Subregular Complexity and Machine Learning

Jeffrey Heinz

Linguistics Department
Institute for Advanced Computational Science
Stony Brook University

IACS Seminar September 14, 2017

Joint work

- Enes Avcu, University of Delaware
- Professor Chihiro Shibata, Tokyo University of Technology

*This research was supported by NIH R01HD87133-01 to JH, and JSPS KAKENHI 26730123 to CS.

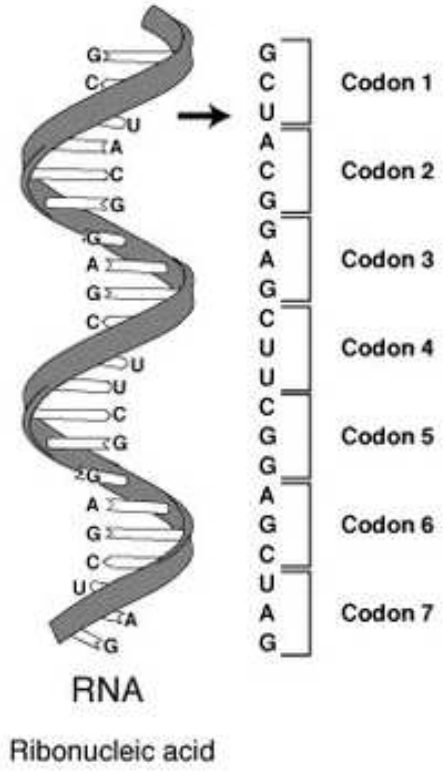
Charles Babbage

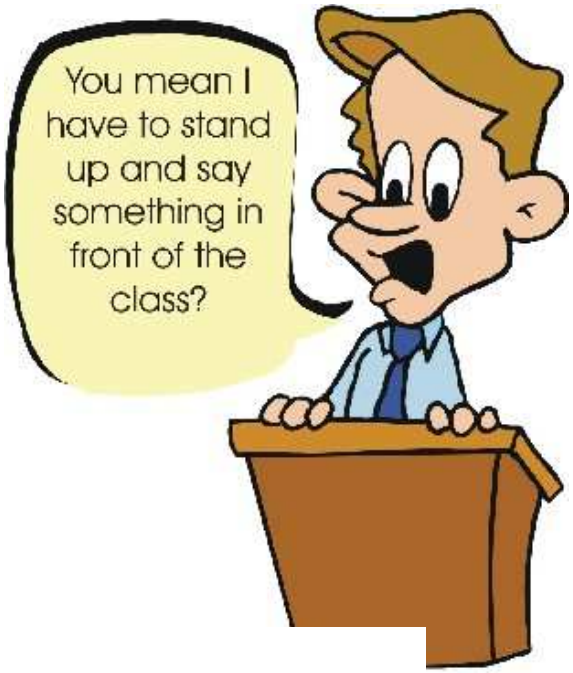
"On two occasions I have been asked [by members of Parliament], 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question."

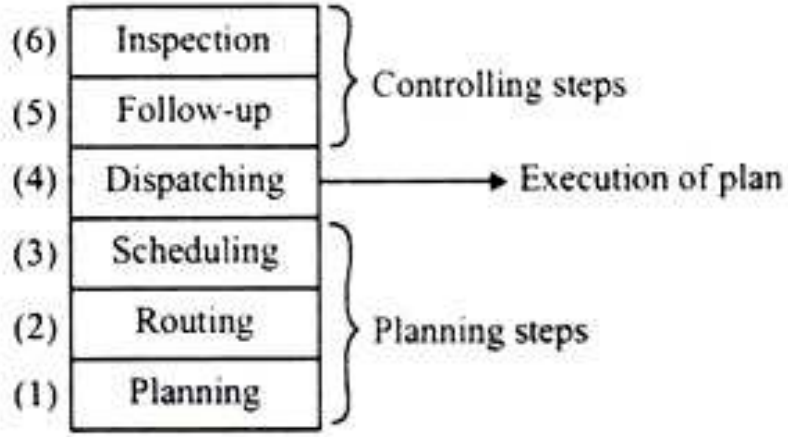
as quoted in de la Higuera 2010, p. 391

Sequences in nature and engineering

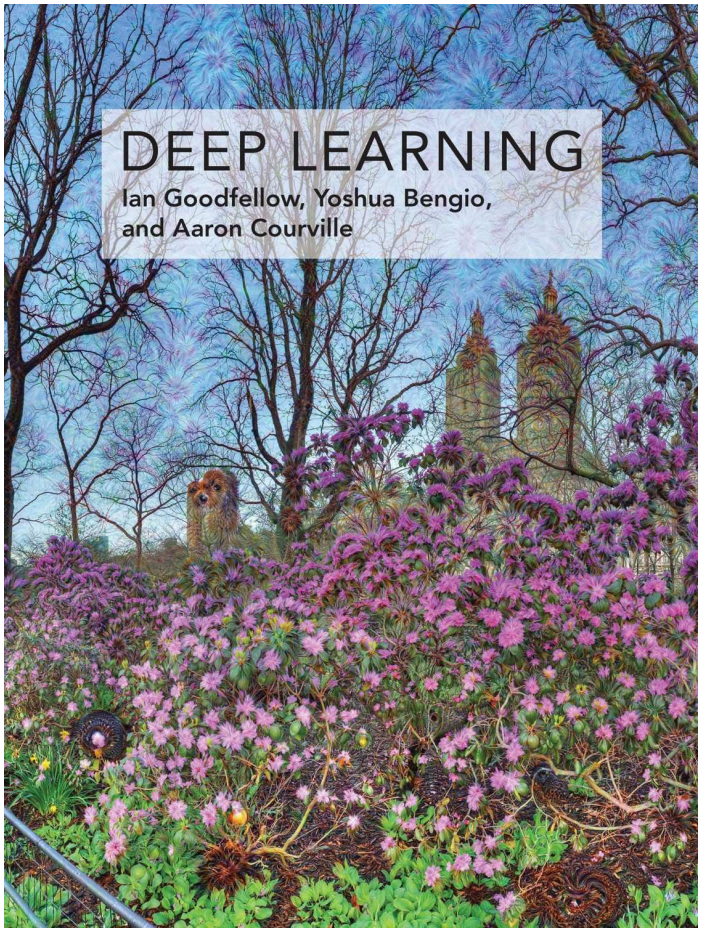
1. Natural languages
2. Nucleic acids
3. Planning and executing actions
4. ...



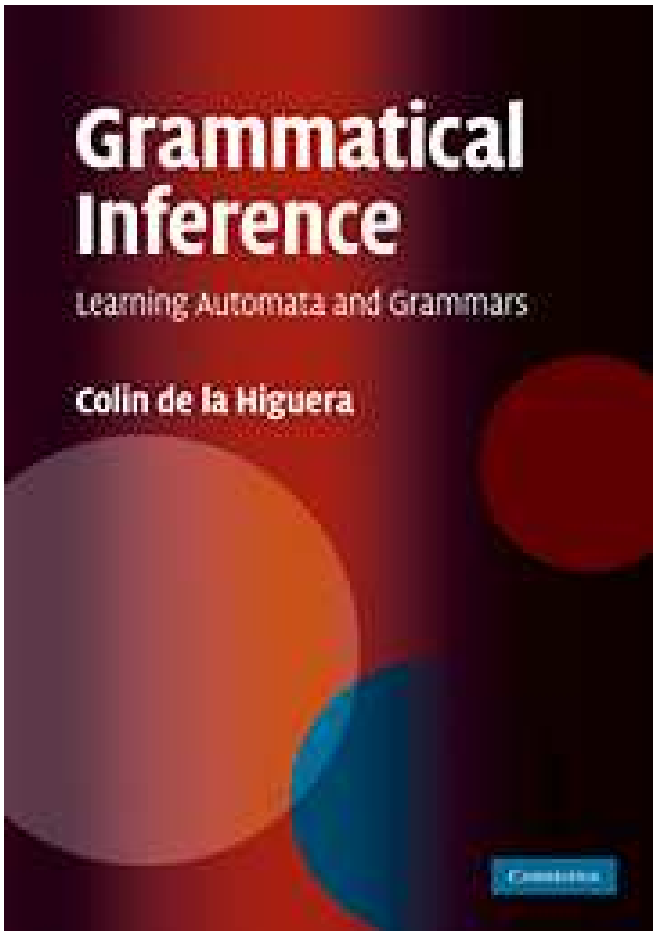




This talk: A tale of two approaches to learning



and



This talk

The judicial use of formal language theory and grammatical inference (GI) can help illuminate the kinds of generalizations deep learning networks can and cannot make.

Contributions

1. Simple *regular* languages discriminate naive LSTMs' ability to generalize. Ultimate goal would try to formalize this relationship.
2. GI algorithms can help us understand whether sufficient information is present for successful learning to occur.

