Formal Abstractions of Neural Sequence Models

Code!?
Formal Abstractions of Neural Sequence Models
Formal Abstractions of Neural Sequence Models

Code!?
Overview

Recurrent Neural Networks (RNNs)
- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
- More
- Analysis

Transformers
- Introduction
- A formal abstraction
Overview

Recurrent Neural Networks (RNNs)
- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
- More
- Analysis

Transformers
- Introduction
- A formal abstraction
RNNs: Introduction

Finding Structure in Time
- Elman 1990

\[ x_1, y \in \mathbb{R}^{d_h}, \quad x_2 \in \mathbb{R}^{d_i} \]
RNNs: Introduction

Finding Structure in Time
- Elman 1990

∀t : \( h_t \in \mathbb{R}^{d_h} \), \( x_t \in \mathbb{R}^{d_i} \)
RNNs: Introduction

\[ e : \Sigma \rightarrow \mathbb{R}^{d_i} \]

*input embedding*

Finding Structure in Time
- Elman 1990

\[ x_t = e(w_t) \quad \forall t: \quad h_t \in \mathbb{R}^{d_h} \quad x_t \in \mathbb{R}^{d_i} \]
RNNs: Introduction

\[ h_0 \]

Initial hidden state

\[ e: \Sigma \rightarrow \mathbb{R}^{d_i} \]

Input embedding

\[ x_t = e(w_t) \]

\[ \forall t : h_t \in \mathbb{R}^{d_h}, x_t \in \mathbb{R}^{d_i} \]

Finding Structure in Time
- Elman 1990
RNNs: Introduction

$w = w_0 w_1 w_2 \in \Sigma^*$

$x_t = e(w_t)$

$\forall t: h_t \in \mathbb{R}^{d_h}, x_t \in \mathbb{R}^{d_i}$

Finding Structure in Time
- Elman 1990
RNNs: Introduction

Finding Structure in Time
- Elman 1990

$$e : \Sigma \rightarrow \mathbb{R}^{d_i}$$

$$h_0$$

input embedding

initial hidden state

$$h_0 \xrightarrow{RNN cell} h_1 \xrightarrow{RNN cell} h_2 \xrightarrow{RNN cell} h_3$$

$$w = w_0 w_1 w_2 \in \Sigma^*$$

$$x_t = e(w_t)$$

$$\forall t: h_t \in \mathbb{R}^{d_h} \quad x_t \in \mathbb{R}^{d_i}$$
RNNs: Introduction

$e : \Sigma \rightarrow \mathbb{R}^{d_i}$

$w = w_0w_1w_2 \in \Sigma^*$

$x_t = e(w_t)$

$\forall t : h_t \in \mathbb{R}^{d_h}$

$x_t \in \mathbb{R}^{d_i}$

Finding Structure in Time
- Elman 1990
RNNs: Introduction

\[ w = w_0 w_1 w_2 \in \Sigma^* \]
RNNs: Introduction

Multi-layer RNNs (Deep RNNs)

\[ w = w_0 w_1 w_2 \in \Sigma^* \]
RNNs: Introduction
Overview

Recurrent Neural Networks (RNNs)
- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformers
- Introduction
- A formal abstraction
RNNs: Automata Relation

\[ w = w_0w_1w_2 \in \Sigma^* \]
RNNs: Automata Relation

\[ w = w_0w_1w_2 \in \Sigma^* \]
RNNs: Automata Relation

\[ w = w_0 w_1 w_2 \in \Sigma^* \]
RNNs: Automata Relation

$w = w_0 w_1 w_2 \in \Sigma^*$
RNNs: Automata Relation

When learning a regular language, simple RNNs (Elman RNNs) cluster their states in a manner that resembles an automaton for that language.

Finite State Automata and Simple Recurrent Networks
- Cleeremans et al, 1989 (references older version of Elman 1990)
Overview

Recurrent Neural Networks (RNNs)
- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformers
- Introduction
- A formal abstraction
All this is just a tiny subset for this talk!!
All this is just a tiny subset for this talk!!
RNNs: Extracting DFAs: Clustering

Omlin and Giles, 1996

Partition the RNN state space by dividing each dimension into $q$ equal portions. Explore the partitions, marking transitions between them according to first-visited state in each partition.

Extraction of Rules from Discrete-time Recurrent Neural Networks
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Input alphabet: \{a, b\}

quantisation level: \( q = 3 \)

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Input alphabet: \{a, b\}

Initial state: (0,0)

Quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Input alphabet: \{a, b\}

- Initial state: (0,0)
- Accepting state
- Rejecting state

quantisation level: \( q = 3 \)

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Input alphabet: \{a, b\}

- Initial state: (0,0)
- Accepting state
- Rejecting state

Quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Input alphabet: \{a,b\}

- Initial state: (0,0)
- Accepting state
- Rejecting state

quantisation level: \( q = 3 \)

Extraction of Rules from Discrete-time Recurrent Neural Networks

- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

- Input alphabet: \{a, b\}
- Initial state: (0,0)
- Accepting state
- Rejecting state
- Quantisation level: \( q = 3 \)

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

- Input alphabet: \{a, b\}
- Initial state: \((0,0)\)
- Accepting state
- Rejecting state
- Quantisation level: \(q = 3\)

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Quantisation, example (on RNN with total hidden state dimension 2)

Input alphabet: \{a,b\}

- Initial state: (0,0)
- Accepting state
- Rejecting state

quantisation level: \( q = 3 \)

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Input alphabet: \{a,b\}

- Initial state: (0,0)
- Accepting state
- Rejecting state

quantisation level: \( q = 3 \)

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Input alphabet: \{a,b\}

Initial state: (0,0)

Accepting state

Rejecting state

quantisation level: \( q = 3 \)

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

Input alphabet: \{a, b\}

- Initial state: (0,0)
- Accepting state
- Rejecting state

quantisation level: \(q = 8\)

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996
RNNs: Extracting DFAs: Clustering

Other approaches to clustering

Learning Finite State Machines With Self-clustering Recurrent Networks

Zeng et al, 1993

Extracting Rules from a (Fuzzy / Crisp) Recurrent Neural Network using a Self-Organizing Map

Blanco et al, 2000

State automata extraction from recurrent neural nets using k-means and fuzzy clustering

Cechin et al, 2003

Surveys:

Rule Extraction from Recurrent Neural Networks: A Taxonomy and Review

Jacobsson, 2005

An Empirical Evaluation of Rule Extraction from Recurrent Neural Networks

Wang et al, 2017
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)

RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)

RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)

L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

Learning Grammars
Extracting from RNNs
Extraction

DFAs

RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)

RNNs to DFAs: L-star
Mayr and Yovine (2018)

RNNs to DFAs: Quantisation

RNNs to WDFAs
Weiss et al (2019)

RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)

RNNs to DFAs: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

RNNs to WDFAs: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

Learning Grammars
Extracting from RNNs

Learning Regular Trees
Sakakibara (1992), Drewes and Högberg (2007)

Regular Trees ← Visibly Pushdown Automata
Alur and Madhusudan (2004)

RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)

RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)

RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

L-star + Spectral for WDFAs
Balle and Mohri (2015)

L-star for WDFAs
Balle and Mohri (2015)

RNNs to WDFAs: L-star + Spectral
Ayache et al (2018)

RNNs to WDFAs: 2-RNNs

RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

RNNs to WFA: 2-RNNs

Quantisation + Exploration of RNNs, for WDFAs
Zhang et al (2021)

Learning Grammars
Extracting from RNNs

Spectral Learning

L-star
Angluin (1987)

RNNs to DFAs: Spectral
Ayache et al (2018)

RNNs to DFAs: RNNs to DFAs: L-star
Mayr and Yovine (2018)

RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Extraction

DFAs

RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)

RNNs to DFAs: L-star
Mayr and Yovine (2018)

RNNs to DFAs: L-star + Quantisation

RNNs to DFAs: L-star + Iterative Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)

RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)

RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)

RNNs to WFA: L-star + Iterative Quantisation
Okudono et al (2019)

RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

L-star for DFAs
Weiss et al (2019)

L-star for WDFAs
Weiss et al (2019)

RNNs to WFs: Spectral
Ayache et al (2018)

RNNs to WFs: Spectral

2-RNNs to WFs: Spectral

Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)

Learning Grammars

Extracting from RNNs

Learning Regular Trees
Sakakibara (1992), Drewes and Högberg (2007)

Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)

Spectral Learning

RNNs to DFAs: Spectral
Ayache et al (2018)

RNNs to DFAs: Quantisation

RNNs to DFAs: Spectral
Ayache et al (2018)

RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)

RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
RNNs: Extracting DFAs: L-star
The L-star algorithm

Membership Queries

Learning Regular Sets from Queries and Counterexamples
Angluin 1987
RNNs: Extracting DFAs: L-star

The L-star algorithm

Membership Queries

Learning Regular Sets from Queries and Counterexamples
Angluin 1987
RNNs: Extracting DFAs: L-star
The L-star algorithm

