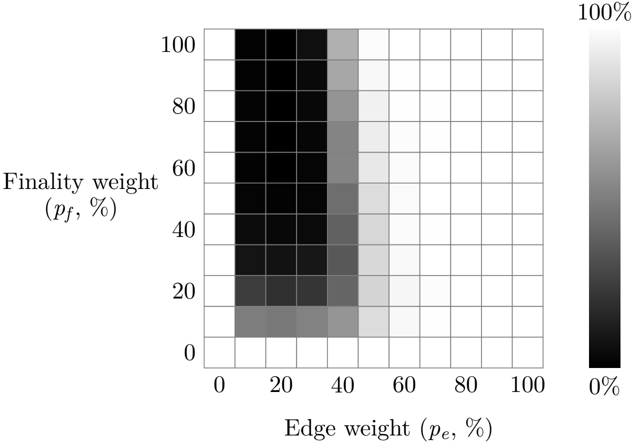


The proportion of Strictly Local languages upon fair generation,

$$p_e = p_f = 0.5.$$

Cf. random construction

Random construction of DFA **may not** lead to diversity.



The proportion of Strictly Local languages for 7 states, 8 symbols.

Part III

ML-RegTest

Languages

class	bases	alphabets	windows	thresholds	total
SL	10	3	3		90
coSL	10	3	3		90
SP	10	3	3		90
coSP	10	3	3		90
LT	10	3	3		90
PLT	10	3	3		90
PT	10	3	3		90
LTT	10	3	3	2(3)*	180
SF	10	3			30
$\mathbb{Z}_p$	10	3			30
Reg	10	3			30
total					**900**

The asterisk (*) indicates that while there were actually 3 thresholds, since they occur in 3:2:1 ratio, they doubled the number of languages.

Languages

class	bases	alphabets	windows	tiers	thresholds	total
TSL	10	1	3	1		30
	10	2	3	2		120
TcoSL	10	1	3	1		30
	10	2	3	2		120
TLT	10	1	3	1		30
	10	2	3	2		120
TPLT	10	1	3	1		30
	10	2	3	2		120
TLTT	10	1	3	1	2(3)*	60
	10	2	3	2	2(3)*	240
total						**900**

The asterisk (*) indicates that while there were actually 3 thresholds, since they occur in 3:2:1 ratio, they doubled the number of languages.

What we mean by "belongs to language class"

- Some classes contain others (for example LT contains SL).
- Some classes are incomparable but overlap (for example LT and PT).

When we say "Language L belongs to class X" we mean

1. L belongs to class X, and
2. L does not belong to any class Y which is a subset of, or incomparable with, X.

Language Variables

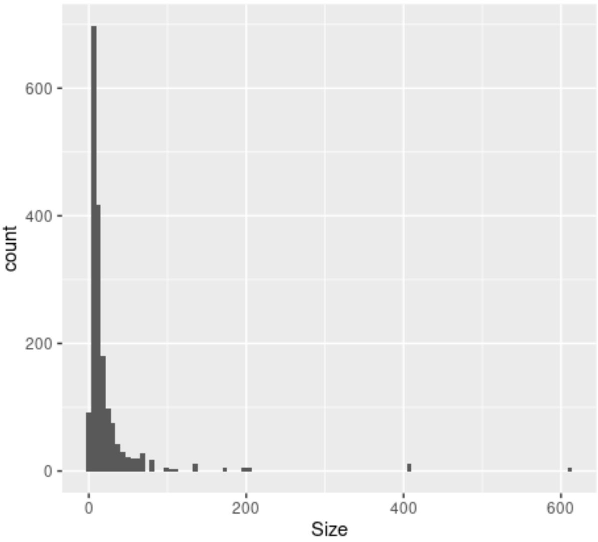
logical level	MSO, FO, Prop, CNL/DPL
order relation	successor, precedence, tier-successor
alphabet size	{4, 16, 64}
factor widths	{2, 4, 6}
tier size	{2, 3}; {4, 7}; {6, 11}
threshold	{2,3,5}

Acceptors for the 1800 languages were created and their class memberships were verified with the The Language Toolkit and amalgam softwares (Lambert, 2022).