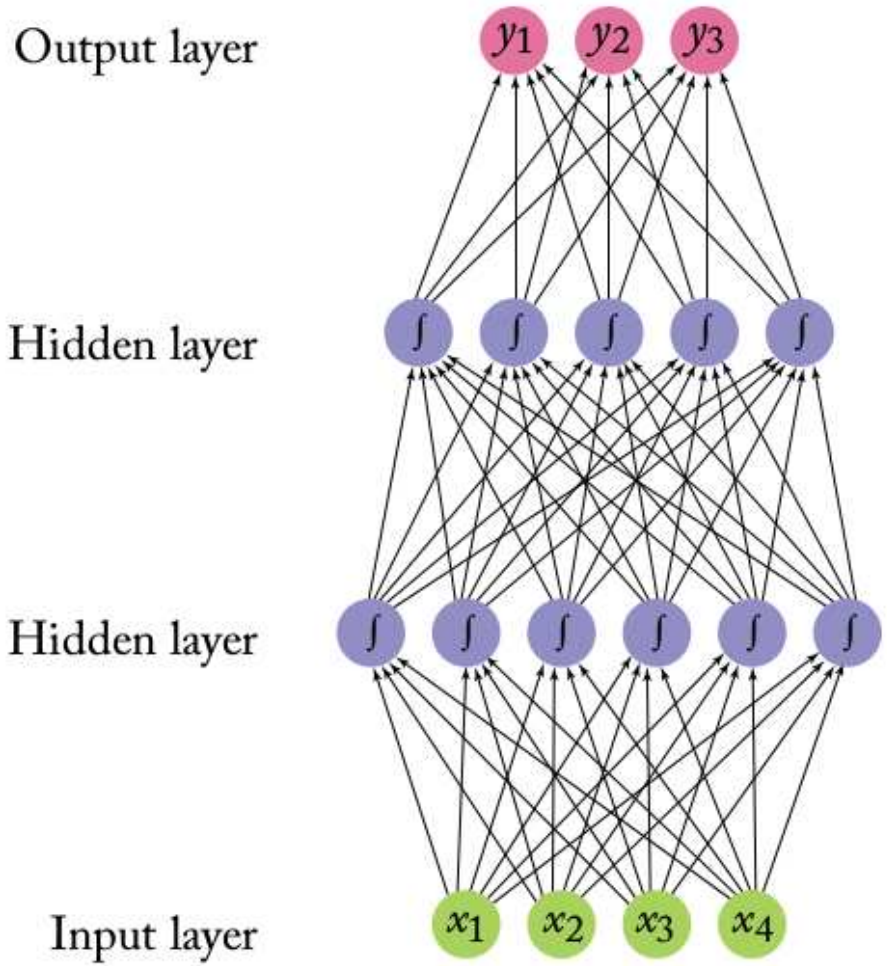
Success of Deep Learning

"Our deep learning methods developed since 1991 have transformed machine learning and Artificial Intelligence (AI), and are now available to billions of users through the five most valuable public companies in the world: Apple (#1 as of 9 August 2017 with a market capitalization of US$ 827 billion), Google (Alphabet, #2, 654bn), Microsoft (#3, 561bn), Facebook (#4, 497bn), and Amazon (#5, 475bn) [1]."

Jürgen Schmidhuber, IDSIA

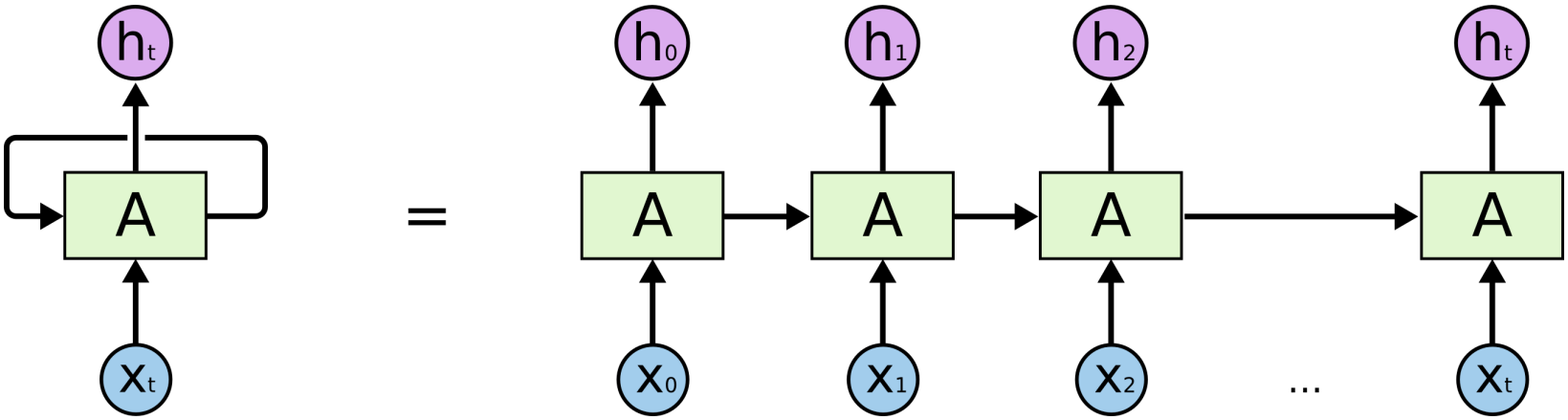
`http://people.idsia.ch/~juergen/impact-on-most-valuable-companies.html`

Feed-forward neural network with two hidden layers



(Goldberg 2017, page 42)

Recurrent Neural Networks (RNNs) add a loop



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

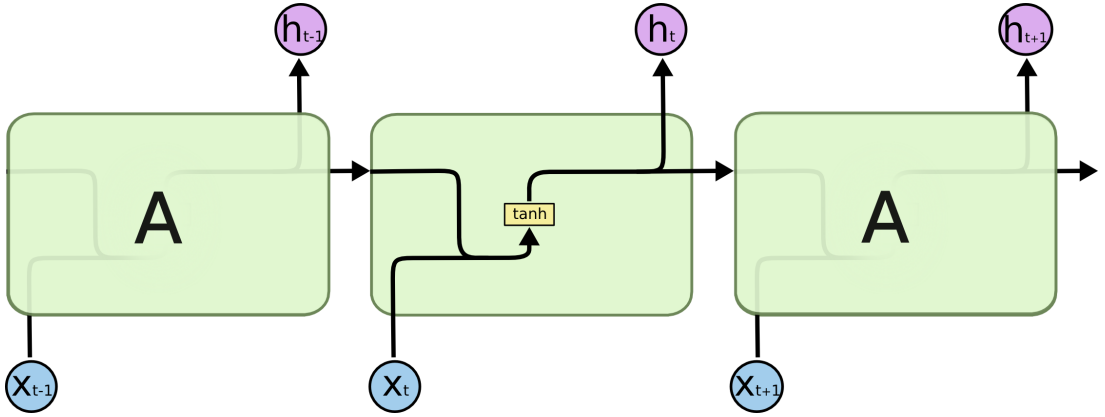
Success of Deep Learning

"Most work in machine learning focuses on machines with reactive behavior. RNNs, however, are more general sequence processors inspired by human brains. They have adaptive feedback connections and are in principle as powerful as any computer. The first RNNs could not learn to look far back into the past. But our 'Long Short-Term Memory' (LSTM) RNN overcomes this fundamental problem, and efficiently learns to solve many previously unlearnable tasks."

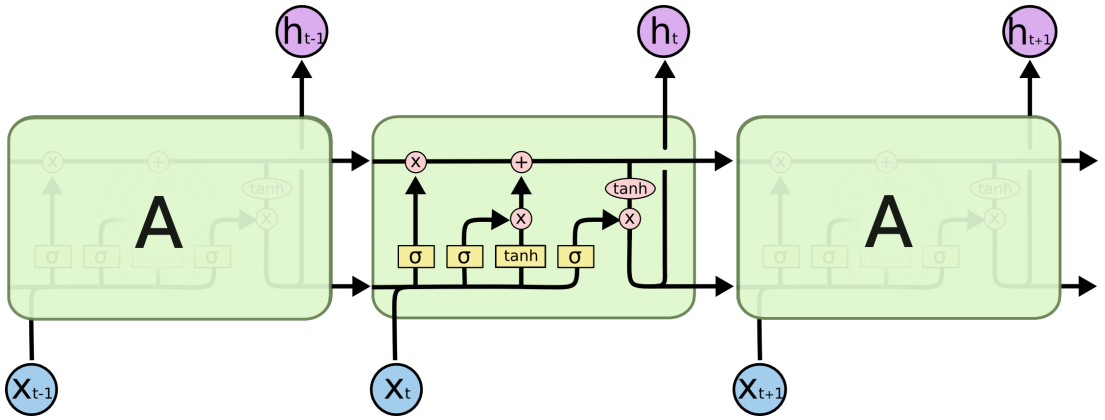
`http://people.idsia.ch/~juergen/`

Jürgen Schmidhuber, IDSIA

RNNs



LSTMs



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Success of Deep Learning

"LSTM-based systems can learn to translate languages, control robots, analyse images, summarise documents, recognise speech and videos and handwriting, run chat bots, predict diseases and click rates and stock markets, compose music, and much more, ..."

Jürgen Schmidhuber, IDSIA

`http://people.idsia.ch/~juergen/impact-on-most-valuable-companies.html`

A contrarian view

"Even the trendy technique of 'deep learning,' which uses artificial neural networks to discern complex statistical correlations in huge amounts of data, often comes up short. Some of the best image-recognition systems, for example, can successfully distinguish dog breeds, yet remain capable of major blunders, like mistaking a simple pattern of yellow and black stripes for a school bus. Such systems can neither comprehend what is going on in complex visual scenes ('Who is chasing whom and why?') nor follow simple instructions ('Read this story and summarize what it means')."

Gary Marcus, NYU

NY Times, Sunday Review, July 29, 2017

Rest of the talk

1. Formal Language Theory
2. Grammatical Inference
3. Learning Experiments
4. Discussion

Sequences, Strings

$$\vdots$$

aaa, aab, aba, abb, baa, bab, bba, bbb

aa, ab, ba, bb

a, b

λ

A string is a finite sequence of symbols from some set of symbols Σ.

