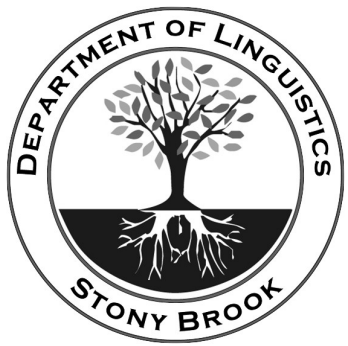
A Benchmark for the Machine Learning of Regular Languages

Jeffrey Heinz



Stony Brook University

ATLAC
October 21, 2022

This Talk

1. Introduce MLRegTest, which contains training and test sets for 1980 regular languages spanning 16 subclasses.
2. 118,000+ experiments on recurrent neural networks using MLRegTest.

Three 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 patterns – such as logical or algebraic complexity – help explain where the difficulty in generalizing lies.

Acknowledgments

People

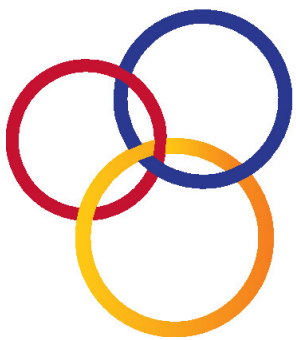
- Sam van der Poel (Georgia Tech PhD student)
- Dakotah Lambert (PhD Ling 2022)
- Kalina Kostyszyn (Ling, current PhD)
- Paul Fodor (Professor, Stony Brook)
- Chihiro Shibata (Professor, Hosei University)

- 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.

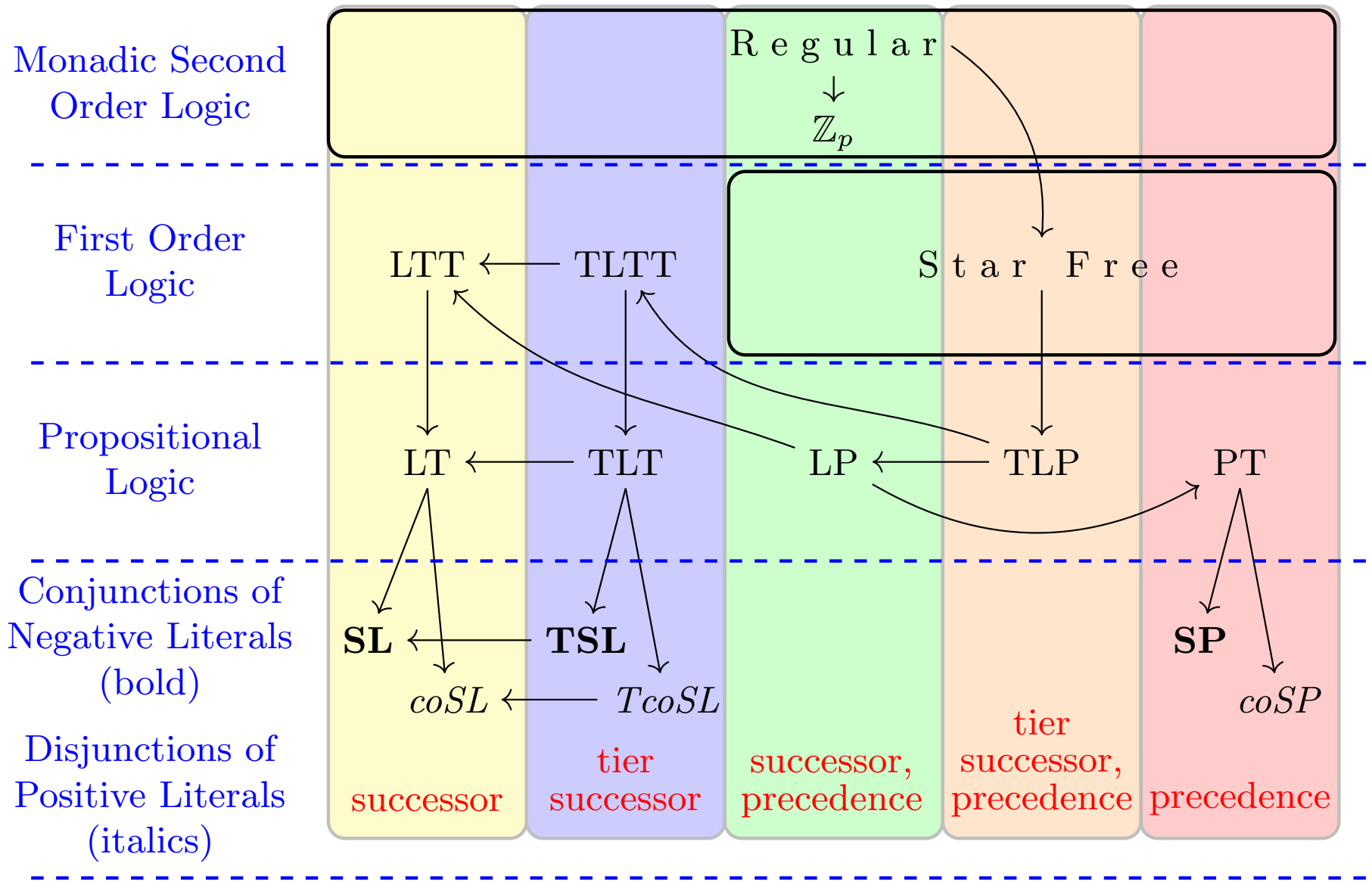
Why 1980 Regular Languages?

1. Regular languages have many characterizations, such as regular expressions and finite-state acceptors
2. These classes are also understood along logical and algebraic dimensions (McNaughton and Papert, 1971; Pin, 2021).
3. The logical characterizations have been argued to have cognitive interpretations (Rogers and Pullum, 2011; Jäger and Rogers, 2012; Rogers *et al.*, 2013).
4. 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; Schmidhuber, 2015).