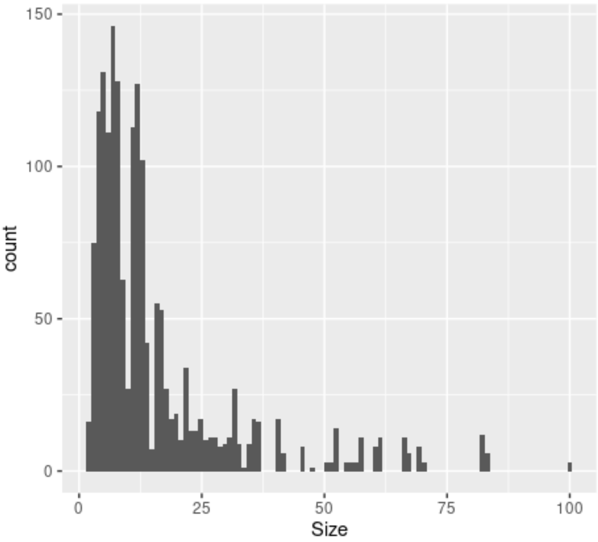
Sizes of Minimal DFAs

Type of Machine	min	max	median	mean	s.d.
Minimal DFA	2	613	11	23.35	53.19
Monoid of Minimal DFA	2	2701	45	142	304.76

Sizes of Minimal DFAs



Sizes of Minimal DFAs



Research Questions

1. What is the effect of the language class?
2. What is the effect of logical level?
3. What is the effect of order relation (successor, tier-successor, precedence)?
4. What is the effect of alphabet size?
5. What is the effect of DFA size? Monoid size?
6. Other good questions are left unexplored (Tier size? Factor width? Threshold?)

Training Data Sets

Training data was randomly generated with duplicates. For each L, We generated 50k positive and 50k negative strings subject to the following constraints:

- Equally many strings of each length between 20 and 29.
- Uniform distribution over the paths in the minimal, acyclic DFA representing words of length n belonging to L

We downsampled to obtain three nested sets:

Small 1k, Mid 10k, Large 100k.

Why no short strings?

- We excluded short strings because we wanted equal numbers of positive and negative strings at each length.
- For some (many) languages, this is not possible if the lengths are small.
- We will return to the issue of short strings at the end.

VALIDATION DATA SETS

Validation data was randomly generated with duplicates. We generated 50k positive and 50k negative strings subject to the same constraints as Training Data and:

- The development set and training sets were disjoint.

We downsampled to ultimately obtain three nested sets:

Small 1k, Mid 10k, Large 100k.

Random Test Sets

For each language there are 4 kinds of test sets.

Short Random

50k positive and 50k negative strings were randomly generated without duplicates subject to the constraints:

- There are equally many strings of each length between 20 and 29.
- Disjoint from the training and validation data.
- Uniform distribution over the paths in the minimal, acyclic DFA representing words belonging to $L \cap \Sigma^n$

Long Random

As above, except that there are equally many strings of each length between 31 and 50.

Adversarial Test Sets

For each string length n, a minimal, acyclic finite-state transducer which mapped each $x \in L$ of length n to each string $y \notin L$ such that $\texttt{string\_edit\_distance}(x, y) = 1$.

Short Adversarial

- 50k positive and 50k negative strings with lengths between 20 and 29.
- The positive and negative strings were disjoint from the training and validation data.
- String sampled without replacement from a uniform distribution over the paths of the transducer.

Long Adversarial

- As above, but strings were of length between 31 and 50.

Test Data Summary

We hypothesized difficulty increases as follows.

Long Adversarial

Long Random

Short Adversarial

Short Random

We downsampled to obtain three nested sets for each kind of test:

Small 1k, Mid 10k, Large 100k

Implementation

Data was generated by exporting the DFA made with The Language Toolkit to the `.att` format, and were then processed with the software libraries `openfst` (Allauzen *et al.*, 2007) and Pynini (Gorman, 2016; Gorman and Sproat, 2021)

One exception. The datasets for coSL, TcoSL, and coSP languages were generated simply by flipping positive and negative strings in the corresponding datasets for the SL, TSL and SP languages.