Formal languages, sets of strings

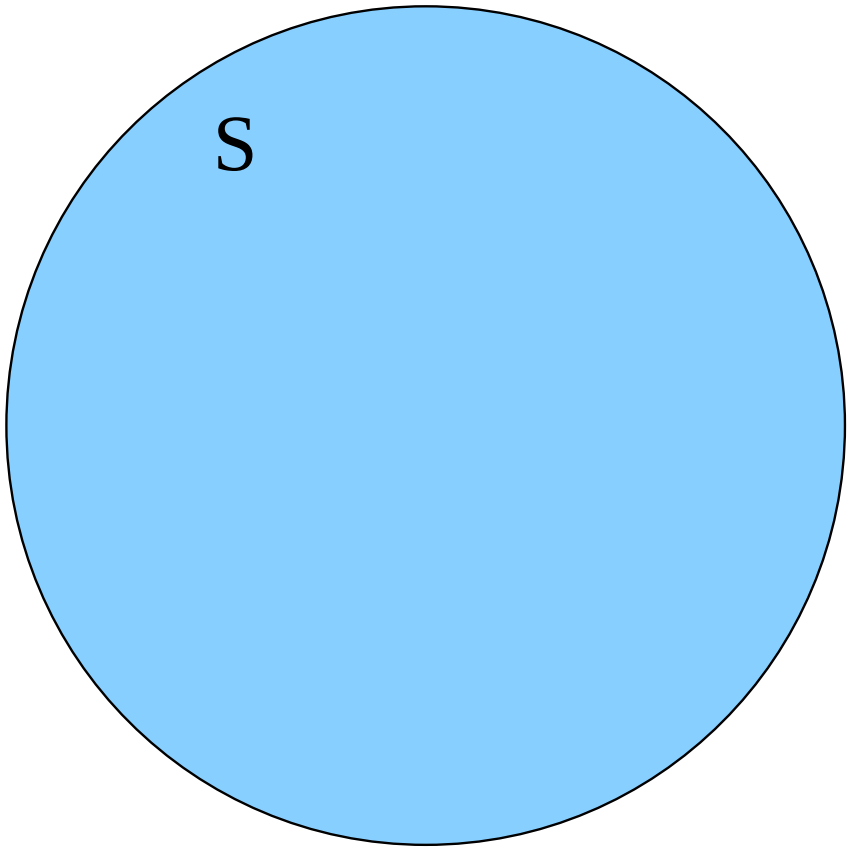
The set of all possible strings is notated Σ^*.

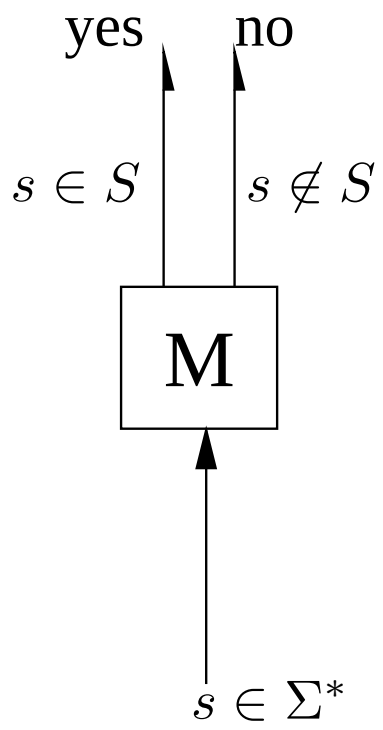
Every subset of Σ^* is a formal language.

Examples

1. Let $\Sigma = \{$ `a,b,c,...,` `z,` $\square$`,` `.` $\}$. Then there is a subset of Σ^* which includes all and only the grammatical sentences of English (modulo capitalization and with $\square$ representing spaces).
2. Let $\Sigma = \{$`Advance-1cm,` `Turn-R-5`$^\circ\}$. Then there is a subset of Σ^* which includes all and only the ways to get from point A to point B.
3. ...

The membership problem



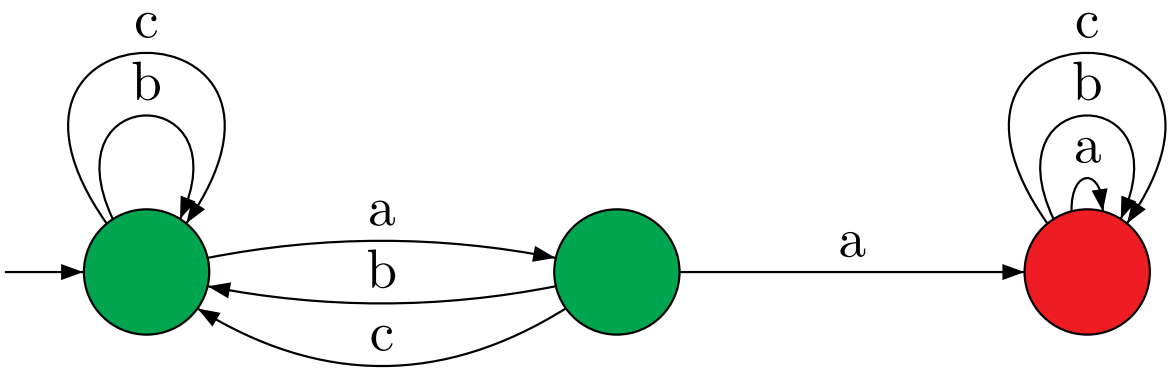


Given a set of strings S and any string s, output whether $s \in S$.

Example 1

A string belongs to S if it does not contain `aa` as a substring.

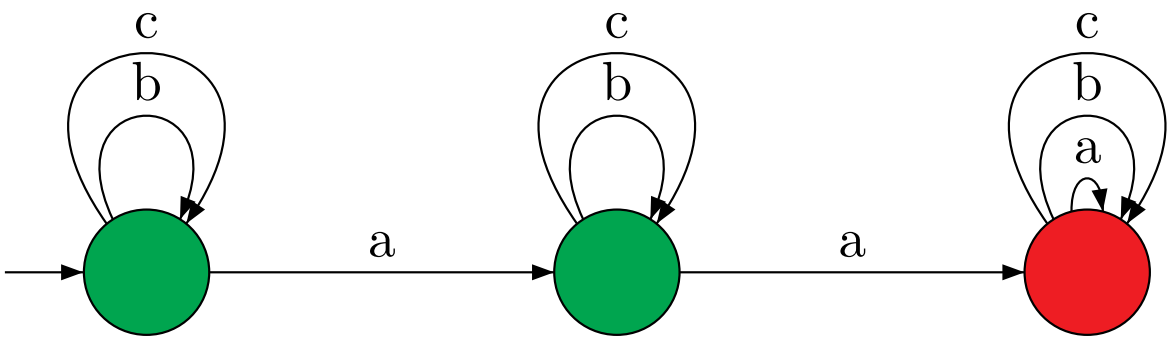
$s \in S$	$s \notin S$
abba	baab
abccba	aaccbb
babababa	ccaaccaacc
...	...



Example 2

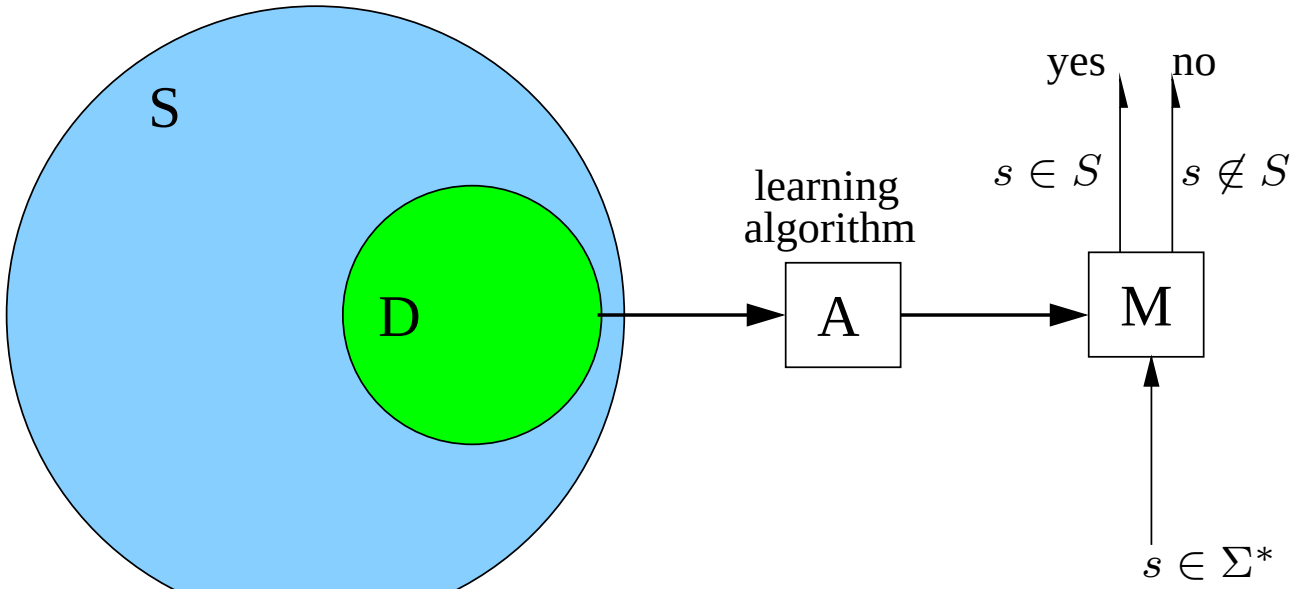
A string belongs to S if it does not contain `aa` as a *subsequence*.

$s \in S$	$s \notin S$
cabb	baab
babccbc	babccba
bbbbbb	bbacccccccccaccc
...	...