Part II

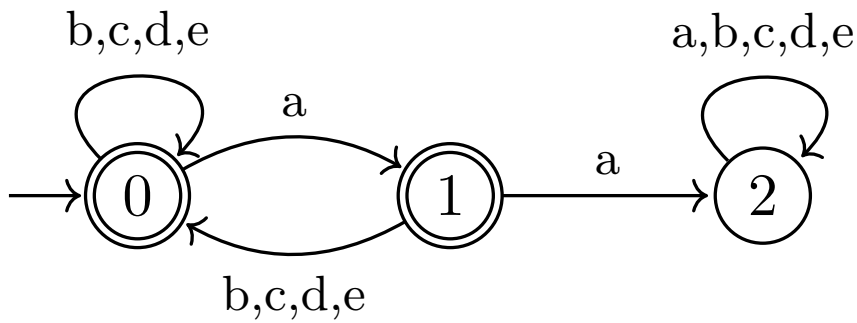
Subregular Hierarchies

The 16 classes

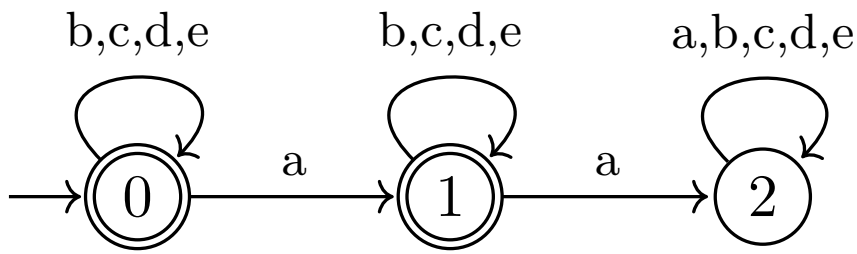


Finite-state acceptors for deciding whether strings contain X

"The substring *aa* is forbidden."

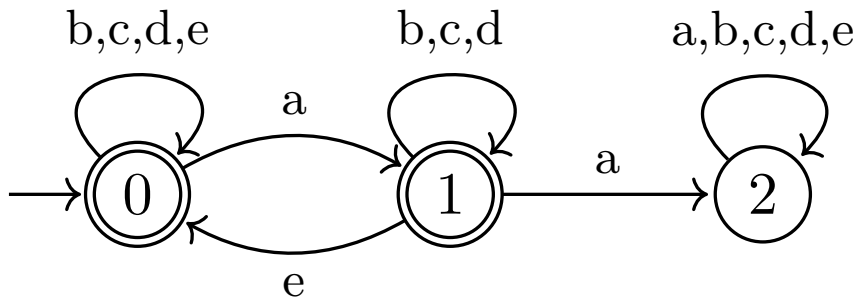


"The subsequence *aa* is forbidden."

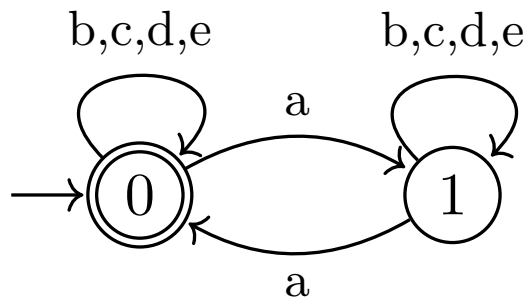


Finite-state acceptors for deciding whether strings contain X

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

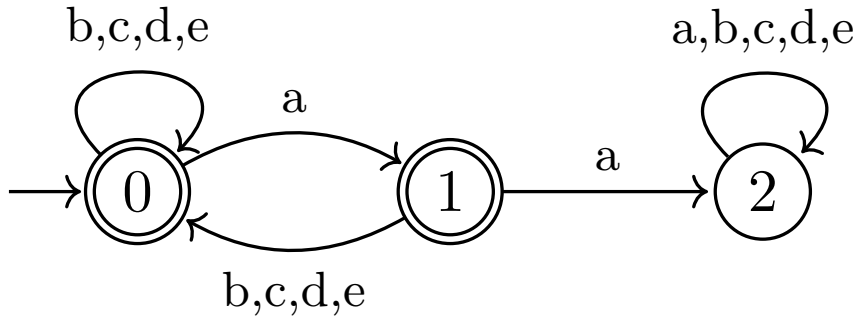


"Odd numbers of a are forbidden."



Containing Substrings or Not

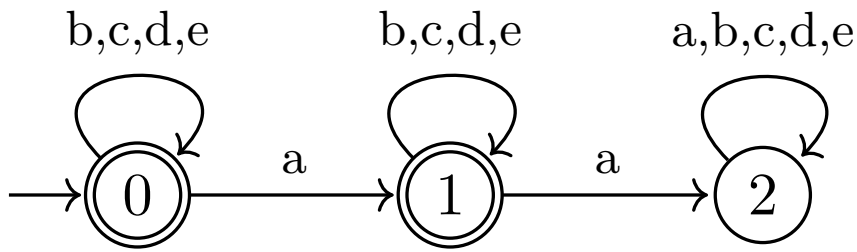
"The substring *aa* is forbidden."



- 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 Subsequences or Not

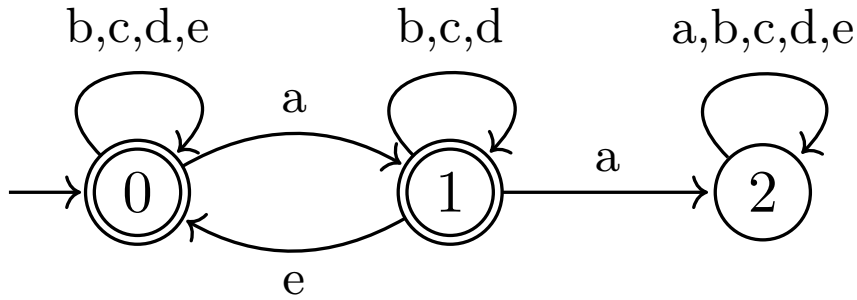
"The subsequence *aa* is forbidden."



- 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 on Tiers or Not

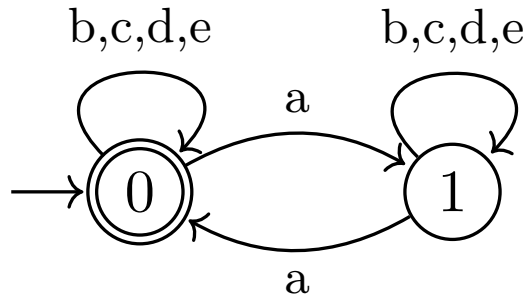
"The substring aa on the $\{a\ e\}$ tier is forbidden."



- 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.)