Part IV

Experiments

Setup

1. For each language, we trained five different neural networks on each of three different training sets:

   Small 1k, Mid 10k, Large 100k

2. Each trained network was tuned with one validation set:

   Small 1k

3. Each trained network was tested on each of four Large 100k test sets:

   SR, SA, LR, LA

Consequently, there are $1800 \times 5 \times 3 \times 4 = 108,000$ experimental observations.

Performance Measures

We collected different measures of performance on the test sets.

1. Accuracy: Proportion correctly classified. (1 best, 0 worst)
2. F-score: Harmonic mean of precision and recall. (1 best, 0 worst)
3. Brier Score: Incorporates confidence with correct classification. (0 best, 1 worst)
4. Area Under the Curve: Balances True positive rate with false positive rate. (1 best, 0 worst)

The Neural Networks

1. Simple Recurrent Neural Network (RNN)
2. Long Short-Term Memory (LSTM)
3. Gated Recurrent Unit (GRU)
4. 2 layer LSTM
5. Transformer

Tensorflow and Keras APIs were used to implement all NNs (Abadi *et al.*, 2015). The total number of trainable parameters depend on alphabet size, but they are ordered roughly as:

simple RNN ($\sim$45k) < transformer ($\sim$115k) < GRU ($\sim$126k) < LSTM ($\sim$165k) < 2-layer LSTM ($\sim$326k)

Additional details

Parameters and features that are the same across all architectures are:

- Batch size = 64
- Epochs = 30
- Loss = Binary cross-entropy
- Optimizer = Adam
- Learning rate = 2e-5

Dropout was not used except for transformers because their performance was especially bad without it. For them, we used dropout probability = 0.2.

Part V

Results

Caveat

Our neural networks are simple. If we added more layers and units or adopted a different architecture, it is possible the distinctions we observe could be erased.

Nonetheless, even these basic networks provide some sense of where some of the challenges are in generalizing over sequences drawn from regular languages.

And the purpose of the ML-RegTest is to challenge folks to find a single ML system that excels across the board.

Analytic Techniques

1. Aggregating over factor width, tier size, threshold, and individual language within a class yields a full factorial design whose design variables are `{training set, test set, language class, network type, alphabet}`, and whose response variables are `{accuracy, fscore, auc, brier}`, yielding 2880 “aggregated observations”.
2. A design variable can be singled out as a treatment with the remaining variables serving as blocking variables. Then a non-parametric, repeated measure ANOVA can be conducted with a Friedman Test to determine whether any of the treatment levels differ.
3. If so, post hoc analyses using the Nemenyi-Wilcoxon-Wilcox all-pairs test can be used to determine *where* the significant differences exist.
4. We can also examine post-hoc the *size* of the effect (Cohen’s d).

Sanity Checks

Do the performance measures correlate with each other?

Sanity Checks

Do the performance measures correlate with each other?

Spearman's rho

	Accuracy	AUC	Brier
AUC	0.950	–	–
Brier	−0.940	−0.897	–
F-score	0.873	0.834	−0.811

YES. Since all measures strongly correlate,
we henceforth just report accuracy.

Sanity Checks

2. Is performance on the SL/coSL, TSL/TcoSL, SP/coSP pairs the same?

Sanity Checks

2. Is performance on the SL/coSL, TSL/TcoSL, SP/coSP pairs the same?

Accuracy		All-pairs p-value
SL	coSL	1.000
0.806	0.806	
SP	coSP	1.000
0.731	0.731	
TSL	TcoSL	1.000
0.769	0.769	

YES. There is no significant difference between these pairs of language classes.