Equivalence Queries
Membership Queries

Learning Regular Sets from Queries and Counterexamples
Angluin 1987
RNNs: Extracting DFAs: L-star

The L-star algorithm

Membership Queries
Equivalence Queries

Learning Regular Sets from Queries and Counterexamples
Angluin 1987
RNNs: Extracting DFAs: L-star
The L-star algorithm

Membership Queries
Equivalence Queries
RNNs: Extracting DFAs: L-star

The L-star algorithm

Membership Queries
Equivalence Queries

Learning Regular Sets from Queries and Counterexamples
Angluin 1987
RNNs: Extracting DFAs: L-star

The L-star algorithm

Membership Queries
Equivalence Queries

Learning Regular Sets from Queries and Counterexamples
Angluin 1987
RNNs: Extracting DFAs: L-star

Apply L-star to an RNN, to learn a DFA representing/approximating it

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017

Regular Inference on Artificial Neural Networks
Mayr and Yovine, 2018
RNNs: Extracting DFAs: L-star

Apply L-star to an RNN, to learn a DFA representing/approximating it

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017

Regular Inference on Artificial Neural Networks
Mayr and Yovine, 2018

Membership Queries

$bab?$
RNNs: Extracting DFAs: L-star

Apply L-star to an RNN, to learn a DFA representing/approximating it

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017

Regular Inference on Artificial Neural Networks
Mayr and Yovine, 2018

Membership Queries

\(bab?\)

Equivalence Queries

🤗 or 😞
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017
RNNs: Extracting DFAs: L-star

Equivalence Queries
RNNs: Extracting DFAs: L-star

Equivalence Queries

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017

Regular Inference on Artificial Neural Networks
Mayr and Yovine, 2018
RNNs: Extracting DFAs: L-star

Equivalence Queries

Randomly Sample for Counterexamples

(Paper provides PAC analysis of this approach for equivalence queries)

Faster
Assumes white-box RNN
Complicated

Slower
Assumes black-box NN
Simple

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017

Regular Inference on Artificial Neural Networks
Mayr and Yovine, 2018
RNNs: Extracting DFAs: L-star

Equivalence Queries

Randomly Sample for Counterexamples

Faster  Slower
RNNs: Extracting DFAs: L-star

Equivalence Queries

Learning Balanced Parentheses over $\Sigma = \{(,), a-z\}$

e.g. $(), (a)b, abc((a))$, etc
RNNs: Extracting DFAs: L-star

Equivalence Queries

Randomly Sample for Counterexamples

Learning Balanced Parentheses over $\Sigma = \{ (, ), a - z \}$

e.g. $(), (a)b, abc((a))$, etc
RNNs: Extracting DFAs: L-star

Equivalence Queries

Learning Balanced Parentheses over $\Sigma = \{(, ), a - z\}$

e.g. $(), ()a(), abc(())(a)$, etc

Random sampling counterexamples:

<table>
<thead>
<tr>
<th>Counterexample</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$))$</td>
<td>1.4</td>
</tr>
<tr>
<td>$(())$</td>
<td>1.6</td>
</tr>
<tr>
<td>$((())$</td>
<td>3.1</td>
</tr>
<tr>
<td>$((()())$</td>
<td>3.1</td>
</tr>
<tr>
<td>$((())())$</td>
<td>3.4</td>
</tr>
<tr>
<td>$(((())))$</td>
<td>4.7</td>
</tr>
<tr>
<td>$(((((())))))$</td>
<td>6.3</td>
</tr>
<tr>
<td>$(((((((())))))))$</td>
<td>9.2</td>
</tr>
<tr>
<td>$((((((((()))))))))$</td>
<td>14.0</td>
</tr>
<tr>
<td>$((wviw(ia)(cr)mrsnqqb)(ew)$</td>
<td>231.5</td>
</tr>
</tbody>
</table>

Abstraction based counterexamples:
Extraction

DFAs

RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)

RNNs to DFAs: L-star
Mayr and Yovine (2018)

L-star
Angluin (1987)

RNNs to DFAs: L-star + Spectral
Weiss et al (2019)

L-star + Spectral for DFAs
Balle and Mohri (2015)

RNNs to WDFAs
Weiss et al (2019)

RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

RNNs to WFA: L-star + Spectral
Okudono et al (2019)

RNNs to WFA: L-star
Mayr and Yovine (2018)

RNNs to DFAs: Quantisation
Zeng et al (1993),
Omlin and Giles (1995),
Blanco et al (2000),

RNNs to DFAs:
Quantisation
Zeng et al (1993),
Omlin and Giles (1995),
Blanco et al (2000),

2-RNNs to WFAs:
Spectral
Rabusseau et al (2017)

RNNs to WFAs:
Spectral
Rabusseau et al (2019)

RNNs to WFA: L-star +
Spectral + Iterative Quantisation
Okudono et al (2019)

RNNs to CFGs:
Trees, then Visibly Pushdown Automata
Barbot et al (2021)

RNNs to CFGs:
Trees, then Visibly Pushdown Automata
Barbot et al (2021)

Learning
Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)

Regular Trees
↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)

Learning Grammars
Extracting from RNNs
Applying Exact* Learning to NNs is possible, and can be effective!

*Well, it’s not quite exact: we can only approximate the equivalence queries
Applying Exact* Learning to NNs is possible, and can be effective!

*Well, it’s not quite exact: we can only approximate the equivalence queries

However, L-star slows quickly: it is polynomial in alphabet, DFA, and counterexample size

Exploring application of efficient variants of L-star (and making them!) could be interesting!
Applying Exact* Learning to NNs is possible, and can be effective!

*Well, it’s not quite exact: we can only approximate the equivalence queries

However, L-star slows quickly: it is polynomial in alphabet, DFA, and counterexample size

Exploring application of efficient variants of L-star (and making them!) could be interesting!

And now: we know RNNs can encode more than just DFAs, so let’s keep going
L-star + Spectral for WFAs

Balle and Mohri (2015)

L-star for WDFAs

Weiss et al (2019)

RNNs to DFAs: L-star

Mayr and Yovine (2018)

RNNs to WDFAs: Spectral

Ayache et al (2018)

RNNs to WFAs: 2-RNNs

Rabusseau et al (2017)

Rabusseau et al (2019)

Learning Grammars

RNNs to WFA: L-star + Spectral + Iterative Quantisation

Okudono et al (2019)

Learning Regular Trees

Sakakibara (1992)

Drewes and Högberg (2007)

Regular Trees ↔ Visibly Pushdown Automata

Alur and Madhusudan (2004)

Quantisation + Exploration of RNNs, for WFAs

Zhang et al (2021)

RNNs to DFAs: Quantisation


Angluin (1987)

RNNs to DFAs: L-star

RNNs to WFNAs: Spectral

Rabbosseau et al (2017)

RNNs to DFAs: L-star + Iterative Quantisation

Weiss et al (2017)

L-star + Spectral for WFAs

Balle and Mohri (2015)

RNNs to CFGs: L-star then Pattern Rule Sets

Yellin and Weiss (2021)

RNNs to CFGs: Trees, then Visibly Pushdown Automata

Barbot et al (2021)

Learning Regular Trees

Sakakibara (1992)

Drewes and Högberg (2007)

Regular Trees ↔ Visibly Pushdown Automata

Alur and Madhusudan (2004)
When considering a finite alphabet, second-order simple RNNs are equivalent to weighted finite automata (WFAs).

Connecting Weighted Automata and Recurrent Neural Networks through Spectral Learning
Rabusseau et al, 2019
RNNs: Extracting WFAs: Background!

- Language-Model RNNs
- WFAs
  - Matrix Representation
RNNs: Extracting WFAs: Background!

- Language-Model RNNs
- WFAs
  - Matrix Representation
RNNs: Extracting WFAs: Background!

![Diagram of RNNs and WFAs]

- **RNN Cell 1**: Input $h_0$ to $h_1$, with output $e(w_0)$.
- **RNN Cell 2**: Input $h'_{0}$ to $h'_{1}$, with output $e(w_1)$.
- **FF (Feedforward)**: Input $h'_{2}$ to produce $v_{out}$.

The diagram illustrates the flow of information through two RNN cells, each processing inputs and producing intermediate and final outputs.
RNNs: Extracting WFAs: Background!

Binary Classifier RNN

RNN cell 1

$e(w_0)$

$RNN$ cell 2

$h'_2$

$h'_{out}$

FF

$P(Acc)$

$P(Rej)$

$P(Rej)$

$P(Acc)$
RNNs: Extracting WFAs: Background!

Language-Model RNN

\[ P(\sigma_1 | w_0w_1) \]
\[ P(\sigma_2 | w_0w_1) \]
\[ P(\sigma_3 | w_0w_1) \]
\[ \ldots \]
\[ P(\sigma_{|\Sigma|} | w_0w_1) \]

Next-Token Distribution

(Including end-of-sequence)
RNNs: Extracting WFAs: Background!

Language-Model RNN

\[ \text{RNN}(w_1w_2) = P(w_1 | \varepsilon) \cdot P(w_2 | w_1) \cdot P(\text{EOS} | w_1w_2) \]
RNNs: Extracting WFAs: Background!