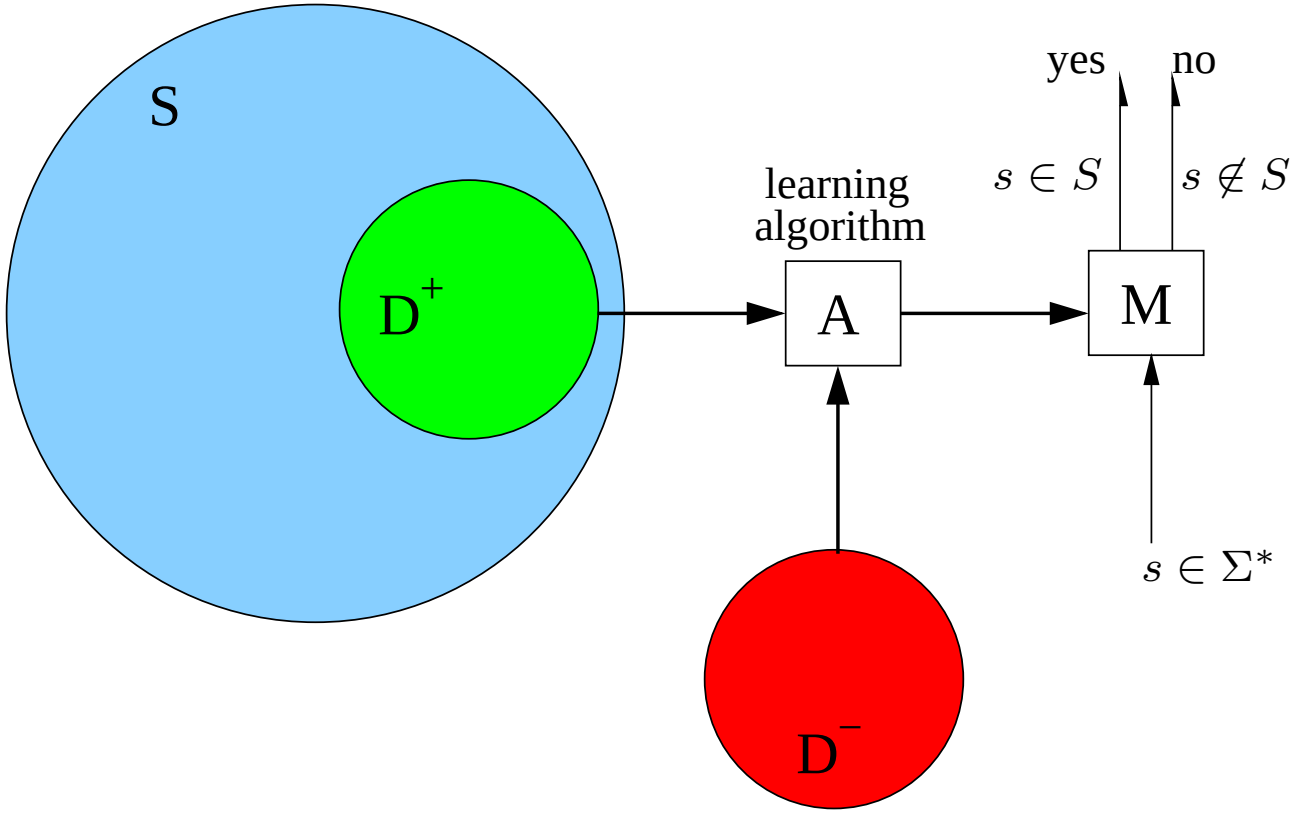
A Learning Problem: Positive Evidence Only

For any set S from some given collection of sets: Drawing finitely many example strings from S, output a program solving the membership problem for S.



A Learning Problem: Positive and Negative Evidence

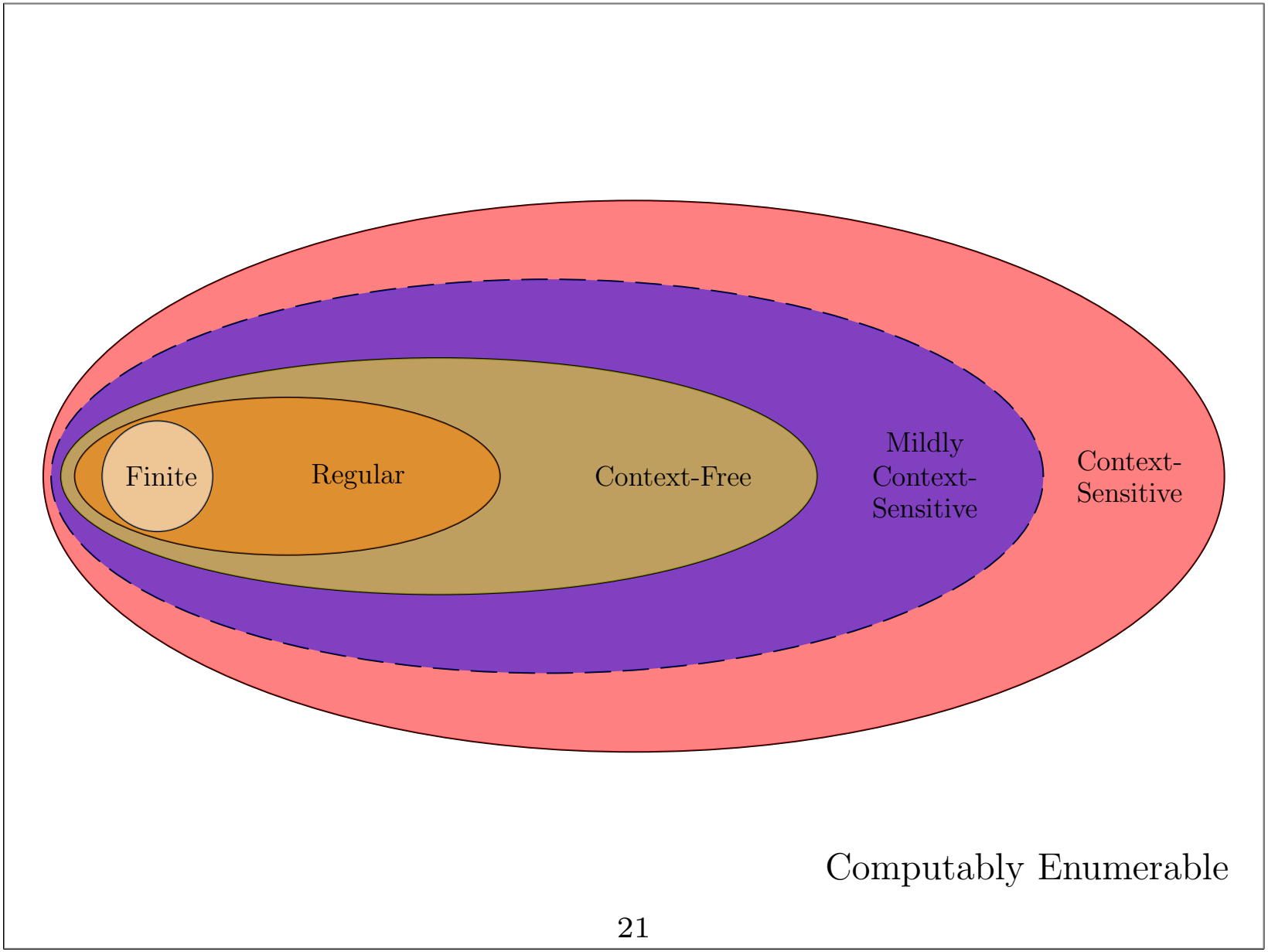
For any set S from some given collection of sets: Drawing finitely many strings labeled as to whether they belong to S or not, output a program solving the membership problem for S.



Generalizing the Membership and Learning Problems

function	Notes
$f : \Sigma^* \to \{0, 1\}$	Binary classification
$f : \Sigma^* \to \mathbb{N}$	Maps strings to numbers
$f : \Sigma^* \to [0, 1]$	Maps strings to real values
$f : \Sigma^* \to \Delta^*$	Maps strings to strings
$f : \Sigma^* \to \wp(\Delta^*)$	Maps strings to sets of strings

Classifying membership problems (1)



RPNI: Regular Positive and Negative Inference

Theorem. For every regular language S, there is a finite set $D^+ \subseteq S$ and a finite set $D^- \not\in S$ such that when the algorithm RPNI takes *any* training sample containing D^+ and D^- as input, RPNI outputs a program which solves the membership problem for S. Furthermore, RPNI is efficient in both time and data.

(Oncina and Garica 1992, de la Higuera 2010)

How does RPNI work?

1. RPNI first builds a finite state machine representing the training sample called a "prefix tree."
2. It iteratively tries to merge states in a breadth-first manner, testing each merge against the training sample.
3. It keeps merges that are consistent with the sample and rejects merges that are not.
4. At the end of this process, if the training data was sufficient then the resulting finite-state machine is guaranteed to solve the membership problem for S.

Let's use formal languages to study LSTMs

1. Grammars generating the formal languages are known.
   (a) Conduct controlled experiments.
   (b) Ask specific questions. Example: To what extent are the generalizations obtained independent of string length?
2. Relative complexity of different formal languages may provide additional insight.
3. Grammatical inference results can inform whether the data was rich enough.
4. May lead to proofs and theorems about abilities of types of networks
5. May lead to new network architectures.