Sanity Checks

3. Does more training data help?

Sanity Checks

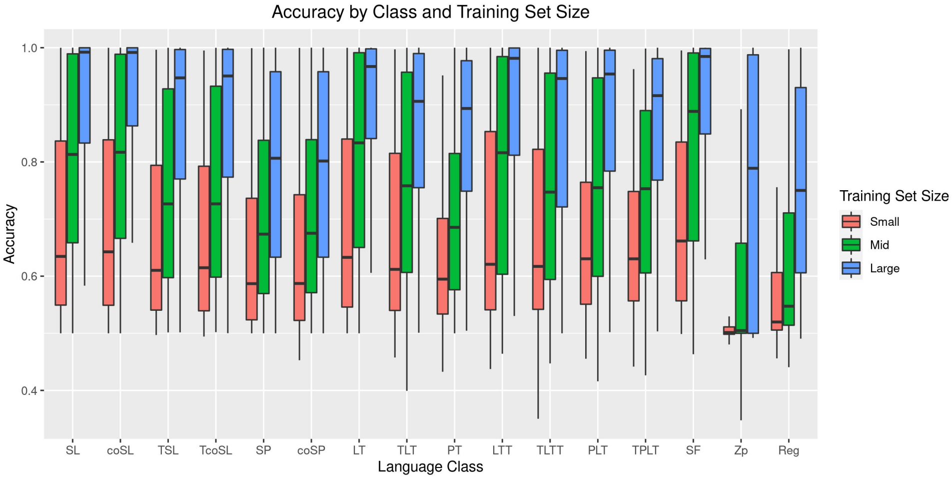
3. Does more training data help?

YES

Average accuracy; each pair is significant ($p < 2.2\text{e}{-16}$).

Small	Mid	Large
0.668	0.772	0.863

Accuracy by Class and Training Set Size



Research Questions

1. What is the effect of the test set?
2. What is the effect of logical level?
3. What is the effect of order relation (successor, tier-successor, precedence)?
4. What is the effect of alphabet size?
5. What is the effect of these basic neural networks?

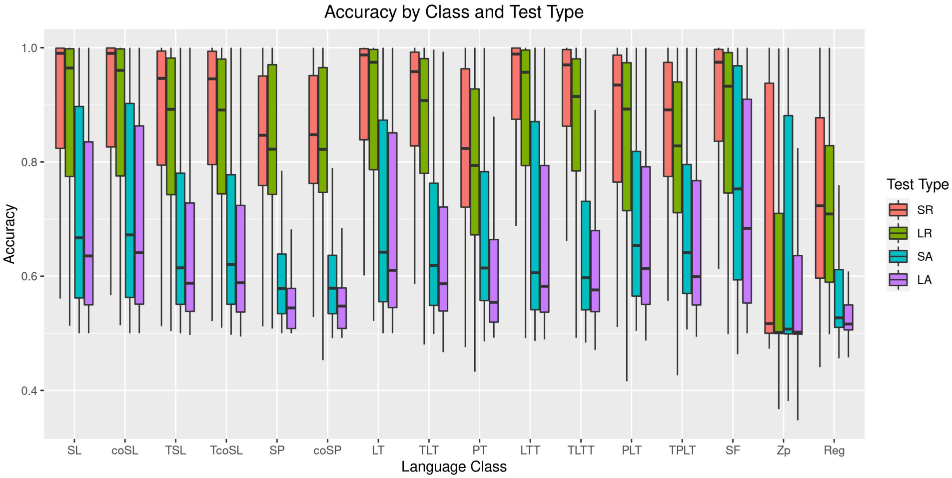
What is the effect of the test set?

These average accuracy scores are pairwise significantly different ($p \leq 2\text{e}{-16}$).

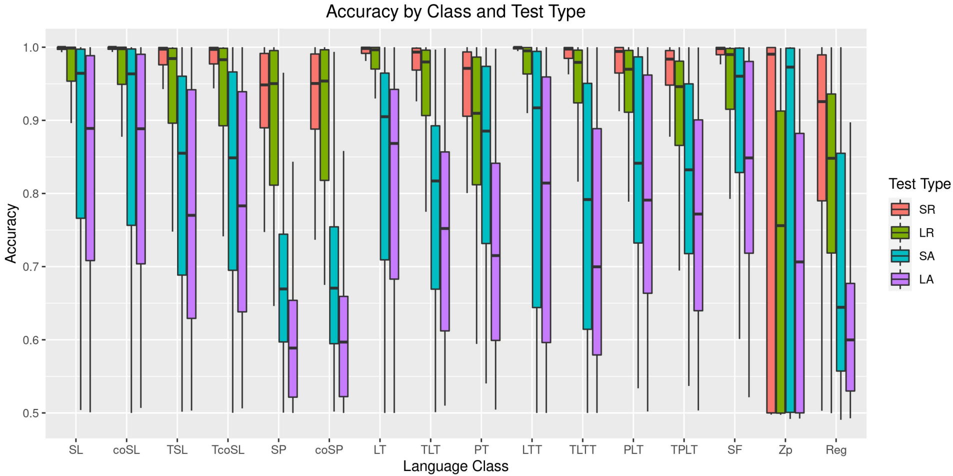
SR	LR	SA	LA
0.884	0.847	0.686	0.653

Pair	Effect Size (Cohen's d))	Interpretation
SR, LR	0.286	Small
LR, SA	1.274	Large
SA, LA	0.188	Small

Accuracy by Class and Test Set



Accuracy by Class and Test Set



What is the effect of logical level?