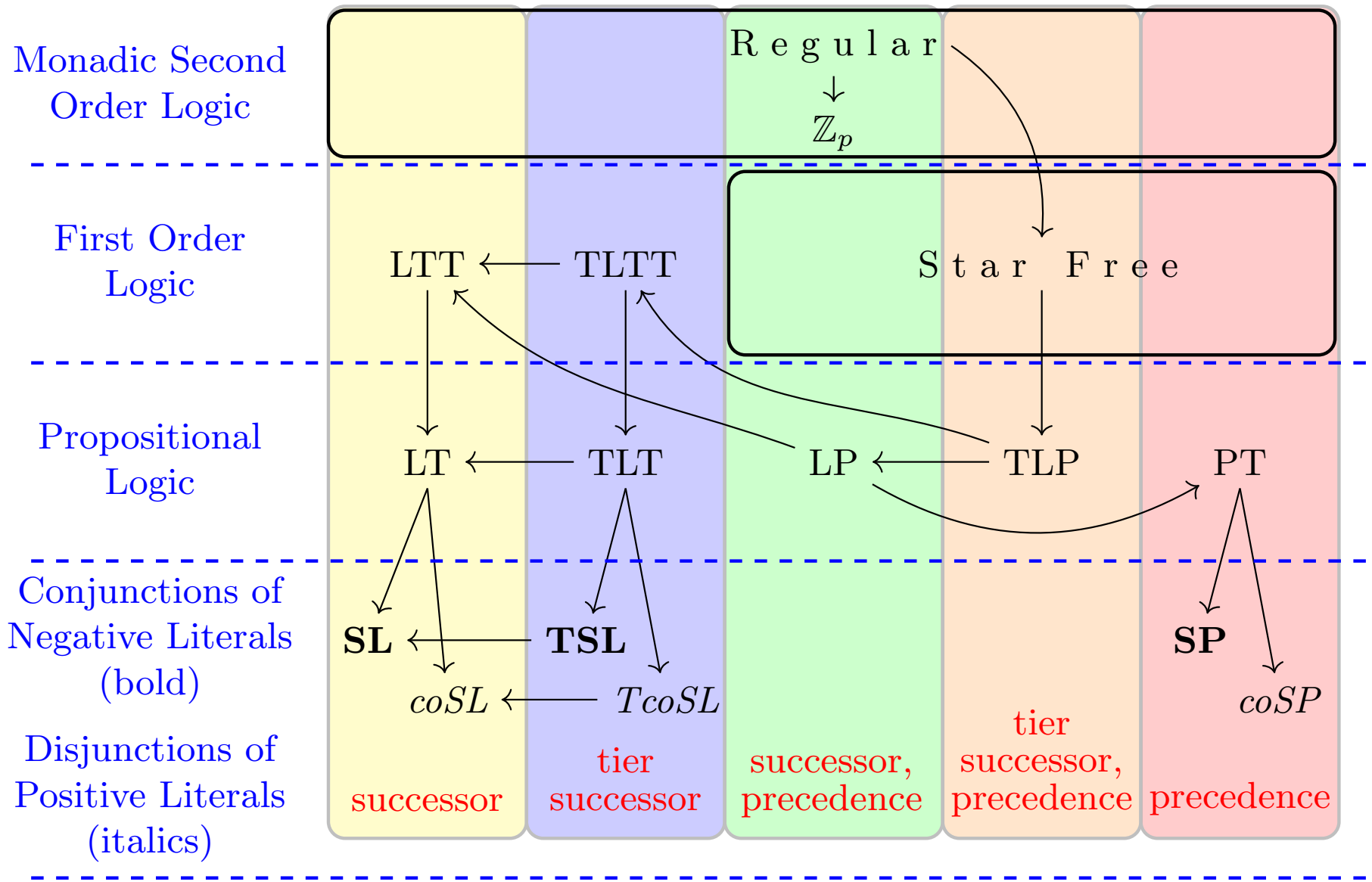
Counting modulo n

"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.

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
LP
TLP
PT
SL
TSL
SP
coSL
TcoSL
coSP
successor
tier successor
successor, precedence
tier successor, precedence
precedence

Part III

ML-RegTest

Languages

class	base lgs	alphabets	widths	tiers	thresholds	total
SL	10	3	3			90
coSL	10	3	3			90
TSL	10	3	3	2		180
TcoSL	10	3	3	2		180
SP	10	3	3			90
coSP	10	3	3			90
LT	10	3	3			90
TLT	10	3	3	2		180
LP	10	3	3			90
TLP	10	3	3	2		180
PT	10	3	3			90
LTT	10	3	3		2*	180
TLTT	10	3	3	2	2*	360
SF	10	3				30
Zp	10	3				30
Reg	10	3				30
total						**1980**