- Language-Model RNNs
- WFAs
  - Matrix Representation
RNNs: Extracting WFAs: Background!

**DFA**

deterministic

\[ A = \langle \Sigma, Q, q_0, F, \delta_Q \rangle \]

\[ \delta_Q : Q \times \Sigma \to Q \]

\[ A(w) = \begin{cases} 
\text{Acc} & \text{if } \hat{\delta}_Q(w) \in F \\
\text{Rej} & \text{else}
\end{cases} \]
RNNs: Extracting WFAs: Background!

DFA

\[ A = \langle \Sigma, Q, q_0, F, \delta_Q \rangle \]

\[ \delta_Q : Q \times \Sigma \rightarrow Q \]

WDFA

\[ A = \langle \Sigma, Q, q_0, \delta_Q, \delta_W, \beta \rangle \]

\[ \delta_W : Q \times \Sigma \rightarrow \mathbb{R} \]

\[ \beta : Q \rightarrow \mathbb{R} \]

\[ A(w) = \begin{cases} 
\text{Acc} & \text{if } \hat{\delta}_Q(w) \in F \\
\text{Rej} & \text{else}
\end{cases} \]

\[ A(w) = \left( \prod_{i \in [\mid w \mid]} \delta_W(\hat{\delta}_Q(w_{1:i-1}), w_i) \right) \cdot \beta(\hat{\delta}_Q(w)) \]
RNNs: Extracting WFAs: Background!

**DFA**
- deterministic

\[ A = \langle \Sigma, Q, q_0, F, \delta_Q \rangle \]

\[ \delta_Q : Q \times \Sigma \to Q \]

\[ A(w) = \begin{cases} 
  \text{Acc} & \text{if } \hat{\delta}_Q(w) \in F \\
  \text{Rej} & \text{else}
\end{cases} \]

**WDFA**
- weighted deterministic

\[ A = \langle \Sigma, Q, q_0, \delta_Q, \delta_W, \beta \rangle \]

\[ \delta_W : Q \times \Sigma \to \mathbb{R} \]

\[ \beta : Q \to \mathbb{R} \]

\[ A(w) = \left( \prod_{i \in [|w|]} \delta_W(\hat{\delta}_Q(w_{1:i-1}), w_i) \right) \cdot \beta(\hat{\delta}_Q(w)) \]

**WFA**
- weighted

\[ A = \langle \Sigma, Q, \alpha, \beta, \{W_\sigma\}_{\sigma \in \Sigma} \rangle \]

\[ \alpha : Q \to \mathbb{R} \]

\[ \beta : Q \to \mathbb{R} \]

\[ W_\sigma \in \mathbb{R}^{Q \times Q} \]

\[ A(w) = \alpha \cdot W_{w_1} \cdot W_{w_2} \cdots \cdot W_{w_{|w|}} \cdot \beta \]
Extraction

WFAs

Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)

RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)

RNNs to DFAs: L-star
Mayr and Yovine (2018)

L-star for WDFAs
Weiss et al (2019)

RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

RNNs to WFA: L-star + Spectral
Ayache et al (2018)

2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)

Learning Grammars

Extracting from RNNs

RNNs to WFs:
Spectral

Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)

Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs
RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

A spectral algorithm for learning hidden Markov models
Hsu et al, 2008

Grammatical inference as a principal component analysis problem
Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective
Balle et al, 2013
RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

\[ T = \langle \Sigma, Q, \alpha^G, \beta^G, \{ W^G_\sigma \}_{\sigma \in \Sigma} \rangle \]

(Example on \( \Sigma = \{ a, b \} \))

A spectral algorithm for learning hidden Markov models
Hsu et al, 2008

Grammatical inference as a principal component analysis problem
Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective
Balle et al, 2013
RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

\[ T = \langle \Sigma, Q, \alpha^G, \beta^G, \{ W^G_\sigma \}_{\sigma \in \Sigma} \rangle \]

(example on \( \Sigma = \{a, b\} \))

1. Make Hankel Sub-blocks

Hankel sub-block \( H \)

<table>
<thead>
<tr>
<th>( \varepsilon )</th>
<th>( b )</th>
<th>( ab )</th>
<th>( \ldots )</th>
<th>( v )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T(\varepsilon) )</td>
<td>( T(b) )</td>
<td>( T(ab) )</td>
<td>( T(v) )</td>
<td></td>
</tr>
<tr>
<td>( T(a) )</td>
<td>( T(ab) )</td>
<td>( T(aab) )</td>
<td>( T(a \cdot v) )</td>
<td></td>
</tr>
<tr>
<td>( T(ab) )</td>
<td>( T(abab) )</td>
<td>( T(abab) )</td>
<td>( T(ab \cdot v) )</td>
<td></td>
</tr>
</tbody>
</table>

A spectral algorithm for learning hidden Markov models
Hsu et al, 2008

Grammatical inference as a principal component analysis problem
Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective
Balle et al, 2013
RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

\[ T = (\Sigma, Q, \alpha^G, \beta^G, \{W^G_\sigma\}_{\sigma \in \Sigma}) \]

(Example on \( \Sigma = \{a, b\} \))

1. Make Hankel Sub-blocks

Hankel sub-block \( H \)

| \( u \) | \( \epsilon \) | \( b \) | \( ab \) | \(...\) | \( \cdot \) | \( v \) |
|---|---|---|---|---|---|
| \( T(u) \) | \( T(\epsilon) \) | \( T(b) \) | \( T(ab) \) | \(...\) | \( T(v) \) |
| \( T(\cdot \cdot \cdot v) \) | \( T(ab) \) | \( T(\cdot \cdot \cdot v) \) | \( T(\cdot \cdot \cdot v) \) | \( T(\cdot \cdot \cdot v) \) | \( T(\cdot \cdot \cdot v) \) |
| \( T(ab) \) | \( T(\cdot \cdot \cdot v) \) | \( T(ab) \) | \( T(\cdot \cdot \cdot v) \) | \( T(ab) \) | \( T(\cdot \cdot \cdot v) \) |
| \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) |

Hankel sub-block \( H^b \)

| \( u \) | \( \epsilon \) | \( b \) | \( ab \) | \(...\) | \( \cdot \) | \( v \) |
|---|---|---|---|---|---|
| \( T(u) \) | \( T(\epsilon) \) | \( T(b) \) | \( T(ab) \) | \(...\) | \( T(v) \) |
| \( T(\cdot \cdot \cdot v) \) | \( T(ab) \) | \( T(\cdot \cdot \cdot v) \) | \( T(\cdot \cdot \cdot v) \) | \( T(ab) \) | \( T(\cdot \cdot \cdot v) \) |
| \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) |
| \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) |

Hankel sub-block \( H^a \)

| \( u \) | \( \epsilon \) | \( b \) | \( ab \) | \(...\) | \( \cdot \) | \( v \) |
|---|---|---|---|---|---|
| \( T(u) \) | \( T(\epsilon) \) | \( T(b) \) | \( T(ab) \) | \(...\) | \( T(v) \) |
| \( T(\cdot \cdot \cdot v) \) | \( T(ab) \) | \( T(\cdot \cdot \cdot v) \) | \( T(\cdot \cdot \cdot v) \) | \( T(ab) \) | \( T(\cdot \cdot \cdot v) \) |
| \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) |
| \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) | \( T(ab) \) |

---

A spectral algorithm for learning hidden Markov models
Hsu et al, 2008

Grammatical inference as a principal component analysis problem
Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective
Balle et al, 2013
## RNNs: Extracting WFAs: Background!

### Spectral Learning of WFAs

\[ T = \langle \Sigma, Q^G, \beta^G, \{ W^G_\sigma \}_{\sigma \in \Sigma} \rangle \]

1. Make Hankel Sub-blocks

**Hankel sub-block \( H \)**

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>ε</th>
<th>b</th>
<th>ab</th>
<th>...</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>T(ε)</td>
<td>T(b)</td>
<td>T(ab)</td>
<td>...</td>
<td>T(v)</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>T(a)</td>
<td>T(ab)</td>
<td>T(aab)</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ab</td>
<td>T(ab)</td>
<td>T(ab)</td>
<td>T(abab)</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>T(ε)</td>
<td>T(b)</td>
<td>T(ab)</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>T(u)</td>
<td>T(u \cdot b)</td>
<td>T(u \cdot ab)</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Hankel sub-block \( H^b \)**

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>ε</th>
<th>b</th>
<th>ab</th>
<th>...</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>T(ε)</td>
<td>T(b)</td>
<td>T(ab)</td>
<td>...</td>
<td>T(v)</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>T(a)</td>
<td>T(ab)</td>
<td>T(aab)</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ab</td>
<td>T(ab)</td>
<td>T(ab)</td>
<td>T(abab)</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>T(ε)</td>
<td>T(b)</td>
<td>T(ab)</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>T(u)</td>
<td>T(u \cdot b)</td>
<td>T(u \cdot ab)</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. \( U, d, V = \text{SVD}(H) \)