Valid idea then, valid now

1. Predicting the next symbol of a string drawn from a regular language
   - Network Type: First-order RNNs,
   - Target language: Reber Grammar (Reber 1967)
   - (Casey 1996; Smith, A.W. 1989)
2. Deciding whether a string s belongs to a regular language S
   - Network Type: Second-order RNNs,
   - Target language: Tomita languages (Tomita 1982).
   - (Pollack 1991; Watrous and Kuhn 1992; Giles et al. 1992)

Later research targeted nonregular languages

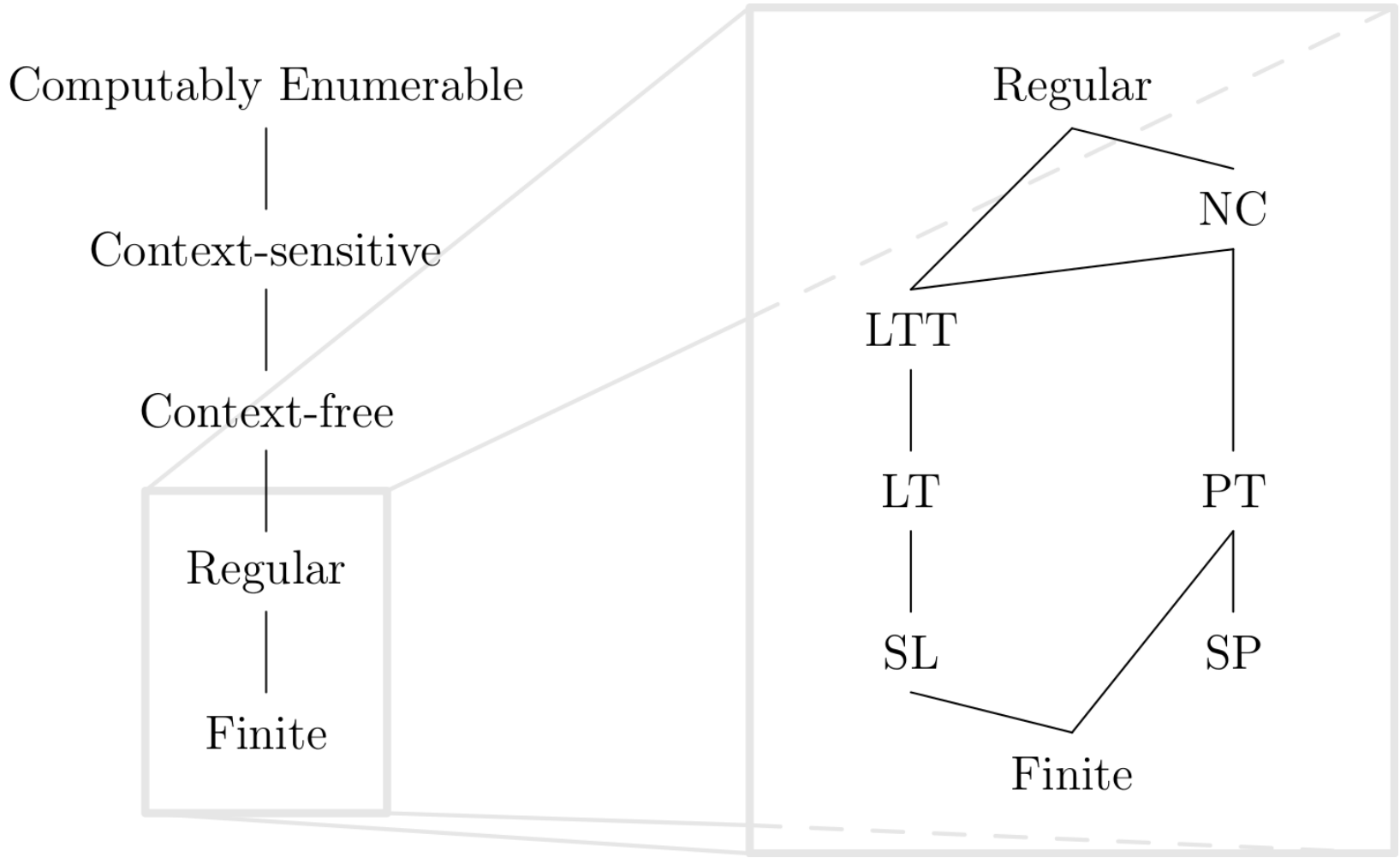
- LSTMs correctly predicted the possible continuations of prefixes in words from $a^n b^n c^n$ for n up to 1000 and more.
- (Schmidhuber et al. 2002; Chalup and Blair 2003; Prez- Ortiz et al. 2003).

Additional Motivation for current study: Subregular complexity

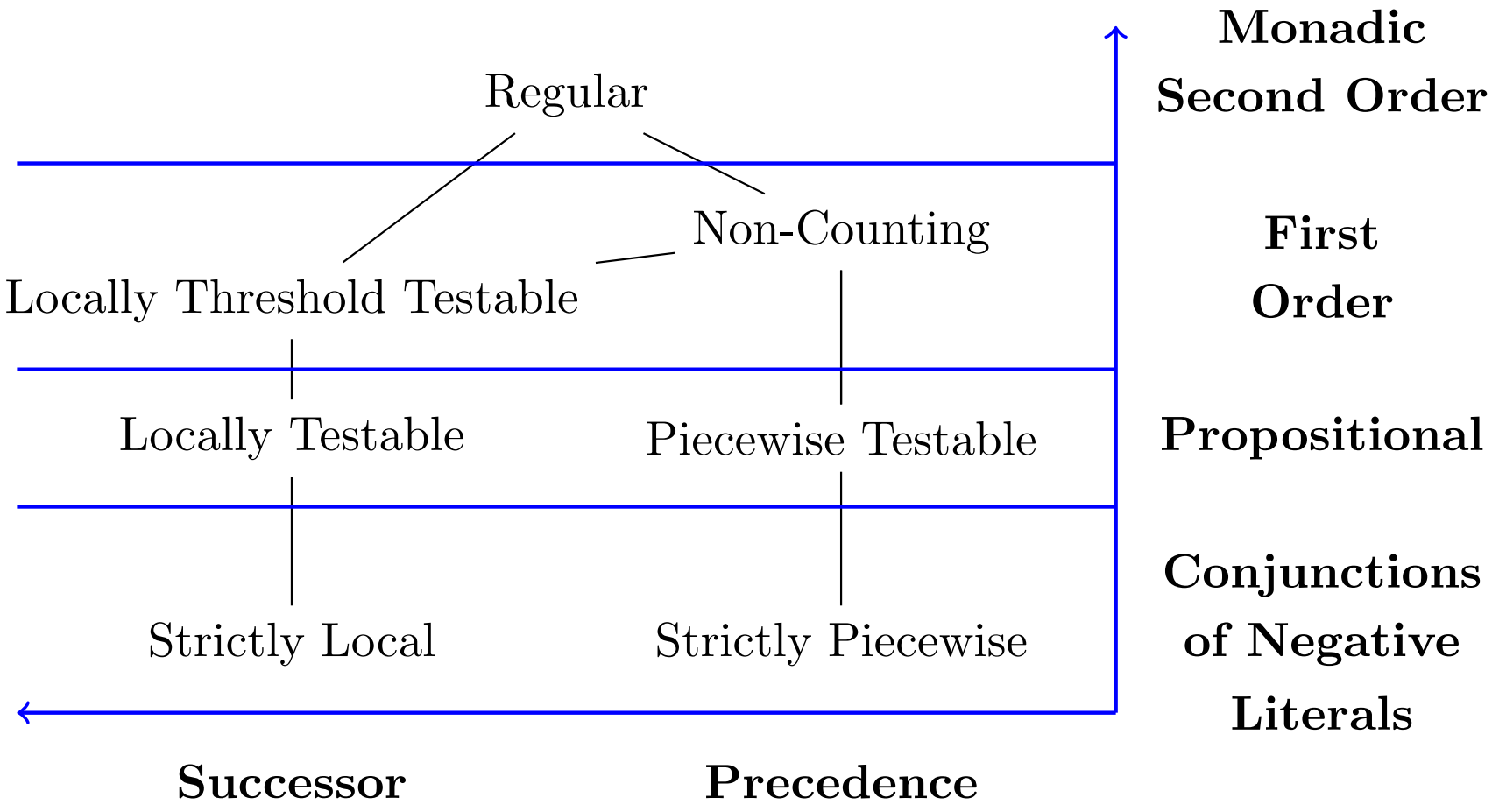
The Reber grammars and Tomita languages were not understood in terms of their abstract properties or pattern complexity.

- Regular languages chosen here are known to have certain properties based on their subregular complexity (McNaughton and Papert 1971, Rogers and Pullum 2011, Rogers et al. 2010, 2013).

Classifying membership problems (2)



Classifying membership problems (3)



(McNaughton and Papert 1971, Heinz 2010, Rogers and Pullum 2011, Rogers et al 2010, 2013)

Subregular complexity

These classes are natural because they have multiple characterizations in terms of logic, automata, regular expressions, and abstract algebra.

1. SL is the formal language-theoretic basis of n-gram models (Jurafsky and Martin, 2008),
2. SP can model aspects of long-distance phonology (Heinz, 2010; Rogers et al. 2010)

SL Characterizations

1. S is closed under suffix substitution. $S \in SL$ if there exists k such that for all u, v, w, x, y if uvw and xvy belong to L and v has length $k-1$ then uvy belongs to S too.
2. SL stringsets can be defined with a finite set of forbidden substrings.
3. From a model-theoretic perspective, the order of elements in strings is represented with the successor (+1) relation and the logical formula conjoin negative literals.

SP Characterizations

1. S belongs to SP iff S is closed under subsequence.
2. SP stringsets can be defined with a finite set of forbidden subsequences.
3. From a model-theoretic perspective, the order of elements in strings is represented with the precedence ($<$) relation and the logical formula conjoin negative literals.

GI results for SL and SP with positive data only

Theorem. For each k, there is an algorithm A such that, for all $S \in$ SL-k (SP-k), there is a finite subset $D \subseteq S$ such that when A takes any finite superset of D as input, A outputs a program which solves the membership problem for S.

Six Target Languages

$\Sigma = \{a, b, c, d\}$.

Language Class	Forbidden k-substrings in target stringsets	DFA size
SL2	⋊b, aa, bb, a⋉	3
SL4	⋊bbb, aaaa, bbbb, aaa⋉	7
SL8	⋊bbbbbbb, aaaaaaaa, bbbbbbbb, aaaaaaa⋉	15
Language Class	Forbidden k-subsequences in target stringsets	
SP2	ab	2
SP4	abba	4
SP8	abbaabba	8

Training

The six target languages were implemented with finite state machines using `foma`, a publicly available, open-source platform (Hulden, 2009).