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

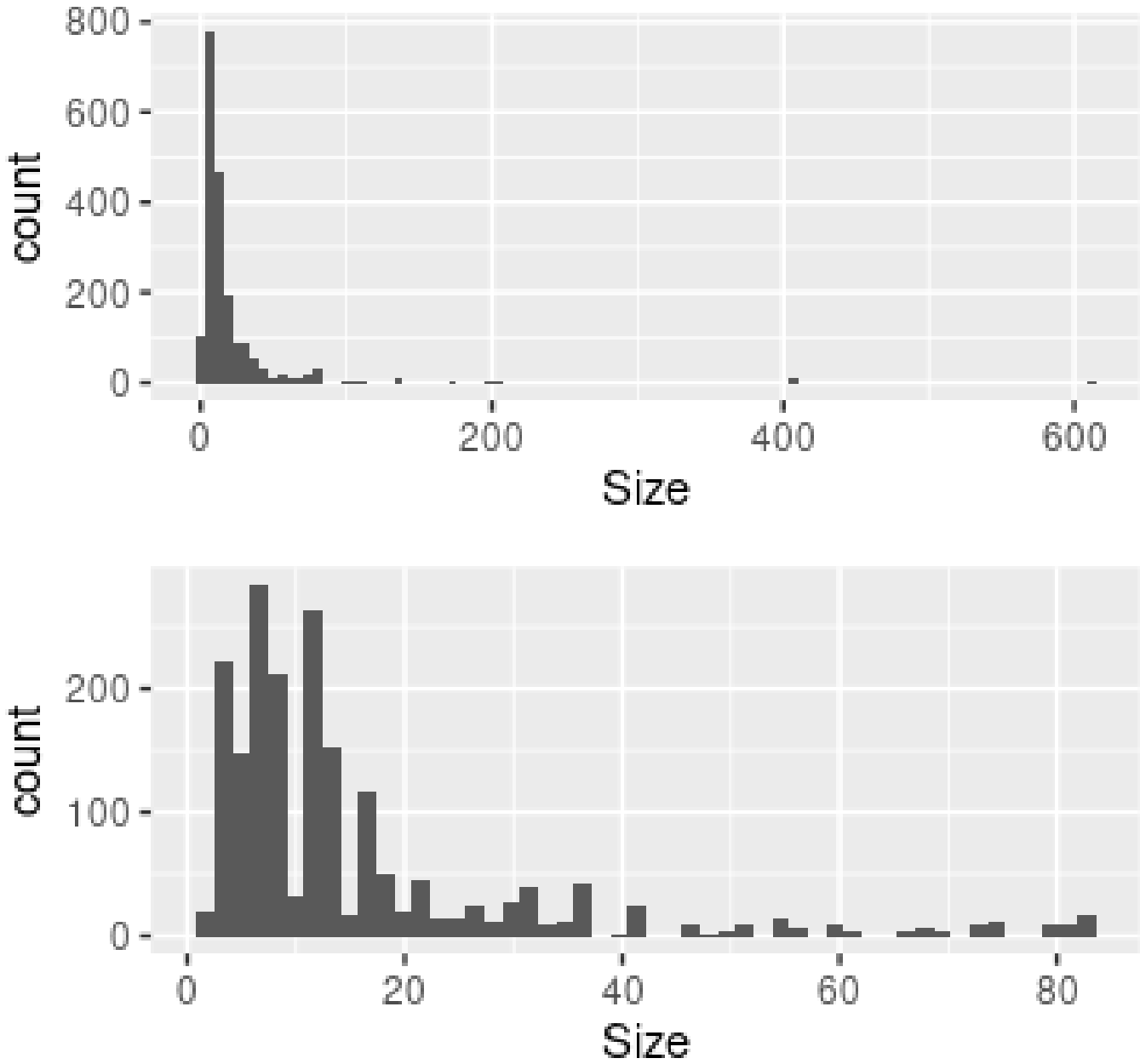
1. L belongs to class X, and
2. L does not belong to any class Y which is a subset of, or incomparable with, X.

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 1980 languages were created and their class memberships were verified with the The Language Toolkit (Lambert, 2022).

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? Tier size? Factor width? Threshold?
5. What is the effect of dfa size? Monoid size?

Training Data Sets

Training data was randomly generated with duplicates. 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 transitions from every state in the minimal dfa.

We downsampled to obtain three nested sets:

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

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

Strings were randomly generated without duplicates subject to these constraints:

- There are equally many strings of each length between 20 and 29.
- These strings are disjoint from the training and validation data.

Long Random

Strings were randomly generated without duplicates subject to the constraint that there are equally many strings of each length between 31 and 50.

Adversarial Test Sets

Short Adversarial

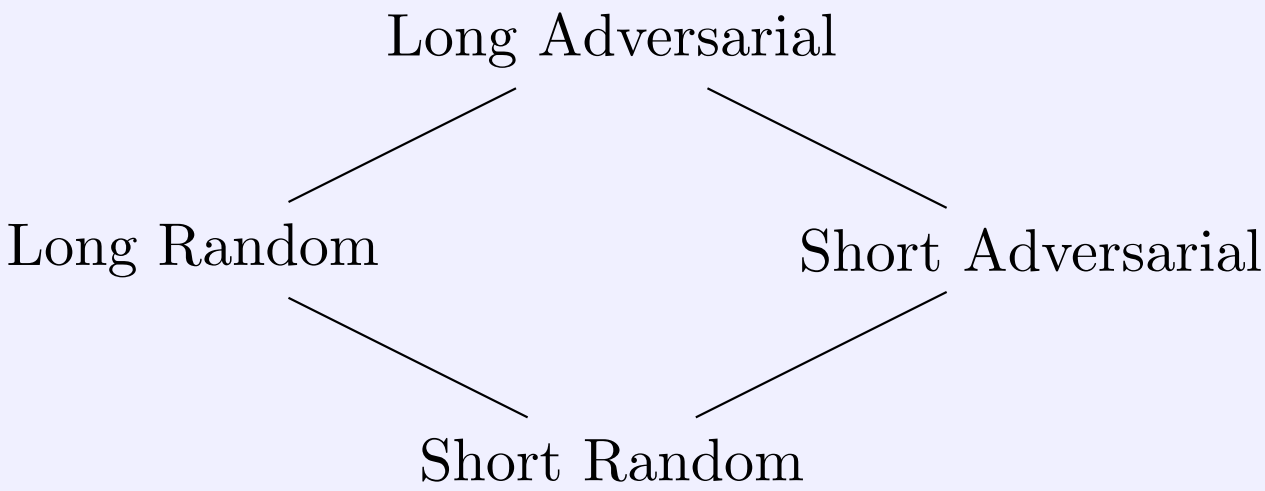
- Positive strings were randomly generated without duplicates with equally many strings for each length between 20 and 29.
- For each positive string x a negative string y was obtained such that $\texttt{string\_edit\_distance}(x, y) = 1$.
- The positive strings were disjoint from the training and validation data.

Long Adversarial

- Positive strings were randomly generated without duplicates but longer lengths between 31 and 50.
- For each positive string x a negative string y was obtained such that $\texttt{string\_edit\_distance}(x, y) = 1$.

Test Data Summary

We hypothesized difficulty increases as follows.



For each kind of test, we generated 50k positive strings and 50k negative strings. 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 $1980 \times 5 \times 3 \times 4 = 118,800$ 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

	Network Type				
Alphabet Size	Simple RNN	Trans-former	GRU	LSTM	2-Layer LSTM
4	41,102	114,030	122,102	161,702	322,502
16	42,302	115,230	123,302	162,902	323,702
64	47,102	120,030	128,102	167,702	328,502

Tensorflow and Keras APIs were used to implement all NNs (Abadi *et al.*, 2015).

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
- token embedding dimension = 100
- linear layer and softmax activation
- recurrent or attention module

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

1. Do the performance measures correlate with each other?

Spearman's ρ

	Accuracy	AUC	Brier
AUC	0.9500		
Brier	−0.9396	−0.8968	
F-score	0.8727	0.8335	−0.8107

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

Sanity Checks

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

	Accuracy	all-pairs p-value
SL	0.794	1.00
coSL	0.799	
SP	0.739	1.00
coSP	0.739	
TSL	0.808	0.937
TcoSL	0.820	

YES. While the Friedman test reveals that language class does have a significant effect on accuracy ($p < 2.2e-16$), the post-hoc all-pairs test shows no significant difference between these particular pairs of language classes.