3. (Optional): Trim \( U, d, V \) to \( k \) largest singular values

4. \( \alpha = H_{\epsilon,:}V, \beta = (HV)^\dagger H_{:,\epsilon} \), \( W_\sigma = (HV)^\dagger H^\sigma V \)

5. \( A = \langle \Sigma, [k], \alpha, \beta, \{ W_\sigma \}_{\sigma \in \Sigma} \rangle \)

---

A spectral algorithm for learning hidden Markov models
Hsu et al, 2008

Grammatical inference as a principal component analysis problem
Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective
Balle et al, 2013
RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

\[ T = \langle \Sigma, Q^G, \beta^G, \{ W^G_\sigma \}_{\sigma \in \Sigma} \rangle \]

1. Make Hankel Sub-blocks

Hankel sub-block \( H \)

Hankel sub-block \( H^b \)

Hankel sub-block \( H^a \)

2. \( U, d, V = \text{SVD}(H) \)

3. (Optional): Trim \( U, d, V \) to \( k \) largest singular values

4. \[ \alpha = H_{\epsilon,:}^* V, \quad \beta = (HV)^\dagger H_{,:\epsilon}^* \]
   \[ W_\sigma = (HV)^\dagger H\sigma V \]

5. \[ A = \langle \Sigma, [k], \alpha, \beta, \{ W_\sigma \}_{\sigma \in \Sigma} \rangle \]
Extraction

WFAs

- Quantisation + Exploration of RNNs, for WFAs
  - Zhang et al (2021)

RNNs to DFAs:
- L-star + Iterative Quantisation
- L-star then Pattern Rule Sets
  - Yellin and Weiss (2021)

RNNs to CFGs:
- Trees, then Visibly Pushdown Automata
  - Alur and Madhusudan (2004)
- L-star + Spectral + Iterative Quantisation
  - Okudono et al (2019)

L-star for WDFAs

2-RNNs to WFAs:
- Spectral

Learning Grammars

- Extracting from RNNs
- Learning Regular Trees
  - Sakakibara (1992)
  - Drewes and Högberg (2007)

Regular Trees ↔ Visibly Pushdown Automata

- Alur and Madhusudan (2004)

Spectral Learning
- Ayache et al (2018)
- Okudono et al (2019)

RNNs to WFA: L-star + Spectral + Iterative Quantisation
RNNs: Extracting WFAs: Spectral Methods

Explaining Black Boxes on Sequential Data Using Weighted Automata

Ayache et al, 2018

Black Box Model

Build Hankel basis (P,S) by sampling sequences according to black box’s distribution

Try multiple sizes for final WFA (truncations $k$ of SVD decomposition) and choose best result

Spectral Learning

Hsu et al (2008), Bailly et al (2009), Balle et al. (2013)
RNNs: Extracting WFAs: Spectral Methods

Explaining Black Boxes on Sequential Data Using Weighted Automata
Ayache et al, 2018

Black Box Model
Build Hankel basis \((P,S)\) by sampling sequences according to black box’s distribution
Try multiple sizes for final WFA (truncations \(k\) of SVD decomposition) and choose best result

Weighted Automata Extraction from Recurrent Neural Networks via Regression on State Spaces
Okudono et al, 2019

White Box Model (specifically RNN)
Build Hankel basis \((P,S)\) according to queries from and counterexamples to active learning algorithm
Continue until reach equivalence

Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al. (2013)

L-star
Angluin 1987

Quantisation-Exploration
Omlin and Giles, 1996
Background: $L^*$
### Background: L*

#### The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>ε 1 1 0</td>
<td>a 1 1 1</td>
</tr>
<tr>
<td>a</td>
<td>a 1 1 1</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b 1 0 0</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>ba 0 0 0</td>
<td></td>
</tr>
<tr>
<td>bb</td>
<td>bb 1 0 0</td>
<td></td>
</tr>
</tbody>
</table>

#### Membership Queries
- ε
- a

#### Equivalence Queries
- ε?
- a?

#### Counter-Examples
- ba

---

**Diagram**

- Start state (ε)
- Transition arrows:
  - a: from 1 to 2
  - b: from 2 to 3
  - b: from 3 to 4
  - a: from 4 to 1

- States:
  - 1
  - 2
  - 3
  - 4
Background: $L^*$

The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$bb$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The diagram shows a transition from state 1 to state 0 with an epsilon transition, question mark $\varepsilon$?
### Background: L*

#### The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>ɛ</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ɛ</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bb</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Closedness**
- **Consistency**

![Transition diagram](image)
### Background: $L^*$

#### The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$bb$</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Closedness

For all $p \in P$ and $\sigma \in \Sigma$, if we were to add $p \cdot \sigma$ to $P$, its row would be identical to that of some $p'$ already in $P$. 

---
Background: \( L^* \)

The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>( \varepsilon )</th>
<th>( a )</th>
<th>( ba )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \varepsilon )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( a )</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( b )</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( ba )</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( bb )</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Closedness**

For all \( p \in P \) and \( \sigma \in \Sigma \), if we were to add \( p \cdot \sigma \) to \( P \), its row would be identical to that of some \( p' \) already in \( P \).
### Background: L*

#### Closedness

For all $p \in P$ and $\sigma \in \Sigma$, if we were to add $p \cdot \sigma$ to $P$, its row would be identical to that of some $p'$ already in $P$.

<table>
<thead>
<tr>
<th></th>
<th>$\epsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$bb$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Background: $L^*$

**The Observation Table**

<table>
<thead>
<tr>
<th>$P$</th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$bb$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Consistency**

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to $P$, their rows would be identical to each other.
### Background: $L^*$

The Observation Table

<table>
<thead>
<tr>
<th>$P$</th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Consistency**

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to $P$, their rows would be identical to each other.
# Background: L*

## The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>$\epsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$bb$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Equivalence Query

- $\epsilon$?
- $a$?
- $b$?
The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>ε</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Equivalence Query

Each group of identical rows describes a single state
# Background: L*

## The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>ε</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bb</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Equivalence Query

- ε?
- a?
- b?
- ?

Diagrams:

1. Start
2. End
3. Transition
4. Transition

Graph:

- Start
- End
- Transition
- Transition
Background: $L^*$

### The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$bb$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Equivalence Query

```
\varepsilon? \rightarrow a? \rightarrow b? \rightarrow ?
```

```
bab \rightarrow \times \times
```
The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>(\varepsilon)</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>(\varepsilon)</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bb</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(this is simplified: it also adds to S)
The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>ε</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>ε</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P</td>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>bb</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Equivalence Query

Equivalence: L*
Background: L*

The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>ε</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bb</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Closedness

Equivalence Query
Background: $L^*$

The Observation Table

<table>
<thead>
<tr>
<th>$P$</th>
<th>$\epsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Consistency?

Equivalence Query

$\epsilon$?

$a$?

$b$?

$bab$
Background: L*

The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bb</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to $P$, their rows would be identical to each other.
Background: $L^*$

### The Observation Table

<table>
<thead>
<tr>
<th>S</th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Consistency**

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to $P$, their rows would be identical to each other.

<table>
<thead>
<tr>
<th>Equivalence Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>start $\xrightarrow{a,b}$ 1</td>
</tr>
<tr>
<td>2 $\xrightarrow{b}$ b</td>
</tr>
<tr>
<td>3 $\xrightarrow{a}$ b</td>
</tr>
<tr>
<td>4 $\xrightarrow{a}$ a</td>
</tr>
<tr>
<td>$ba \xleftarrow{a,b}$ 0</td>
</tr>
</tbody>
</table>

- $\varepsilon$? Yes
- $a$? Yes
- $b$? No
- $\varepsilon$? No
The Observation Table

<table>
<thead>
<tr>
<th>S</th>
<th>ε</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>ε</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ba</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>bb</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to $P$, their rows would be identical to each other.
Background: $L^*$

The Observation Table

<table>
<thead>
<tr>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to $P$, their rows would be identical to each other.
Background: L*

The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ba</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to $P$, their rows would be identical to each other.
## Background: $L^*$

### The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>$\epsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to $P$, their rows would be identical to each other.

### Equivalence Query

- `\epsilon` to `a` and `b` to `ba` are valid transitions.
- `b` to `2` and `a` to `3` are valid transitions.
- `a` to `4` is a valid transition.

- `ba` to `X` is a valid transition.
Background: $L^*$

The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$bb$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Extraction
WFAs

Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)

RNNs to DFAs: Quantisation
L-star + Iterative Quantisation
Weiss et al (2017)

RNNs to DFAs: L-star
Mayr and Yovine (2018)

RNNs to PDFAs
Weiss et al (2019)

RNNs to WFA: L-star
Zeng et al (1993),
Omlin and Giles (1995),
Blanco et al (2000),

Learning Grammars
Extracting from RNNs

Spectral Learning
Hsu et al (2008),
Bailly et al (2009),
Balle et al (2013)

2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)

Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)

Regular Trees
↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)

RNNs to DFAs: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

RNNs to WFA: L-star + Spectral
Ayache et al (2018)

RNNs to WFNs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)

Learning Grammars
Extracting from RNNs

RNNs to CFGs:
L-star then Pattern Rule Sets
Yellin and Weiss (2021)

RNNs to CFGs:
Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Adapting L*

The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$ba$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$a$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$b$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$ba$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

RNN, trained on

there may be some noise...