Does accuracy decrease as we expressivity increases logically?

Group	Classes
CNL	SL, SP, TSL
DPL	coSL, coSP, TcoSL
PROP	LT, PLT, PT, TLT, TPLT
FO	LTT, TLTT, SF
REG	Zp, Reg

The MSO divide matters most, but...

Average accuracy by logical level in decreasing order.

FO	PROP	CNL	DPL	REG
0.781	0.776	0.768	0.768	0.697

The MSO divide matters most, but...

Average accuracy by logical level in decreasing order.

FO	PROP	CNL	DPL	REG
0.781	0.776	0.768	0.768	0.697

p-values from Nemenyi-Wilcoxon-Wilcox all-pairs test

	CNL	DPL	FO	PROP
DPL	0.99	–	–	–
FO	3.4e−6	3.5e−5	–	–
PROP	5.7e−6	5.7e−5	1.00	–
MSO	1.7e−6	1.2e−7	4.8e−14	4.2e−14

The MSO divide matters most, but...

Average accuracy by logical level in decreasing order.

FO	PROP	CNL	DPL	REG
0.781	0.776	0.768	0.768	0.697

Effect sizes as measured by Cohen's d

	CNL	DPL	FO	PROP
DPL	nss	–	–	–
FO	-0.079	-0.077	–	–
PROP	-0.059	-0.057	nss	–
REG	0.394	0.397	0.463	0.452

The MSO divide matters most, but...

Average accuracy by logical level in decreasing order.

FO	PROP	CNL	DPL	REG
0.781	0.776	0.768	0.768	0.697

Takeaways:

- In aggregate, FO had highest accuracy, contrary to expectations.
- At the FO level and below, differences, if significant, have very small effects.
- The significant differences between MSO and everything else are medium effects.

What is the effect of order relation?

Is there a significant difference in accuracy?

Group	Classes
SUCC	SL, coSL, LT, LTT
PREC	coSP, PT, SF, SP
TSUCC	TcoSL, TLT, TLTT, TSL
OTHER	PLT, TPLT, Reg, Zp

SUCC then TSUCC then PREC

Average accuracy by order relation in decreasing order.

SUCC	TSUCC	OTHER	PREC
0.800	0.777	0.759	0.747

SUCC then TSUCC then PREC

Average accuracy by order relation in decreasing order.

SUCC	TSUCC	OTHER	PREC
0.800	0.777	0.759	0.747

p-values from Nemenyi-Wilcoxon-Wilcox all-pairs test

	OTHER	PREC	SUCC
PREC	0.007	–	–
SUCC	1.6e−13	4.0e−14	–
TSUCC	0.559	0.220	2.8e−14

SUCC THEN TSUCC THEN PREC

Average accuracy by order relation in decreasing order.

SUCC	TSUCC	OTHER	PREC
0.800	0.777	0.759	0.747

Takeaways:

- In aggregate, successor-based patterns have highest accuracies.
- The significant difference between SUCC and PREC is a small effect ($d = 0.292$).
- The significant differences between other comparisons is very small ($|d| < 0.2$).

What is the effect of alphabet size?

Accuracy

64	16	4
0.798	0.764	0.740

- Each difference is significant.
- The effect sizes between 64/16 and 16/4 are very small ($|d| \sim 0.1$).

What is the effect of the neural network?

Aggregate Accuracy

Simple RNN	2-layer LSTM	LSTM	GRU	Transformer
0.784	0.776	0.773	0.770	0.734

What is the effect of the neural network?

Accuracy by Training Set

	Small	Mid	Large
Simple RNN	**0.719**	**0.781**	0.853
GRU	0.645	**0.776**	**0.889**
LSTM	0.671	0.770	0.879
2-layer LSTM	0.649	**0.783**	**0.897**
Transformer	0.656	0.750	0.795

Bold faced scores are not significantly different from each other, but are significantly different from the non-boldfaced scores.