Sanity Checks

3. Does more training data help?

Accuracy by Train Set (Friedman p < 2.2e-16)

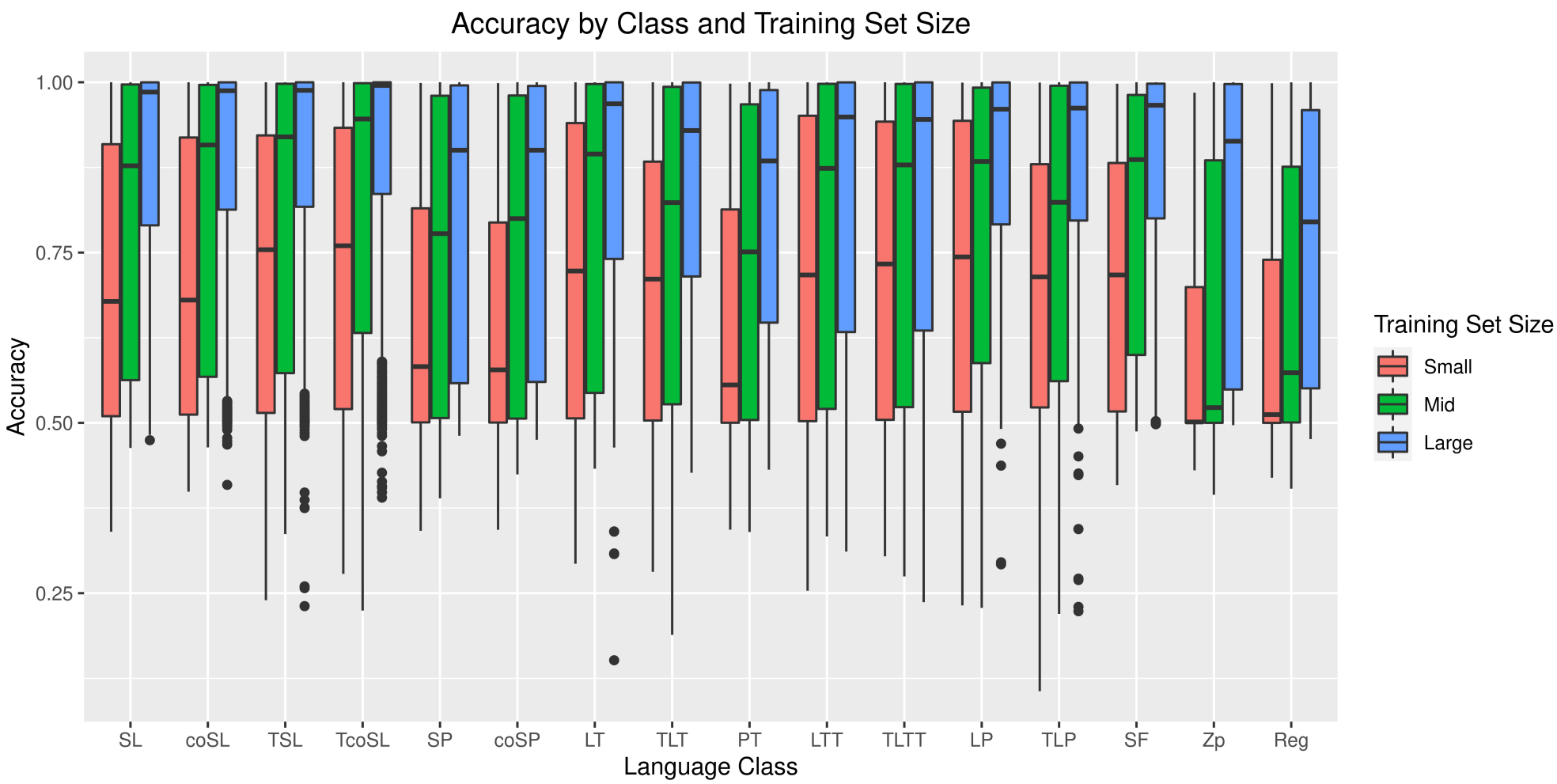
Small	Mid	Large
0.696	0.773	0.844

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

	Large	Mid
Mid	$< 2e-16$	-
Small	$< 2e-16$	$< 2e-16$

YES. Accuracy improves with more training.

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?

Accuracy by Test Set (Friedman p < 2.2e-16)

SR	LR	SA	LA
0.898	0.859	0.673	0.653

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

	LA	LR	SA
LR	$< 2e-16$	-	-
SA	$3.3e-14$	$< 2e-16$	-
SR	$< 2e-16$	$8.2e-14$	$< 2e-16$