Training Data

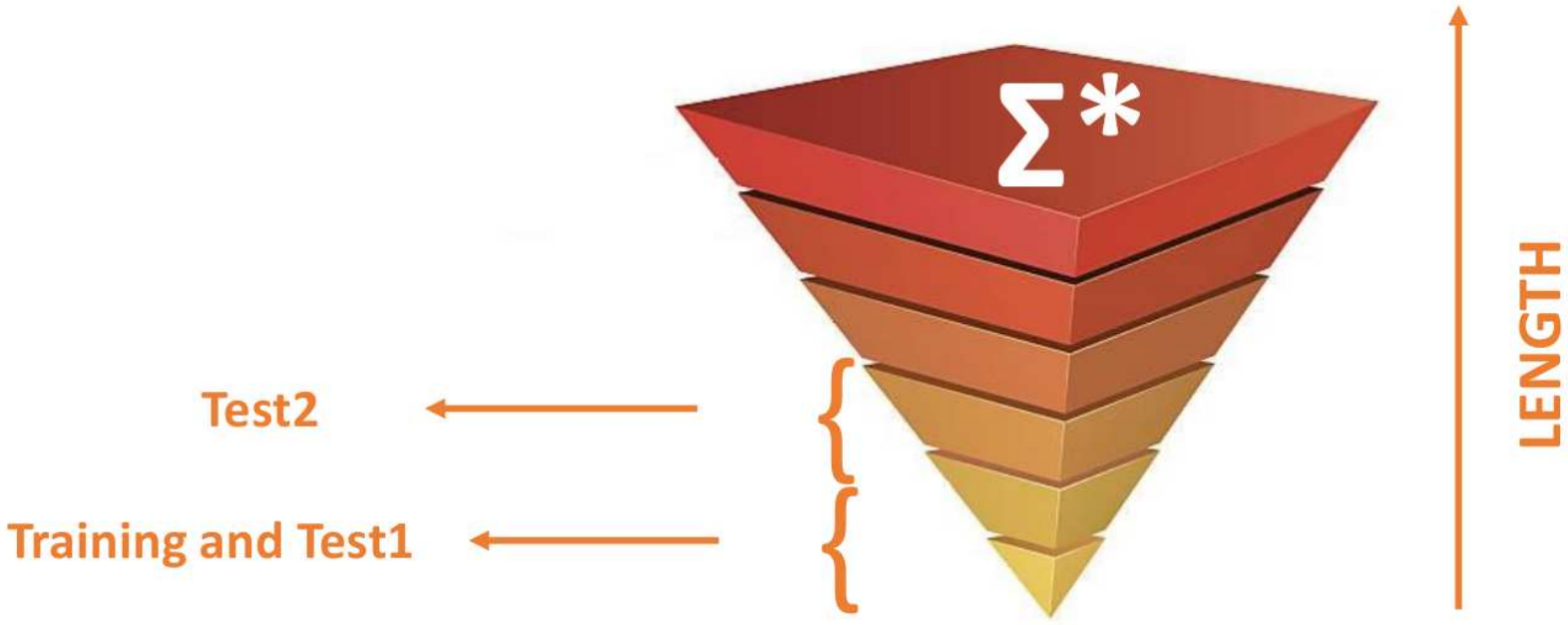
- Three training data sets: 1k, 10k, and 100k.
- Half of the words in each training set were positive examples and the other half were negative examples.
- Training words were generated for each length between 1 and 25.
- Training sets were generated randomly with `foma` and thus may contain duplicates.

Test Data

We developed two test sets: Test1 and Test2.

- Each contain 1k, 10k, 100k novel words (so not in training).
- Half of the test words belong to L and half do not.
- Test1: The length of the words are no longer than 25.
- Test2: Test2 words are of length between 26 and 50.

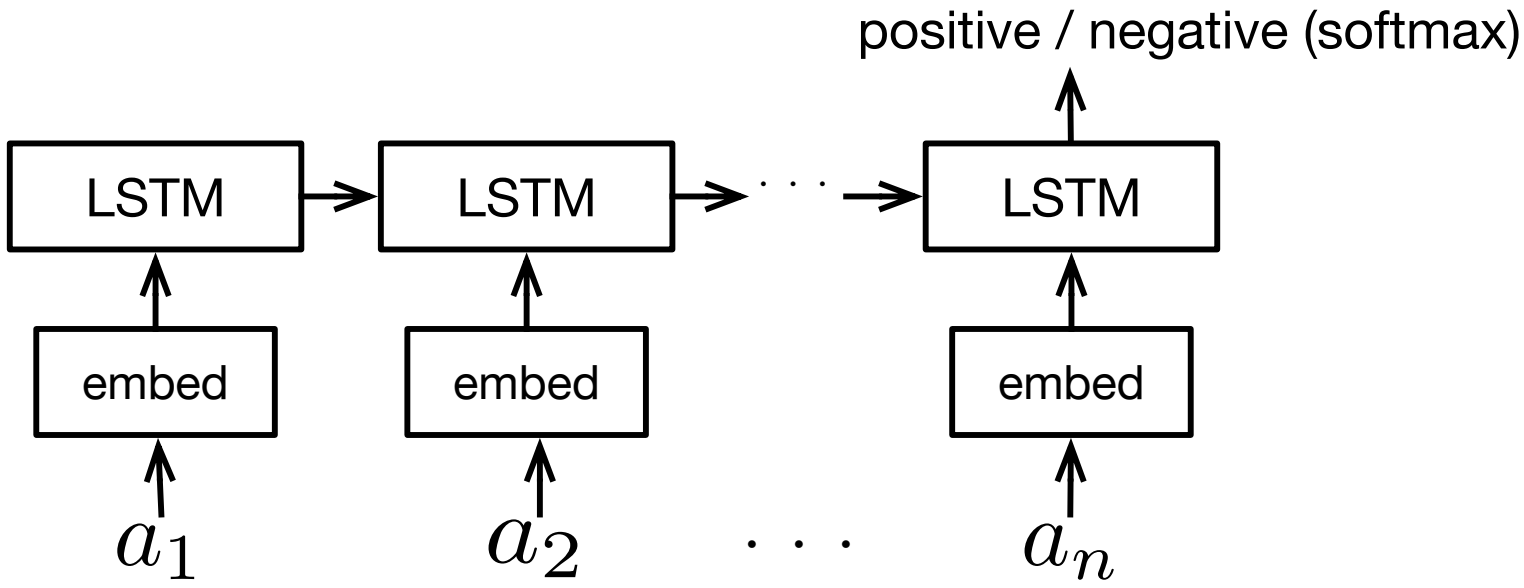
Test Data



The LSTMs in these experiments

- We constructed simple LSTM networks to test the capability of the LSTM itself.
- We used a package for RNNs called Chainer (`http://chainer.org`) to implement them.

LSTM architecture



- The embed layer maps the one-hot vector of each symbol to a real-valued vector.
- All outputs of the LSTM except for the last one are ignored.
- The last output of the LSTM is input to the softmax layer.
- The output of the softmax layer represents the positive and negative probability.

Learning Parameters

- Three fully connected LSTMs: vector sizes were 10, 30, and 100 for all layers.
- The weights of the forget gate of LSTM are initialized according to the normal distribution with mean 1.
- The batch size is 128.
- The L2 norm of the gradient is clipped with 1.0.
- The optimization algorithm is Adam (Kingma et. al, 2014) .
- The lengths of strings in each batch are aligned through padding with the zero vector.
- . . . many choices, but basically done in a standard fashion

SL Results

< 0.60 < 0.80 < 0.99

		LSTM type					
	Training	v10		v30		v100	
	Regimen	Test1	Test2	Test1	Test2	Test1	Test2
SL2	1k	0.9026	0.9238	0.9663	0.9978	0.9832	1.0000
	10k	0.9798	0.9989	0.9942	1.0000	0.7495	0.7323
	100k	0.6379	0.6180	0.7121	0.6980	0.7898	0.7645
SL4	1k	0.9111	0.8515	0.9522	0.8851	0.9691	0.9278
	10k	0.7193	0.7116	0.9993	0.9999	1.0000	0.9987
	100k	0.9871	0.9940	0.9904	0.9957	0.9934	0.9970
SL8	1k	0.9971	0.9941	0.9951	0.9902	1.0000	0.9902
	10k	0.9950	0.9989	0.9420	0.9992	0.9190	0.9988
	100k	0.9949	0.9997	0.8695	0.9989	0.9996	0.9995