What is the effect of the neural network?

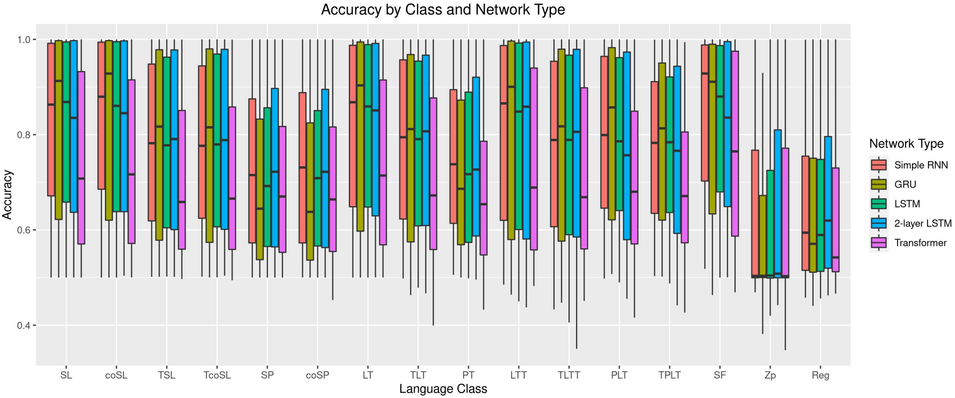
Accuracy by Training Set

	Small	Mid	Large
Simple RNN	**0.719**	**0.781**	0.853
GRU	0.645	**0.776**	**0.889**
LSTM	0.671	0.770	0.879
2-layer LSTM	0.649	**0.783**	**0.897**
Transformer	0.656	0.750	0.795

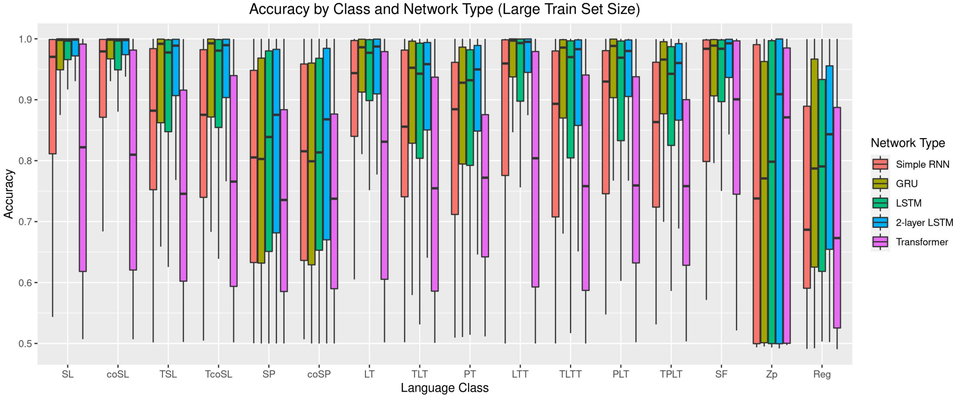
Bold faced scores are not significantly different from each other, but are significantly different from the non-boldfaced scores.

Effect sizes range from negligible (Mid, 2-layer LSTM to LSTM, $d = 0.069$) to medium (Large, 2-layer LSTM to transformer, $d = 0.646$).

Accuracy by Class and Neural Network



Accuracy by Class and Neural Network



What is the effect of grammar size?

Overall correlations

	All train	Large train
DFA size $\sim$ accuracy	-0.050	-0.129
monoid size $\sim$ accuracy	-0.045	-0.121

We calculated other correlations making finer distinctions by network type, test type, and training size. The strongest correlation we found is shown.

2-layer LSTM / Large training set / Short Adversarial test set

Size	Monoid
-0.460	-0.470

Lessons from these experiments on ML-RegTest

1. High performance on Random Test sets does not imply correct generalization as measured by performance on the Adversarial tests.
2. Classification which depends on counting modulo n is hard for neural ML systems to learn.
3. Classification which depends on precedence is harder for neural ML systems to learn than successor
4. Classification ability of neural ML systems does not correlate well with DFA size or monoid size.
5. On small training sets, simple RNNs perform best.
6. On large training sets, 2-layer LSTMs perform best.