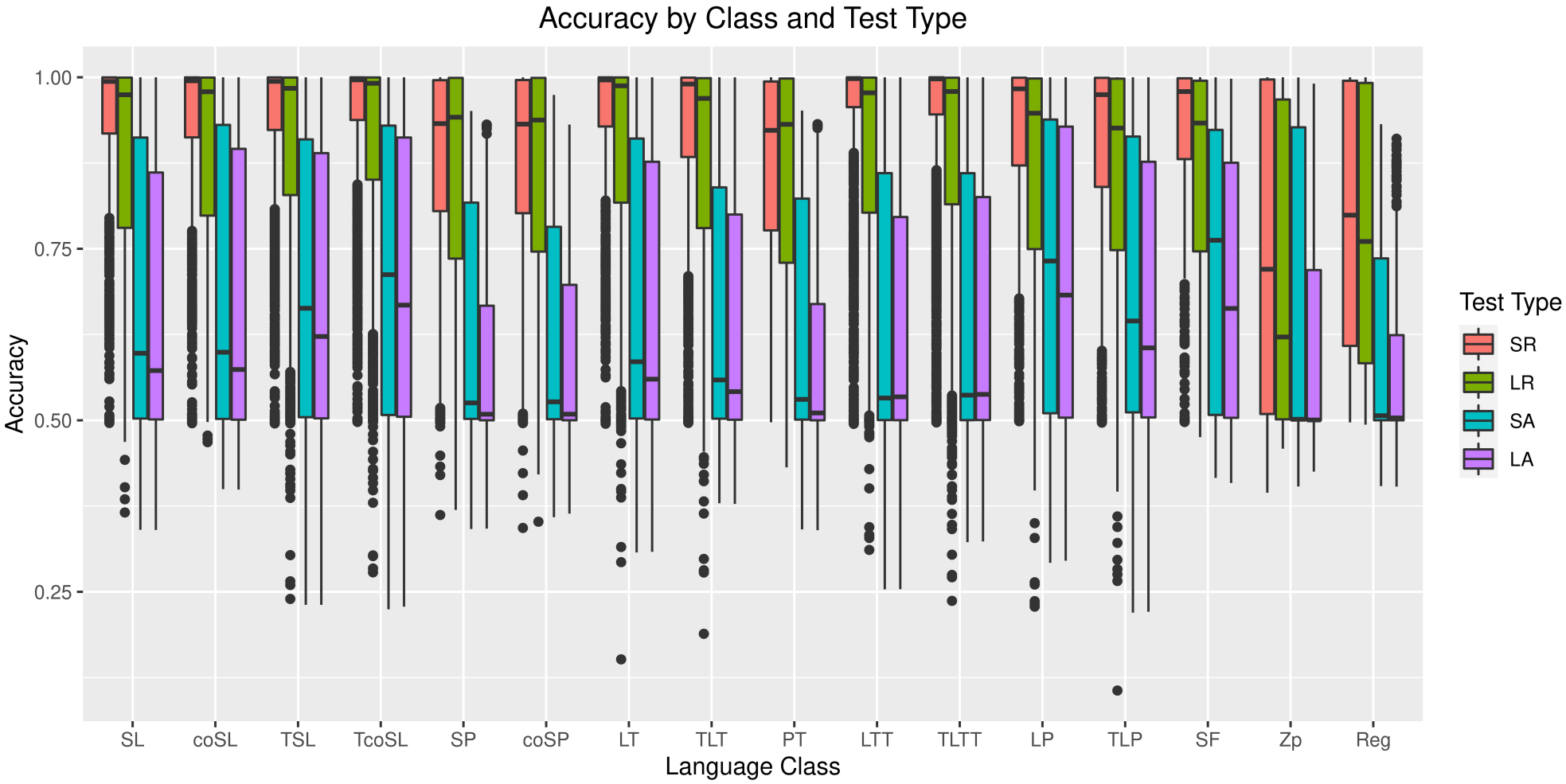
What is the effect of the test set?

Accuracy by Test Set (Friedman p < 2.2e-16)

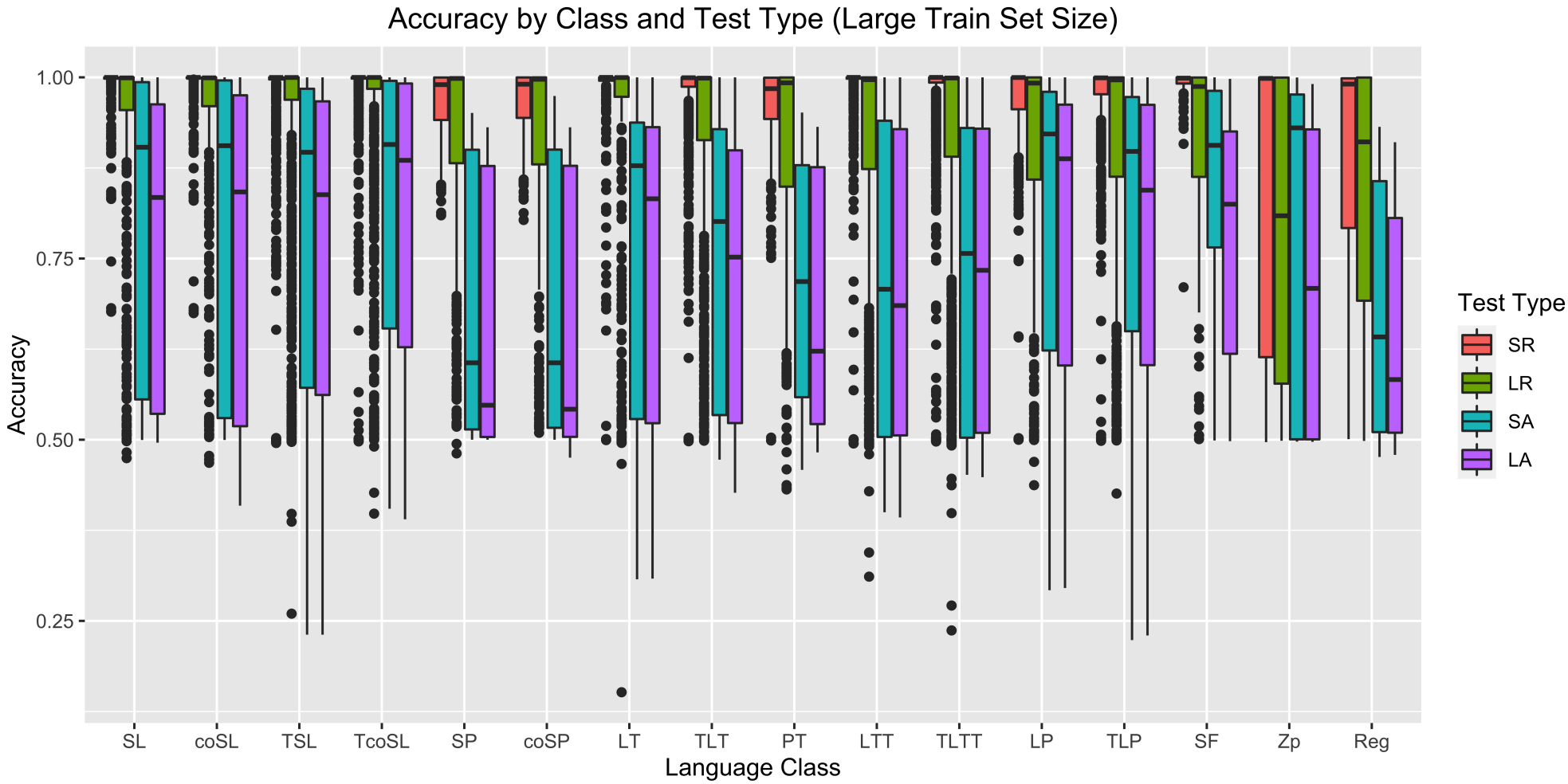
SR	LR	SA	LA
0.898	0.859	0.673	0.653

Pair	Effect Size (Cohen's d)	Interpretation
SR, LR	0.279	Small
LR, SA	1.142	Large
SA, LA	0.084	Negligible

Accuracy by Class and Test Set



Accuracy by Class and Test Set



What is the effect of logical level?