What shall we put in the table?
Adapting L*

The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>ε</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>a</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>b</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>ba</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>bb</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

RNN, trained on

there may be some noise...

What shall we put in the table?

**Direct approach:** Full sequence weight
- **Flaw:** Will quickly degrade
**Intuition:** Conditional probabilities
- **Flaw:** Also degrade as S grows
- **Fix:** Last token probabilities
Adapting L* 

The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>ε</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td></td>
<td>ε?</td>
<td>a?</td>
<td>ba?</td>
</tr>
<tr>
<td>ε</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>a</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>b</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>ba</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>bb</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

RNN, trained on

there may be some noise...

What shall we put in the table?

Final Choice: Last Token Probabilities
Adapting L*

The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>?</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>a</td>
<td>?</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>b</td>
<td>?</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>ba</td>
<td>?</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

RNN, trained on

there may be some noise...

What shall we put in the table?

Final Choice: Last Token Probabilities

Realisation:
empty suffix doesn’t mean anything anymore...
Adapting L* 

The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>0.1</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>a</td>
<td>0.05</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>b</td>
<td>0.1</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>ba</td>
<td>0.1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>bb</td>
<td>0.05</td>
<td>0.7</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Final Choice: Last Token Probabilities

Realisation:
empty suffix doesn’t mean anything anymore…
but end-of-sequence does
Adapting L*

The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>$</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>0.1</td>
<td>0.5</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0.05</td>
<td>0.7</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0.1</td>
<td>0.5</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>ba</td>
<td>0.1</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>bb</td>
<td>0.05</td>
<td>0.7</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

RNN, trained on

there may be some noise...

What shall we put in the table?

Final Choice: Last Token Probabilities

Okay, we have our adaptation. Let’s go!?
Adapting L*

The Observation Table

<table>
<thead>
<tr>
<th>P</th>
<th>S</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>0.1</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>a</td>
<td>0.05</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>b</td>
<td>0.1</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>ba</td>
<td>0.1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

RNN, trained on

start

there may be some noise...

What shall we put in the table?

Final Choice: Last Token Probabilities
Adapting L*

The Observation Table

<table>
<thead>
<tr>
<th>S</th>
<th>$</th>
<th>a</th>
<th>ba</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ε</td>
<td>0.1</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>a</td>
<td>0.05</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>b</td>
<td>0.1</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>ba</td>
<td>0.1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>bb</td>
<td></td>
<td>0.7</td>
<td>0.5</td>
</tr>
</tbody>
</table>

RNN, trained on

There may be some noise...

What shall we put in the table?

Final Choice: Last Token Probabilities

Nice Realisation: Can use additive tolerance
Adapting \( L^* \)

The Observation Table

<table>
<thead>
<tr>
<th>( P )</th>
<th>( S )</th>
<th>( $$ )</th>
<th>( a )</th>
<th>( ba )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \epsilon )</td>
<td>0.1</td>
<td>0.5</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>( a )</td>
<td>0.05</td>
<td>0.7</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>( b )</td>
<td>0.1</td>
<td>0.5</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>( ba )</td>
<td>0.1</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

RNN, trained on

there may be some noise...

What shall we put in the table?

Final Choice: Last Token Probabilities

Nice Realisation: Can use additive tolerance

Challenge: Non-transitivity of tolerance
Adapting L*

The Observation Table

<table>
<thead>
<tr>
<th></th>
<th>$</th>
<th>$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>$</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>0.1</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>a</td>
<td>0.05</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>b</td>
<td>0.1</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>ba</td>
<td>0.1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>bb</td>
<td>0.05</td>
<td>0.7</td>
<td>0.5</td>
</tr>
</tbody>
</table>

RNN, trained on

there may be some noise...

What shall we put in the table?

Final Choice: Last Token Probabilities

Nice Realisation: Can use additive tolerance

Challenge: Non-transitivity of tolerance
Adapting L*

Dealing with the Additive Tolerance

In particular: dealing with ‘chains’ of similar prefixes

- a
- b
- aa
- aba

- close enough
- close enough
- close enough
- nope
Adapting L*

Dealing with the Additive Tolerance

In particular: dealing with ‘chains’ of similar prefixes

How can we fix the “closedness” and “consistency” definitions, to avoid mistaken groupings?
Adapting L*

Dealing with the Additive Tolerance

   In particular: dealing with ‘chains’ of similar prefixes

**Immediate realisation:** Attempting to fix definitions for table is painful
Adapting L*

Dealing with the Additive Tolerance

In particular: dealing with ‘chains’ of similar prefixes

Immediate realisation: Attempting to fix definitions for table is painful

Solution:
Fill table optimistically, and fix problems post-hoc
Adapting L*

Optimistic Table and Post-Hoc Fixes
Adapting L*

Optimistic Table and Post-Hoc Fixes

1. Check closedness as normal, just with the additive tolerance

2. Check consistency as normal, just with the additive tolerance

3. Make hypothesis with caution!
Adapting L*

Optimistic Table and Post-Hoc Fixes

3. Make hypothesis with caution!

Potential Problems:

1. Clustering of prefixes causes states with **non-deterministic transitions**
   
   post hoc fix: refine

2. Clustering of prefixes creates states with prefixes **beyond threshold** of each other
   
   post hoc fix: refine
Anytime Stopping

This algorithm is unlikely to complete on real-world tasks. Thus, we allow anytime stopping:

- Prioritise high-weight prefixes
- Avoid very low-weight separating suffixes
- On stop, map remaining prefixes to best match
  - This is actually quite slow, and might not be very beneficial (*needs to be tested!*).
Extraction

CFGs

RNNs to DFAs: L-star + Spectral
Balle and Mohri (2015)

Angluin (1987)

RNNs to DFAs: L-star
Mayr and Yovine (2018)

RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)

RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)

Learning Grammars
Extracting from RNNs

(Coming up: (some) RNNs can do more than just finite automata)
Observation: L-star learning a CFG seems to have structured increases (example on balanced parentheses)

etc...
Yellin and Weiss (2021)

Observation: L-star learning a CFG seems to have structured increases (example on balanced parentheses)

1. In the limit, the union of all DFAs in this sequence accepts the non-regular language BP
Observation: L-star learning a CFG seems to have structured increases (example on balanced parentheses)

1. In the limit, the union of all DFAs in this sequence accepts the non-regular language BP

2. The difference between each pair of successive RNNs is structured (for some CFGs at least)
Observation: L-star learning a CFG seems to have structured increases (example on balanced parentheses)

1. In the limit, the union of all DFAs in this sequence accepts the non-regular language BP

2. The difference between each pair of successive RNNs is structured (for some CFGs at least)
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

**Patterns**
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

**Patterns**

- Structure
RNNs: Extraction: CFGs: **Pattern Rule Sets**
Yellin and Weiss (2021)

**Patterns**
- Structure
- Entry
RNNs: Extraction: CFGs: **Pattern Rule Sets**
Yellin and Weiss (2021)

**Patterns**

- Structure
  - Entry
  - Exit
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

**Patterns**

- Structure
  - Entry
  - Exit
- Connection Point(s)
RNNs: Extraction: CFGs: **Pattern Rule Sets**
Yellin and Weiss (2021)

**Patterns**

- Structure
  - Entry
  - Exit
- Connection Point(s)
- Composable
  - Connection points are on compositions
RNNs: Extraction: CFGs: Pattern Rule Sets
Yellin and Weiss (2021)

**Patterns**

- Structure
  - Entry
  - Exit
- Connection Point(s)
- Composable
  - Connection points are on compositions
  - Composition can be *serial* or *circular*
Patterns

- Structure
  - Entry
  - Exit
- Connection Point(s)
- Composable
  - Connection points are on compositions
  - Composition can be *serial* or *circular*
Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$.
2. Insert circular pattern $p_1$ on join state of $p_2$.
3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$. 

RNNs: Extraction: CFGs: Pattern Rule Sets

Yellin and Weiss (2021)
RNNs: Extraction: CFGs: **Pattern Rule Sets**
Yellin and Weiss (2021)

**Rules**

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. **The first DFA**: an initial pattern $p_1$.
2. Insert circular pattern $p_1$ on join state of $p_2$.
3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$.
Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. **The first DFA**: an initial pattern $p_1$.
2. Insert circular pattern $p_1$ on join state of $p_2$.
3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$

(When we get to extraction: this might not be the same first DFA that L-star suggests)
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

**Rules**

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$. 2. **Insert circular pattern** $p_1$ on join state of $p_2$. 3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$.

---

**Legend:**
- initial state
- exit state
- join state
- transitions added to successor DFA
**RNNs: Extraction: CFGs: Pattern Rule Sets**

Yellin and Weiss (2021)

**Rules**

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$.
2. **Insert circular pattern** $p_1$ on join state of $p_2$.
3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$.