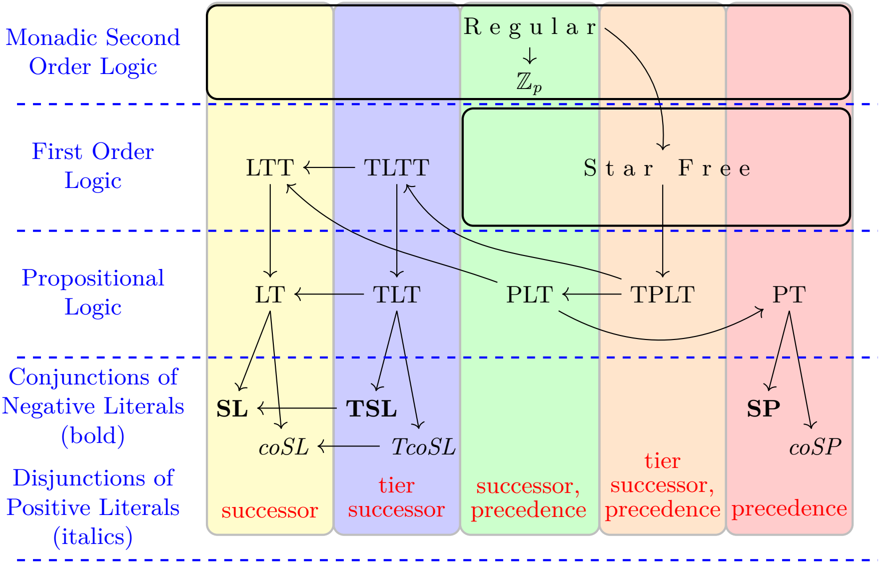
Rethinking Short Strings

- At ICGI 2023, Dana Angluin (Yale) wondered whether the exclusion of shorter strings mattered and presented an analysis indicating they could help.
- At the same event, Adil Soubki (Stony Brook) presented work on testing classical grammatical inference algorithms for sequence classification (RPNI, EDSM, ALERGIA) on MLRegTest. He found including short strings in training dramatically improved outcomes.
- We still need to train the vanilla NNs with shorter strings and evaluate them.

MLRegTest

https://doi.org/10.5061/dryad.dncjsxm4h
https://arxiv.org/abs/2304.07687
https://github.com/heinz-jeffrey/subregular-learning

MERCI BEAUCOUP!



Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
URL https://www.tensorflow.org/

Allauzen, Cyril, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on*

Implementation and Application of Automata, (CIAA 2007), vol. 4783 of *Lecture Notes in Computer Science*, 11–23. Springer.
URL http://www.openfst.org

Bhattamishra, Satwik, Kabir Ahuja, and Navin Goyal. 2020. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 7096–7116. Online: Association for Computational Linguistics.
URL https://aclanthology.org/2020.emnlp-main.576

Gorman, Kyle. 2016. Pynini: A python library for weighted finite-state grammar compilation. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, 75–80. Berlin, Germany.

Gorman, Kyle, and Richard Sproat. 2021. *Finite-State Text Processing*. Morgan & Claypool Publishers.

Jäger, Gerhard, and James Rogers. 2012. Formal language theory: Refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society B* 367:1956–1970.

Lambert, Dakotah. 2022. Unifying classification schemes for languages and processes with attention to locality and relativizations thereof. Doctoral dissertation, Stony Brook University.
URL
https://vvulpes0.github.io/PDF/dissertation.pdf/

McNaughton, Robert, and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

Pin, Jean-Éric. 2021. *Handbook of Automata Theory*. European Mathematical Society Publishing House.
URL https://hal.archives-ouvertes.fr/hal-03579131

Reber, A. S. 1967. Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior* 6:855–863.

Rogers, James, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *Formal Grammar*, edited by Glyn

Morrill and Mark-Jan Nederhof, vol. 8036 of *Lecture Notes in Computer Science*, 90–108. Springer.

Rogers, James, and Geoffrey Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20:329–342.

Tomita, Masaru. 1982. Learning of construction of finite automata from examples using hill-climbing. *Proc. Fourth Int. Cog. Sci. Conf.* 105 – 108.