SP Results

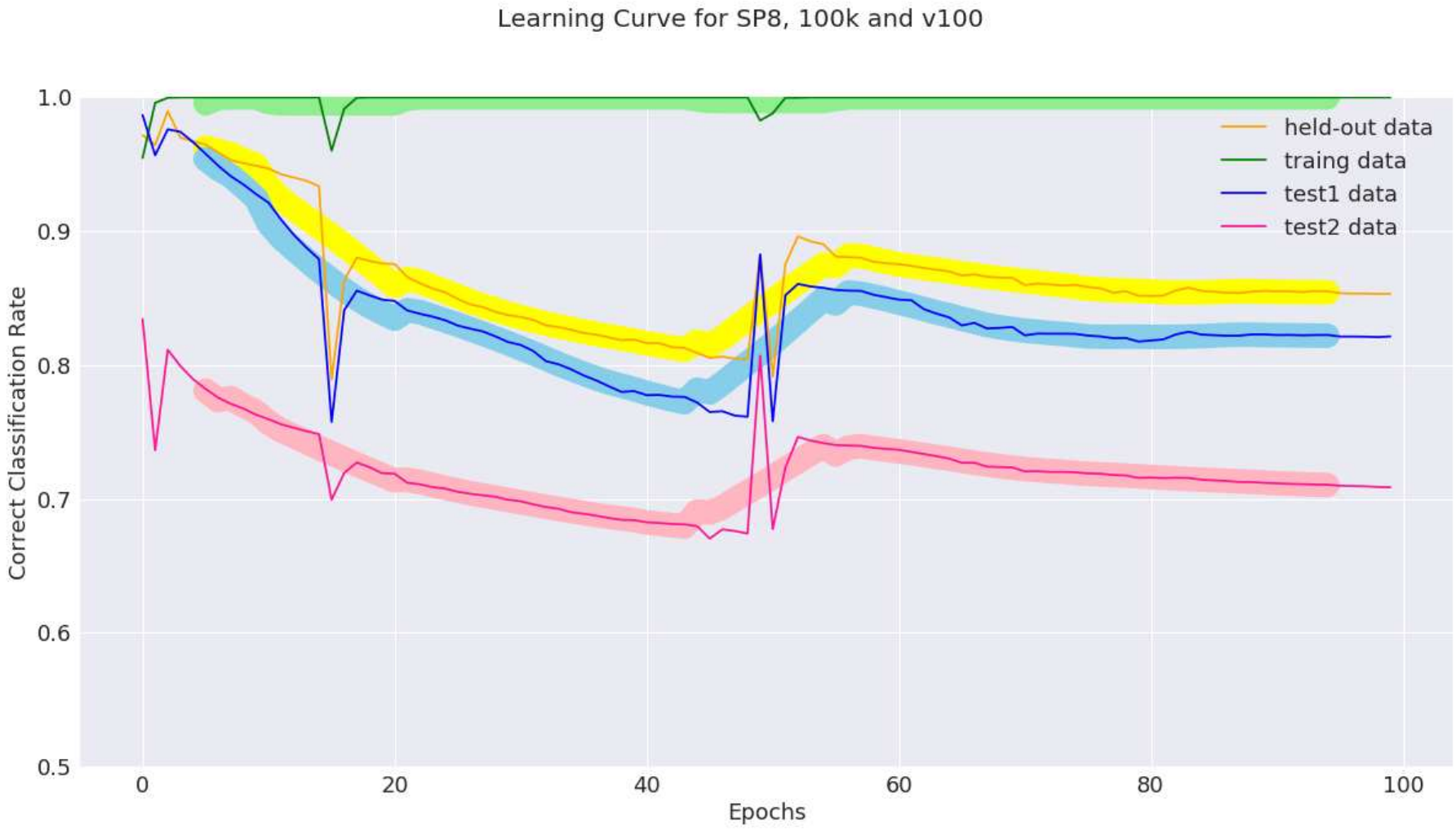
< 0.60 < 0.80 < 0.99

		LSTM type					
	Training	v10		v30		v100	
	Regimen	Test1	Test2	Test1	Test2	Test1	Test2
SP2	1k	0.9685	0.9873	0.9869	1.0000	0.9971	1.0000
	10k	1.0000	1.0000	0.7722	0.7615	1.0000	1.0000
	100k	1.0000	1.0000	0.9995	1.0000	1.0000	1.0000
SP4	1k	0.9351	0.9603	0.9697	0.9635	0.8633	0.8624
	10k	0.9991	0.9953	0.7223	0.8014	0.9872	0.9838
	100k	0.9502	0.9600	0.7988	0.7930	1.0000	1.0000
SP8	1k	0.8846	0.5850	0.8902	0.6148	0.9153	0.6472
	10k	0.7547	0.5151	0.9661	0.6099	0.8056	0.6471
	100k	0.8485	0.6569	0.8599	0.6899	0.8214	0.7087

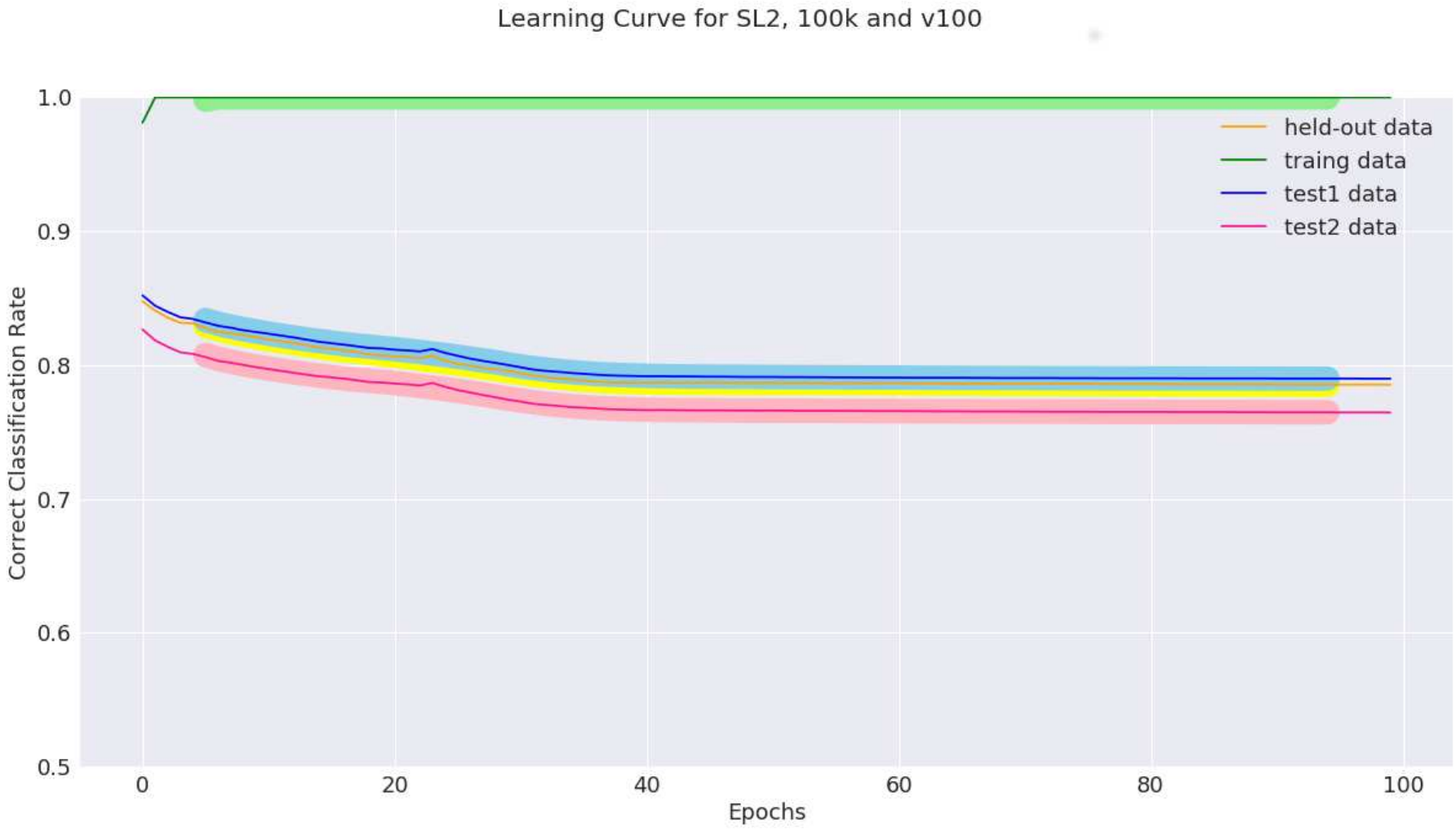
Summary

1. Across-the-board success in SL4, SL8, SP2, SP4 experiments.
2. SL2 is not as successful, especially with 100k training examples.
3. SP8 is not as successful with Test 2 always worse than Test 1 across-the-board.

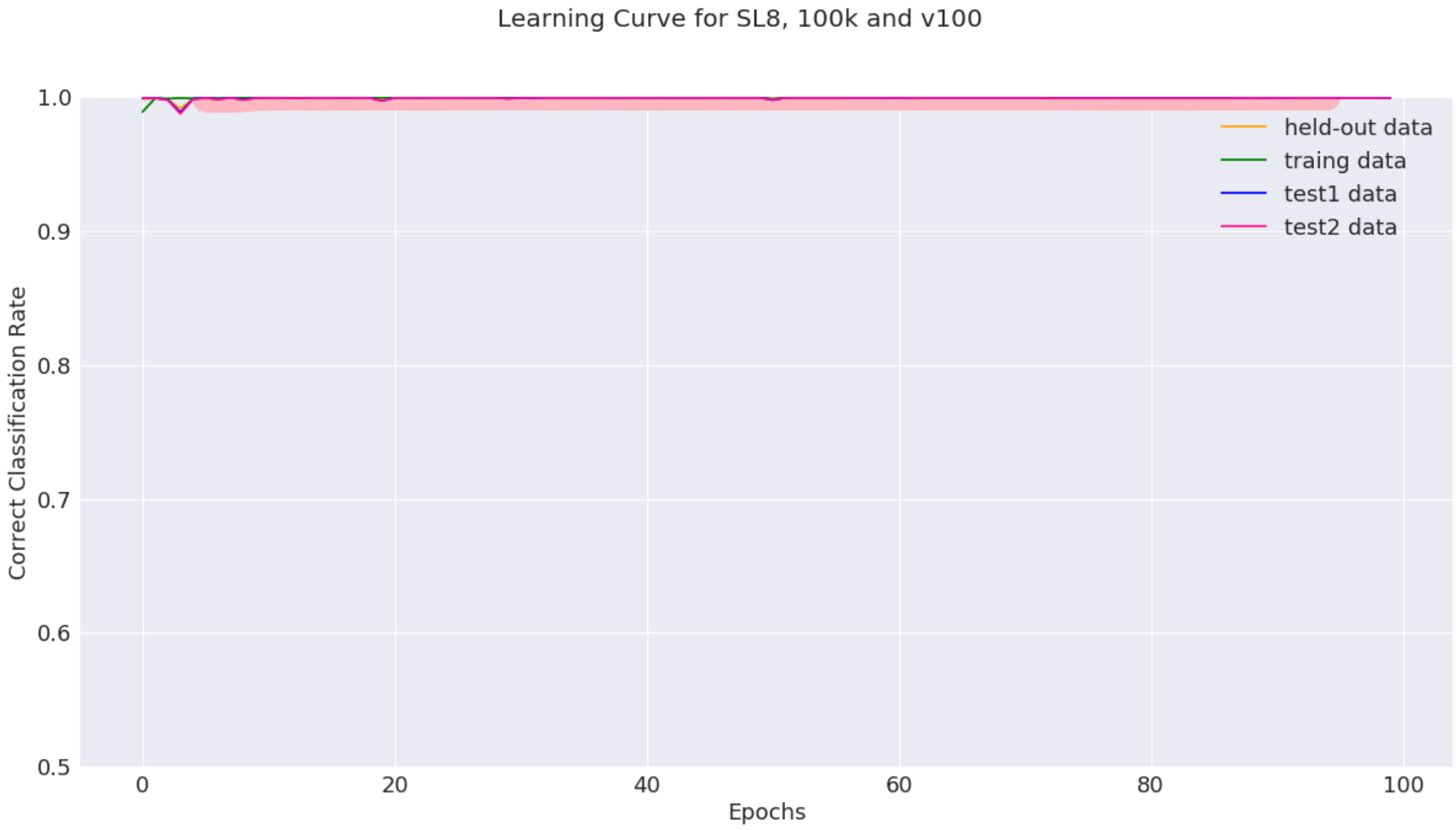
SP8 100k v100 : accuracy per epoch



SL2 100k v100 : accuracy per epoch



SL8 100k v100 : accuracy per epoch



What can explain these worse outcomes?