What happens when adding another (different) circular pattern to the same state?

---

**Legend:**
- ✨ initial state
- 🔴 exit state
- 🟧 join state
- ⬇️ transitions added to successor DFA
Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$. 2. **Insert circular pattern** $p_1$ on join state of $p_2$. 3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$.

What happens when adding another (different) circular pattern to the same state?
**RNNs: Extraction: CFGs: Pattern Rule Sets**

Yellin and Weiss (2021)

**Rules**

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern \(p_1\).
2. Insert circular pattern \(p_1\) on join state of \(p_2\).
3. **Insert serial pattern** \(p_1\) on join state of serial pattern \(p_2\).

---

**Legend:**

- ○ initial state
- ● exit state
- ● join state
- ⬛ transitions added to successor DFA
Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$.
2. Insert circular pattern $p_1$ on join state of $p_2$.
3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$.

\[ p_1 \cup p_2 \Rightarrow (p_1 \cup p_2) \bowtie p_3 \]

Legend:
- initial state
- exit state
- join state
- transitions added to successor DFA
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

**Rules**

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$.
2. Insert circular pattern $p_1$ on join state of $p_2$.
3. **Insert serial pattern** $p_1$ on join state of serial pattern $p_2$.

![Diagram](image)

**Legend:**
- initial state
- exit state
- join state
- transitions added to successor DFA
Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$. 2. Insert circular pattern $p_1$ on join state of $p_2$. 3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$.

What happens when adding another (different) serial pattern to the same state?
Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$.
2. Insert circular pattern $p_1$ on join state of $p_2$.
3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$.

What happens when adding another (different) serial pattern to the same state?
**RNNs: Extraction: CFGs: Pattern Rule Sets**

*Yellin and Weiss (2021)*

**Rules**

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$.  
2. Insert circular pattern $p_1$ on join state of $p_2$.  
3. **Insert serial pattern** $p_1$ on join state of serial pattern $p_2$.

What happens when adding another (different) serial pattern to the same state? 

---

**Legend:**
- **initial state**
- **exit state**
- **join state**
- **transitions added to successor DFA**
- **transitions in original DFA that are not part of $p_1 \circ p_2$**
**Rules**

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern $p_1$.
2. Insert circular pattern $p_1$ on join state of $p_2$.
3. Insert serial pattern $p_1$ on join state of serial pattern $p_2$.

Legend:
- Initial state
- Exit state
- Join state
- Transitions added to successor DFA
- Transitions in original DFA that are not part of $p_1 \circ p_2$
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

**Recovering a Pattern Rule Set**
RNNs: Extraction: CFGs: **Pattern Rule Sets**
Yellin and Weiss (2021)

**Recovering a Pattern Rule Set**

1. Identify (some of) the patterns: the new states between consecutive RNNs
   1. (Note that reject state is treated as not there)
RNNs: Extraction: CFGs: Pattern Rule Sets
Yellin and Weiss (2021)

Recovering a Pattern Rule Set

1. Identify (some of) the patterns: the new states between consecutive RNNs
   1. (Note that reject state is treated as not there)
1. Identify (some of) the patterns: the new states between consecutive RNNs
   1. (Note that reject state is treated as not there)
2. Identify composite patterns: patterns onto which other patterns have been grafted in some of the expansions
1. **Identify (some of) the patterns**: the new states between consecutive RNNs
   1. (Note that reject state is treated as not there)
2. **Identify composite patterns**: patterns onto which other patterns have been grafted in some of the expansions
   1. Split composite patterns into base patterns according to observed join state.
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Recovering a Pattern Rule Set

1. Identify (some of) the patterns: the new states between consecutive RNNs
   1. (Note that reject state is treated as not there)

2. Identify composite patterns: patterns onto which other patterns have been grafted in some of the expansions
   1. Split composite patterns into base patterns according to observed join state.
   2. Record which pattern was grafted on - i.e., which rule was used.
1. Identify (some of) the patterns: the new states between consecutive RNNs
   1. (Note that reject state is treated as not there)
2. Identify composite patterns: patterns onto which other patterns have been grafted in some of the expansions
   1. Split composite patterns into base patterns according to observed join state.
   2. Record which pattern was grafted on - i.e., which rule was used.
3. To handle noise: have threshold, and only keep patterns and rules that have frequency above that threshold
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

**Recovering a Pattern Rule Set**

1. Identify (some of) the patterns: the new states between consecutive RNNs
   1. (Note that reject state is treated as not there)

2. Identify composite patterns: patterns onto which other patterns have been grafted in some of the expansions
   1. Split composite patterns into base patterns according to observed join state.
   2. Record which pattern was grafted on - i.e., which rule was used.

3. To handle noise: have threshold, and only keep patterns and rules that have frequency above that threshold

4. Convert PRS to CFG!
Extraction

CFGs

RNNs to CFGs:
L-star then Pattern Rule Sets
Yellin and Weiss (2021)

RNNs to DFAs:
L-star for WDFAs
Weiss et al (2019)

L-star for
WDFAs
Weiss et al (2019)

L-star
Angluin (1987)

RNNs to DFAs:
Quantisation

RNNs to DFAs:
L-star
Mayr and Yovine (2018)

RNNs to DFAs:
L-star + Iterative Quantisation
Weiss et al (2017)

RNNs to WFs:
Spectral
Ayache et al (2018)

RNNs to WFs:
Spectral + Iterative Quantisation
Okudono et al (2019)

RNNs to WFs:
Quantisation + Exploration of RNNs, for WFs
Zhang et al (2021)

RNNs to WFs:
L-star + Spectral + Iterative Quantisation
Okudono et al (2019)

Learning Grammars
Extracting from RNNs

Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)

Learning Regular Trees
Sakakibara (1992), Drewes and Höglberg (2007)

RNNs to WFs:
Spectral
Rabusseau et al (2017)

RNNs to WFs:
Quantisation

RNNs to WFs:
L-star
Mayr and Yovine (2018)

RNNs to DFAs:
L-star
Balle and Mohri (2015)

RNNs to DFAs:
Quantisation

RNNs to DFAs:
L-star
Balle and Mohri (2015)

RNNs to WFs:
Quantisation
All this is just a tiny subset for this talk!!
Overview

Recurrent Neural Networks (RNNs)
- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformers
- Introduction
- A formal abstraction
RNNs: Expressive Power

- Simple RNNs
  Elman 1990 (/1988)

- LSTMs
  Hochreiter and Schmidhuber 1997

- GRUs
  Cho et al 2014, Chung et al 2014

- Saturated RNNs
  Merril 2019

- Rational Recurrences
  Peng et al 2018

- Hierarchy of RNNs
  Merril et al 2020

- RNNs can do Bounded Hierarchical Languages
  Hewitt et al, 2020

- LSTM counting mechanism
  Weiss et al 2018

- Evaluating LSTM-CFG connection
  Skachkova et al 2018

- Bernardy 2018

- Sennhauser and Berwick 2018

- Yu et al 2019

- RNNs are like DFAs
  Cleeremans et al 1989
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

RNNs are like DFAs
Cleeremans et al 1989

RNNs are Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

Saturated RNNs
Merril 2019

Rational Recurrences
Peng et al 2018

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

LSTM counting mechanism
Weiss et al 2018

Evaluating LSTM-CFG connection
Skachkova et al 2018

Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

RNNs are like DFAs
Cleeremans et al 1989
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Chung et al 2014

RNNs are like DFAs
Cleeremans et al 1989

Rational Recurrences
Peng et al 2018

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

LSTM counting mechanism
Weiss et al 2018

Evaluating LSTM-CFG connection
Skachkova et al 2018

Bernardy 2018

Sennhauser and Berwick 2018

Yu et al 2019

RNNs are like DFAs
Cleeremans et al 1989
RNNs: Expressive Power

- Simple RNNs
  - Elman 1990
  - Hochreiter and Schmidhuber 1997

- LSTMs
  - Hochreiter and Schmidhuber 1997
  - Gers and Schmidhuber 2001

- GRUs
  - Cho et al 2014
  - Chung et al 2014

- Saturated RNNs
  - Merril 2019

- Rational Recurrences
  - Peng et al 2018

- Hierarchy of RNNs
  - Merril et al 2020

- RNNs can do Bounded Hierarchical Languages
  - Hewitt et al, 2020

- LSTM counting mechanism
  - Weiss et al 2018

- Evaluating LSTM-CFG connection
  - Skachkova et al 2018

- Bernardy 2018
- Sennhauser and Berwick 2018
- Yu et al 2019

- RNNs are like DFAs
  - Cleeremans et al 1989

- RNNs are like DFAs
  - Cleeremans et al 1989
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs are like DFAs
Cleeremans et al 1989

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

LSTM counting mechanism
Weiss et al 2018

Saturated RNNs
Merril 2019

Rational Recurrences
Peng et al 2018

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

LSTM counting mechanism
Weiss et al 2018

Saturated RNNs
Merril 2019

Hierarchical of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

RNNs are like DFAs
Cleeremans et al 1989

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

Rational Recurrences
Peng et al 2018
RNNs: Expressive Power: Theory

RNNs are Turing Complete:
Given infinite precision, RNNs can emulate pushing and popping to/from stacks in their hidden state. Thus, given also infinite time, they can simulate any Turing Machine.