Accuracies by Logical Level (Friedman p < 2.2e-16)

CNL	DPL	FO	PROP	REG
0.787	0.795	0.785	0.781	0.683

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

	CNL	DPL	FO	PROP
DPL	0.0077	-	-	-
FO	0.9946	0.0277	-	-
PROP	0.3172	$1.7e-06$	0.1448	-
REG	$7.9e-14$	$< 2e-16$	$5.2e-14$	$1.1e-08$

There is a clear divide between MSO and everything else.

What is the effect of order relation?

Accuracy by Order Relation (Friedman p=2.244e-08)

SUCC	TSUCC	PREC	OTHER
0.790	0.795	0.744	0.775

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

	OTHER	PREC	SUCC
PREC	0.075	-	-
SUCC	0.025	1.0e-06	-
TSUCC	0.022	8.3e-07	1.000

NN performance decreases significantly as one goes from successor and tier-successor to precedence.

What is the effect of alphabet size?

Accuracies by Alphabet Size (Friedman p < 2.2e-16)

4	16	64
0.738	0.745	0.829

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

	4	16
16	$7e-04$	-
64	$< 2e-16$	$< 2e-16$

Patterns embedded in larger alphabets are easier to learn.

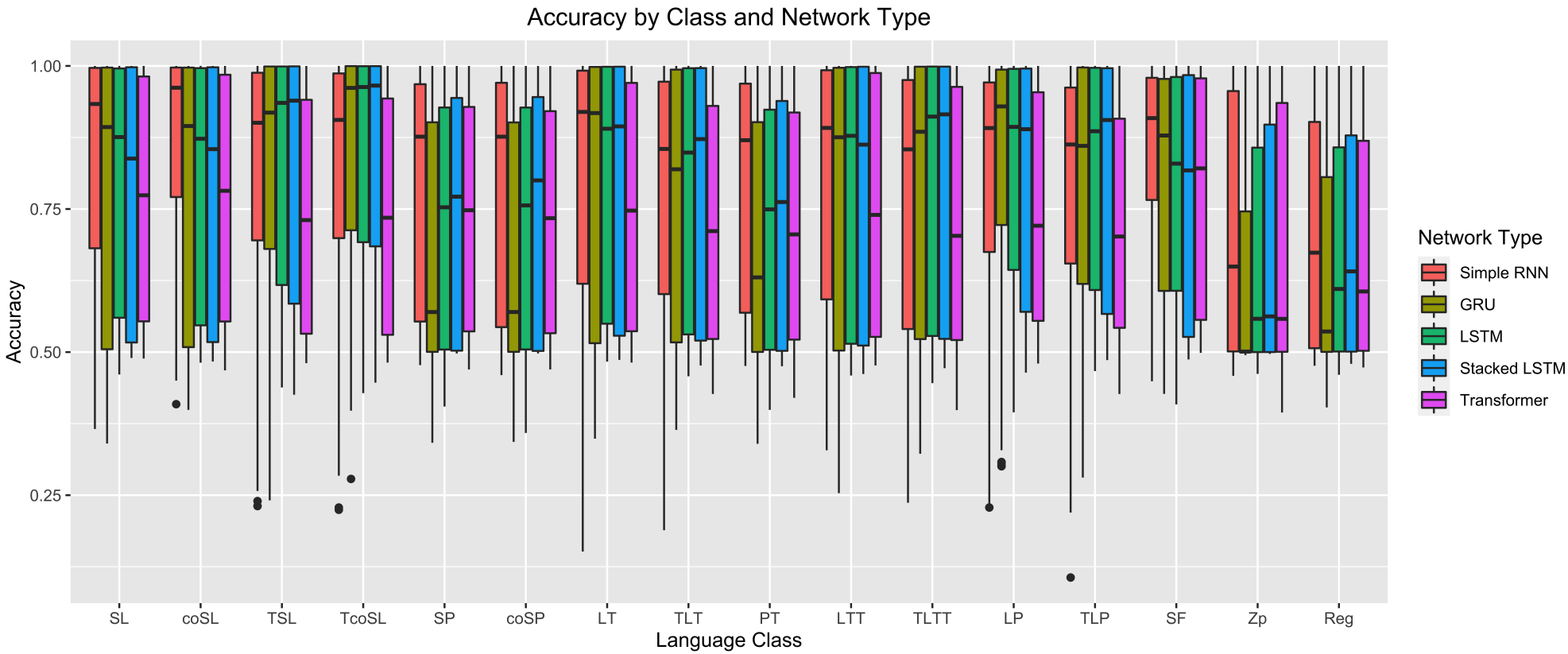
What is the effect of the neural network?