1. The training data is not rich enough.
2. The architecture is too naive and the pattern is beyond the capacity of the specific architecture employed.

< 0.60 < 0.80 < 0.99

RPNI lets us answer Hypothesis 1: SL Results

		LSTM				
	Training	v100		Sufficient Data?	RPNI	
	Regimen	Test1	Test2		Test1	Test2
SL2	1k	0.9832	1.0000	NO	0.8550	0.8440
	10k	0.7495	0.7323	YES	1.0000	1.0000
	100k	0.7898	0.7645	YES	1.0000	1.0000
SL4	1k	0.9691	0.9278	NO	0.9180	0.8130
	10k	1.0000	0.9987	NO	0.9946	0.9785
	100k	0.9934	0.9970	YES	1.0000	1.0000
SL8	1k	1.0000	0.9902	NO	0.9910	0.9660
	10k	0.9190	0.9988	NO	0.9980	0.9937
	100k	0.9996	0.9995	NO	0.9998	0.9996

Implementation with `gi-toolbox` (https://code.google.com/archive/p/gitoolbox/)

< 0.60 < 0.80 < 0.99

RPNI lets us answer Hypothesis 1: SP Results

		LSTM				
	Training	v100		Sufficient Data?	RPNI	
	Regimen	Test1	Test2		Test1	Test2
SP2	1k	0.9971	1.0000	YES	1.0000	1.0000
	10k	1.0000	1.0000	YES	1.0000	1.0000
	100k	1.0000	1.0000	YES	1.0000	1.0000
SP4	1k	0.8633	0.8624	YES	1.0000	1.0000
	10k	0.9872	0.9838	YES	1.0000	1.0000
	100k	1.0000	1.0000	YES	1.0000	1.0000
SP8	1k	0.9153	0.6472	NO	0.8710	0.5870
	10k	0.8056	0.6471	NO	0.8729	0.63.38
	100k	0.8214	0.7087	YES	1.0000	1.0000

Discussion

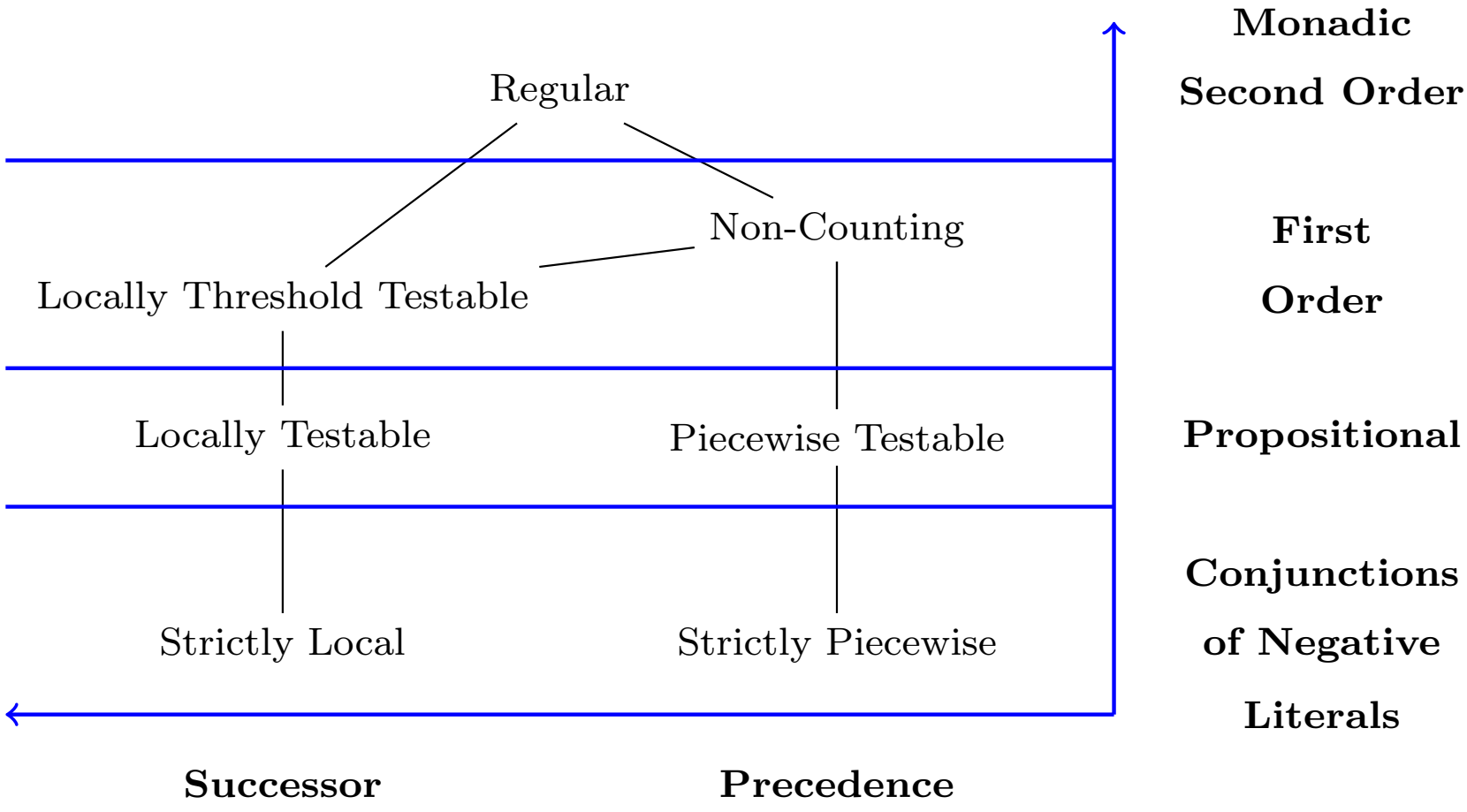
The data was sufficient in the SL2 and SP8 cases.

So the LSTM failed to generalize correctly despite it.

1. For SL2 case, overfitting
    - The dropout method somewhat improved results but overall picture is the same.
2. For SP8 case, the LSTM architecture is still challenged by long-distance dependencies.

Comparing SL8 vs SP8 cases

The difference between substring and subsequence—which reduces logically to the difference between the successor and precedence relations (Rogers et al., 2013)—is significant for naive LSTMs.



Not about One-Upmanship

- Of course we can modify the network: more nodes, more hidden layers, use Kalman filters ...
- But we can also increase the k value. We can move from Strictly Local/Piecwise to Locally Testable/Piecewise.

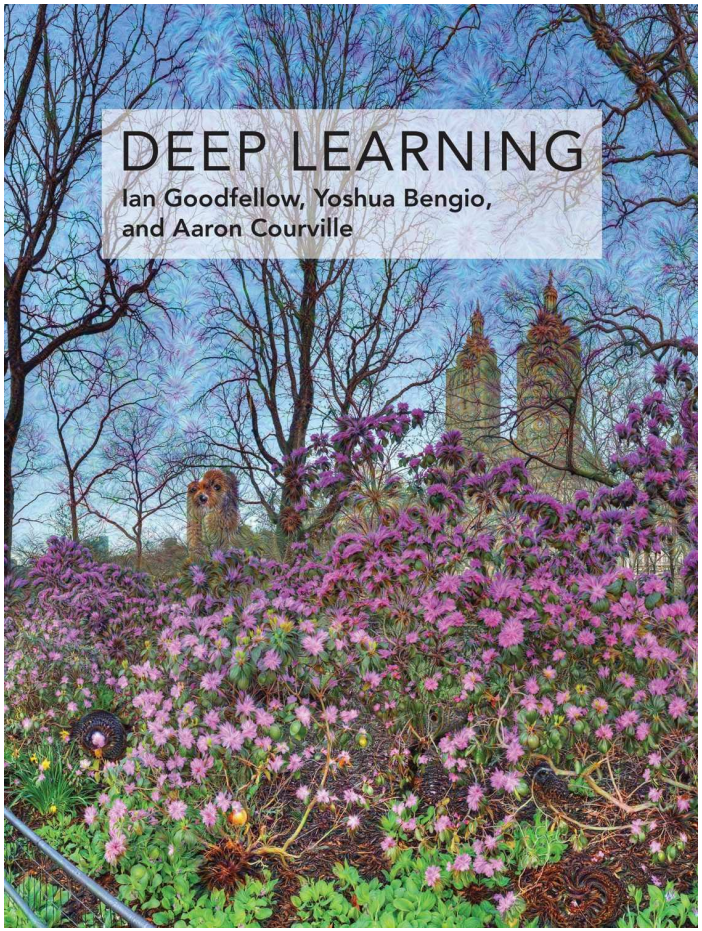
Goal

Establishing a relationship between architectures and subregular complexity.

Conclusion

1. Simple *subregular* languages discriminate naive LSTMs' ability to generalize.
2. GI algorithms can help us understand whether sufficient information is present for successful learning to occur.

Thanks for listening!



and