On the computational power of Neural Nets
Siegelmann and Sonntag (1995)
RNNs: Expressive Power: Theory

RNNs are Turing Complete:
Given infinite precision, RNNs can emulate pushing and popping to/from stacks in their hidden state. Thus, given also infinite time, they can simulate any Turing Machine

On the computational power of Neural Nets
Siegelmann and Sonntag (1995)
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

LSTM counting mechanism
Weiss et al 2018

Rational Recurrences
Peng et al 2018

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

Saturated RNNs
Merril 2019

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

RNNs are like DFAs
Cleeremans et al 1989
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

Saturated RNNs
Merril 2019

Hierarchical of RNNs
Merril et al 2020

Rational Recurrences
Peng et al 2018

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

LSTM counting mechanism
Weiss et al 2018

RNNs are like DFAs
Cleeremans et al 1989
RNNs: Expressive Power: Practice
LSTMs can count

LSTM recurrent networks learn simple context-free and context-sensitive languages
Gers and Schmidhuber, 2001
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

RNNs are like DFAs
Cleeremans et al 1989

RNNs Turing Complete
Siegelman and Sonntag 1995

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

Saturated RNNs
Merril 2019

Hierarchy of RNNs
Merril et al 2020

Rational Recurrences
Peng et al 2018

LSTM counting mechanism
Weiss et al 2018

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

RNNs are like DFAs
Cleeremans et al 1989

LSTM counting mechanism
Weiss et al 2018

Saturated RNNs
Merril 2019

Hierarchy of RNNNs
Merril et al 2020

Rational Recurrences
Peng et al 2018

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020
LSTMs: Counting Mechanism

Simple RNN

\[ h_{t+1} = \tanh(W^h h_t + W^x x_t + b) \]

Elman (1990)
LSTMs: Counting Mechanism

Simple RNN

$$h_{t+1} = \tanh(W^h h_t + W^x x_t + b)$$

Elman (1990)
LSTMs: Counting Mechanism

**Simple RNN**

\[ h_{t+1} = \tanh(W^h h_t + W^x x_t + b) \]

Elman (1990)

**GRU**

- \[ z_t = \sigma(W^z x_t + U^z h_{t-1} + b^z) \]
- \[ r_t = \sigma(W^r x_t + U^r h_{t-1} + b^r) \]
- \[ \tilde{h}_t = \tanh(W^h x_t + U^h (r_t \cdot h_{t-1}) + b^h) \]
- \[ h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t \]


**LSTM**

- \[ f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \]
- \[ i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \]
- \[ o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \]
- \[ \tilde{c}_t = \tanh(W^c x_t + U^c h_{t-1} + b^c) \]
- \[ c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \]
- \[ h_t = o_t \cdot g(c_t) \]

Hochreiter and Schmidhuber (1997)
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

**GRU**

\[ z_t = \sigma(W^z x_t + U^z h_{t-1} + b^z) \]
\[ r_t = \sigma(W^r x_t + U^r h_{t-1} + b^r) \]
\[ \tilde{h}_t = \tanh(W^h x_t + U^h (r_t \odot h_{t-1}) + b^h) \]
\[ h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \]

**LSTM**

\[ f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \]
\[ i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \]
\[ o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \]
\[ \tilde{c}_t = \tanh(W^c x_t + U^c h_{t-1} + b^c) \]
\[ c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \]
\[ h_t = o_t \odot g(c_t) \]
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

**GRU**

- $z_t \in (0,1)$
- $r_t \in (0,1)$

\[
\tilde{h}_t = \tanh(W^h x_t + U^h (r_t \cdot h_{t-1}) + b^h)
\]

\[
h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t
\]

**LSTM**

- $f_t \in (0,1)$
- $i_t \in (0,1)$
- $o_t \in (0,1)$

\[
\tilde{c}_t = \tanh(W^c x_t + U^c h_{t-1} + b^c)
\]

\[
c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t
\]

\[
h_t = o_t \cdot g(c_t)
\]
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

\[ h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \]

LSTM

\[ c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \]

\[ h_t = o_t \odot g(c_t) \]
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

**GRU**

- \( z_t \in (0,1) \)
- \( r_t \in (0,1) \)
- \( \tilde{h}_t \in (-1,1) \)
- \( h_t = z_t \cdot h_{t-1} + (1 - z) \cdot \tilde{h}_t \)

**LSTM**

- \( f_t \in (0,1) \)
- \( i_t \in (0,1) \)
- \( o_t \in (0,1) \)
- \( \tilde{c}_t \in (-1,1) \)
- \( c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \)
- \( h_t = o_t \cdot g(c_t) \)
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

**GRU**

\[ z_t \in (0,1) \]
\[ r_t \in (0,1) \]
\[ \tilde{h}_t \in (-1,1) \]
\[ h_t = z_t \cdot h_{t-1} + (1 - z) \cdot \tilde{h}_t \]

**LSTM**

\[ f_t \in (0,1) \]
\[ i_t \in (0,1) \]
\[ o_t \in (0,1) \]
\[ \tilde{c}_t \in (-1,1) \]
\[ c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \]
\[ h_t = o_t \cdot g(c_t) \]

Interpolation
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

\[ h_t = z_t \odot h_{t-1} + (1 - z) \odot \tilde{h}_t \]

\[ \tilde{h}_t \in (-1, 1) \]

\[ z_t \in (0, 1), \quad r_t \in (0, 1) \]

\[ Bounded! \]

Interpolation

LSTM

\[ f_t \in (0, 1), \quad i_t \in (0, 1), \quad o_t \in (0, 1), \quad c_t \in (-1, 1) \]

\[ \tilde{c}_t \in (-1, 1) \]

\[ c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \]

\[ h_t = o_t \odot g(c_t) \]
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

\[ z_t \in (0,1) \]
\[ r_t \in (0,1) \]
\[ \tilde{h}_t \in (-1,1) \]
\[ h_t = z_t \cdot h_{t-1} + (1 - z) \cdot \tilde{h}_t \]

Bounded!

Interpolation

LSTM

\[ f_t \in (0,1) \]
\[ i_t \in (0,1) \]
\[ o_t \in (0,1) \]
\[ \tilde{c}_t \in (-1,1) \]
\[ c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \]
\[ h_t = o_t \cdot g(c_t) \]
**LSTMs: Counting Mechanism**

LSTMs can count (and GRUs cannot)

**GRU**

\[ z_t \in (0,1) \]
\[ r_t \in (0,1) \]
\[ \tilde{h}_t \in (-1,1) \]
\[ h_t = z_t \cdot h_{t-1} + (1 - z) \cdot \tilde{h}_t \]

**Interpolation**

**LSTM**

\[ f_t \in (0,1) \]
\[ i_t \in (0,1) \]
\[ o_t \in (0,1) \]
\[ \tilde{c}_t \in (-1,1) \]
\[ c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \]
\[ h_t = o_t \cdot g(c_t) \]

**Addition**

**Bounded!**
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

**GRU**

\[
\begin{align*}
  z_t &\in (0,1) \\
  r_t &\in (0,1) \\
  \tilde{h}_t &\in (-1,1) \\
  h_t &= z_t \cdot h_{t-1} + (1 - z) \cdot \tilde{h}_t
\end{align*}
\]

**Bounded!**

**LSTM**

\[
\begin{align*}
  f_t &\approx 1 \\
  i_t &\approx 1 \\
  o_t &\in (0,1) \\
  \tilde{c}_t &\in (-1,1) \\
  c_t &\approx c_{t-1} + \tilde{c}_t \\
  h_t &= o_t \cdot g(c_t)
\end{align*}
\]

**Interpolation**

**Addition**

\[
\begin{align*}
  c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t
\end{align*}
\]
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

**GRU**

\[ h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t \]

\[ z_t \in (0,1) \]
\[ r_t \in (0,1) \]
\[ \tilde{h}_t \in (-1,1) \]

**Interpolation**

**Bounded!**

**LSTM**

\[ f_t \approx 1 \]
\[ i_t \approx 1 \]
\[ o_t \in (0,1) \]
\[ \tilde{c}_t \approx 1 \]
\[ c_t \approx c_{t-1} + 1 \]
\[ h_t = o_t \cdot g(c_t) \]

Increase by 1

\[ c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \]
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

**GRU**

\[
\begin{align*}
z_t & \in (0, 1) \\
r_t & \in (0, 1) \\
\tilde{h}_t & \in (-1, 1) \\
h_t = z_t \circ h_{t-1} + (1 - z) \circ \tilde{h}_t
\end{align*}
\]

**Interpolation**

**LSTM**

\[
\begin{align*}
f_t & \approx 1 \\
i_t & \approx 1 \\
o_t & \in (0, 1) \\
\tilde{c}_t & \approx -1 \\
c_t & \approx c_{t-1} - 1 \\
h_t = o_t \circ g(c_t)
\end{align*}
\]