Accuracy by NN and Training Set

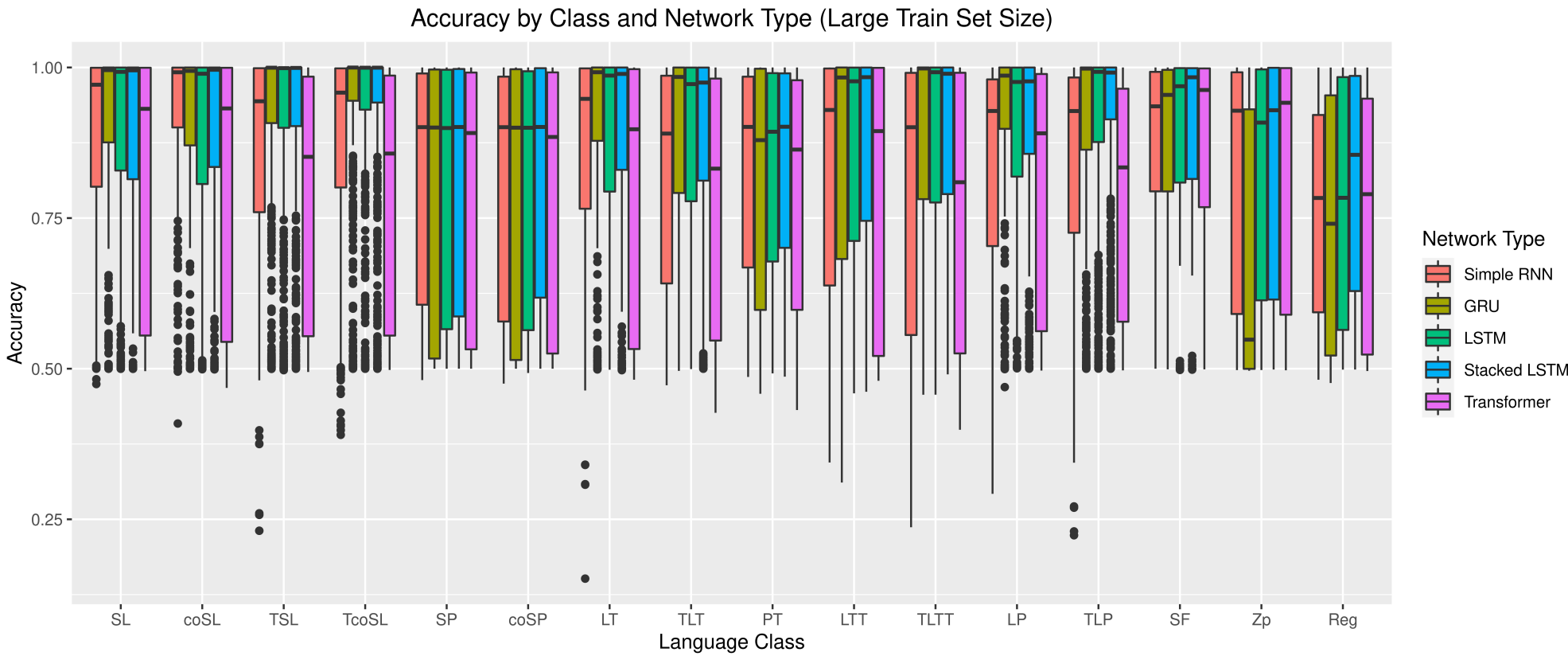
	Small	Mid	Large
Simple RNN	**0.765**	**0.805**	0.836
GRU	0.702	0.735	0.851
LSTM	0.675	**0.789**	0.863
2-layer LSTM	0.675	**0.783**	**0.871**
Transformer	0.663	0.752	0.799

Within each training set, the Friedman test is significant ($p < 2.2e\text{-}16$). According to the all-pairs test, the bold-faced scores are significantly better than the non-bold ones, and not significantly different from each other.

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	-0.063	-0.138
monoid size $\sim$ accuracy	-0.047	-0.112

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
-0.433	-0.392

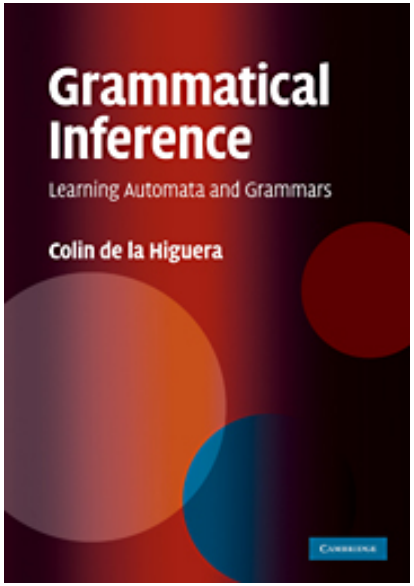
Lessons from these experiments on ML-RegTest

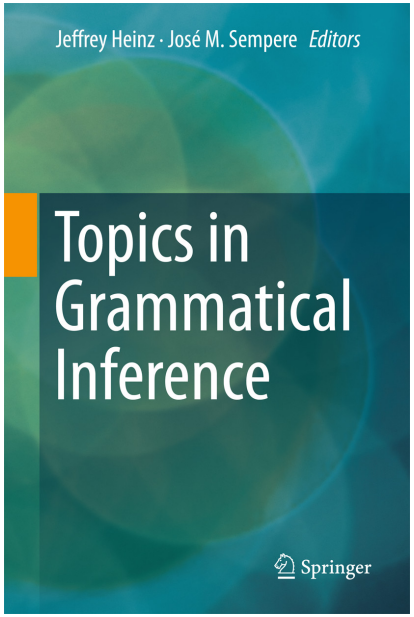
1. High performance on Random Test sets does not imply correct generalization as measured by performance on the Adversarial tests.
2. Classification of patterns which count modulo n is hard for neural ML systems to learn from.
3. Classification of patterns which can be described with the precedence relation is harder for neural ML systems to learn
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.

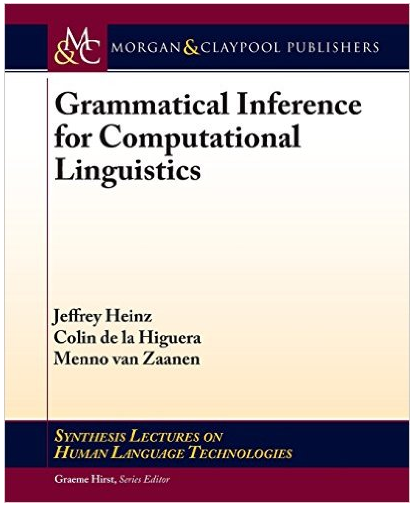
Broader Lessons

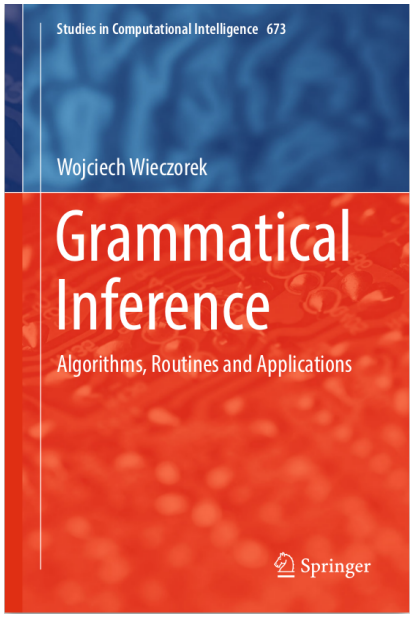
1. Experiments on artificial data sets can help
   - give confidence that patterns of certain types can be found with particular ML methods.
   - So for NNs, they help open the black box.
2. Classes of formal languages exemplify different kinds of patterns.
3. The same kind of work can be conducted for regression tasks on strings, string-to-string functions and so on.

Grammatical Inference









ICGI 2023 in Rabat, Morocco July 10-13!!

http://www.fsr.ac.ma/icgi2023/

https://grammarlearning.org/

Thank you!

MLRegTest

https://github.com/heinz-jeffrey/subregular-learning

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/

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.

Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61:85–117.

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