**Decrease by 1**

\[
c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t
\]

Bounded!
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

**GRU**

\[
\begin{align*}
z_t &\in (0,1) \\
r_t &\in (0,1) \\
\tilde{h}_t &\in (-1,1) \\
h_t &= z_t \cdot h_{t-1} + (1 - z) \cdot \tilde{h}_t
\end{align*}
\]

**Interpolation**

**LSTM**

\[
\begin{align*}
f_t &\approx 1 \\
\tilde{c}_t &\in (-1,1) \\
\tilde{c}_t &\approx c_{t-1} \\
o_t &\in (0,1) \\
h_t &= o_t \circ g(c_t)
\end{align*}
\]

**Do Nothing**

\[
c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t
\]
**LSTMs: Counting Mechanism**

LSTMs can count (and GRUs cannot)

**GRU**

- \( z_t \in (0,1) \)
- \( r_t \in (0,1) \)
- \( \tilde{h}_t \in (-1,1) \)
- \( h_t = z_t \cdot h_{t-1} + (1 - z) \cdot \tilde{h}_t \)

**LSTM**

- \( f_t \approx 0 \)
- \( i_t \approx 0 \)
- \( o_t \in (0,1) \)
- \( \tilde{c}_t \in (-1,1) \)
- \( c_t \approx 0 \)
- \( h_t = o_t \cdot g(c_t) \)

**Interpolation**

**Reset**

\[
c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t
\]

**Bounded!**
LSTMs: Counting Mechanism
LSTMs can count (and GRUs cannot)

**GRU**

\[
\begin{align*}
z_t &\in (0,1) \\
r_t &\in (0,1) \\
\tilde{h}_t &\in (-1,1) \\
h_t &= z_t \cdot h_{t-1} + (1 - z) \cdot \tilde{h}_t
\end{align*}
\]

**Bounded!**

**Interpolation**

**LSTM**

\[
\begin{align*}
f_t &\approx 0 \\
i_t &\approx 0 \\
o_t &\in (0,1) \\
\tilde{c}_t &\in (-1,1) \\
c_t &\approx 0 \\
h_t &= o_t \circ g(c_t)
\end{align*}
\]

**Can Count!**

**Reset**

\[
c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t
\]
LSTMs: Counting Mechanism
LSTMs can count (and GRUs cannot)

Trained $a^n b^n$, (on positive examples up to length 100)

Activations on $a^{1000} b^{1000}$:
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

Rational Recurrences
Peng et al 2018

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

Saturated RNNs
Merril 2019

LSTM counting mechanism
Weiss et al 2018

RNNs are like DFAs
Cleeremans et al 1989
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

Saturated RNNs
Merril 2019

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

LSTM counting mechanism
Weiss et al 2018

Rational Recurrences
Peng et al 2018

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

RNNs are like DFAs
Cleeremans et al 1989
RNNs: Expressive Power

**Simple RNNs**
Elman 1990 (1988)

**LSTMs**
Hochreiter and Schmidhuber 1997

**GRUs**
Cho et al 2014, Chung et al 2014

**LSTMs can count/learn simple CFGs**
Gers and Schmidhuber 2001

**Saturated RNNs**
Merril 2019

**RNNs Turing Complete**
Siegelman and Sonntag 1995

**LSTM counting mechanism**
Weiss et al 2018

**Rational Recurrences**
Peng et al 2018

**Hierarchy of RNNs**
Merril et al 2020

**RNNs can do Bounded Hierarchical Languages**
Hewitt et al, 2020

**Evaluating LSTM-CFG connection**
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

**RNNs are like DFAs**
Cleeremans et al 1989
Saturated RNNs

\[ f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \]
\[ i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \]
\[ o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \]

\[ f_t \in (0,1) \]
\[ i_t \in (0,1) \]
\[ o_t \in (0,1) \]
Saturated RNNs

\[ f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \]
\[ i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \]
\[ o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \]

\[ f_t \in (0, 1) \]
\[ i_t \in (0, 1) \]
\[ o_t \in (0, 1) \]

\[ \sigma : \mathbb{R} \rightarrow (0, 1) \]
\[ \tanh : \mathbb{R} \rightarrow (-1, 1) \]
Saturated RNNs

\[ f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \]
\[ i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \]
\[ o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \]

\( f_t \approx 1 \)
\( i_t \approx 0 \)
\( o_t \in (0,1) \)

\( \sigma : \mathbb{R} \rightarrow (0,1) \)
\( \tanh : \mathbb{R} \rightarrow (-1,1) \)
Saturated RNNs

\[ f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \]

\[ i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \]

\[ o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \]

\[ f_t \approx 1 \]

\[ i_t \approx 0 \]

\[ o_t \in (0,1) \]

\[ \sigma : \mathbb{R} \rightarrow (0,1) \]

\[ \tanh : \mathbb{R} \rightarrow (-1,1) \]
Saturated RNNs

Sequential Neural Networks as Automata - Merrill (2019)

\[ f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \]
\[ i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \]
\[ o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \]

\[ \sigma : \mathbb{R} \rightarrow (0,1) \]
\[ \text{tanh} : \mathbb{R} \rightarrow (-1,1) \]
Saturated RNNs

Sequential Neural Networks as Automata - Merrill (2019)

\[ f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \]
\[ i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \]
\[ o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \]

\[ f_t \approx 1 \]
\[ i_t \approx 0 \]
\[ o_t \in (0,1) \]

Σ : \( \mathbb{R} \rightarrow (0,1) \)
\[ \tanh : \mathbb{R} \rightarrow (-1,1) \]

RNN is a parameterised function, \( R(w : \theta) \)

As \( \theta \) “increases”, inputs to activations increase, saturating them

Saturated RNN: \( \text{sat-}R(w : \theta) = \lim_{N \rightarrow \infty} R(w : N\theta) \)
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

RNNs are like DFAs
Cleeremans et al 1989

LSTM counting mechanism
Weiss et al 2018

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

Saturated RNNs
Merril 2019

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

Rational Recurrences
Peng et al 2018
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

LSTM counting mechanism
Weiss et al 2018

Saturated RNNs
Merril 2019

Rational Recurrences
Peng et al 2018

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

RNNs are like DFAs
Cleeremans et al 1989
RNNs: Expressive Power

- **Simple RNNs**
  - Elman 1990 (/1988)

- **LSTMs**
  - Hochreiter and Schmidhuber 1997

- **GRUs**
  - Cho et al 2014, Chung et al 2014

- **RNNs Turing Complete**
  - Siegelman and Sonntag 1995

- **LSTMs can count/learn simple CFGs**
  - Gers and Schmidhuber 2001

- **Saturated RNNs**
  - Merril 2019

- **LSTM counting mechanism**
  - Weiss et al 2018

- **Rational Recurrences**
  - Peng et al 2018

- **Evaluating LSTM-CFG connection**
  - Skachkova et al 2018
  - Bernardy 2018
  - Sennhauser and Berwick 2018
  - Yu et al 2019

- **Hierarchy of RNNs**
  - Merril et al 2020

- **RNNs are like DFAs**
  - Cleeremans et al 1989

- **RNNs can do Bounded Hierarchical Languages**
  - Hewitt et al, 2020
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs are like DFAs
Cleeremans et al 1989

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

LSTM counting mechanism
Weiss et al 2018

Saturated RNNs
Merril 2019

Hierarchy of RNNNs
Merril et al 2020

Rational Recurrences
Peng et al 2018

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020
RNNs: Expressive Power

- Simple RNNs
  Elman 1990 (/1988)

- LSTMs
  Hochreiter and Schmidhuber 1997

- GRUs
  Cho et al 2014, Chung et al 2014

- RNNs Turing Complete
  Siegelman and Sonntag 1995

- LSTMs can count/learn simple CFGs
  Gers and Schmidhuber 2001

- Saturated RNNs
  Merril 2019

- Hierarchy of RNNs
  Merril et al 2020

- LSTM counting mechanism
  Weiss et al 2018

- Rational Recurrences
  Peng et al 2018

- Evaluating LSTM-CFG connection
  Skachkova et al 2018
  Bernardy 2018
  Sennhauser and Berwick 2018
  Yu et al 2019

- RNNs are like DFAs
  Cleeremans et al 1989

- RNNs can do Bounded Hierarchical Languages
  Hewitt et al, 2020
Overview

Recurrent Neural Networks (RNNs)
- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformers
- Introduction
- A formal abstraction
Overview

Recurrent Neural Networks (RNNs)
- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformers
- Introduction
- A formal abstraction
Overview

Recurrent Neural Networks (RNNs)
- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformer Encoders
- Introduction
- A formal abstraction

Didn’t make it! :(  
But my website has links to talks on “Thinking Like Transformers”, the work I wanted to introduce here: 
https://sgailw.cs.wp.cs.technion.ac.il/publications/

The 1 hour talk includes an introduction on transformers, while the 5 minute talk assumes familiarity.