Discrete Representations

Weiss et al 2020, Dalvi et al 2018

Evan Hernandez and Geeticka Chauhan

dez@mit.edu, geeticka@mit.edu

Neuro-Symbolic Models for NLP (6.884), Oct 8 2020

Outline

1. Paper 1: Weiss et al	25 min	**11:35-12:00p**
2. Breakout room	10 min	**12:00-12:10p**
3. Discussion	5 min	**12:10-12:15p**
4. Break	15 min	**12:15p-12:30p**

--------------------------------------- 1 hour mark ---------------------------------------

5. Paper 2: Dalvi et al	40 min	**12:30-1:10p**
6. Breakout room	10 min	**1:10-1:20p**
7. Discussion	5 min	**1:20-1:25p**

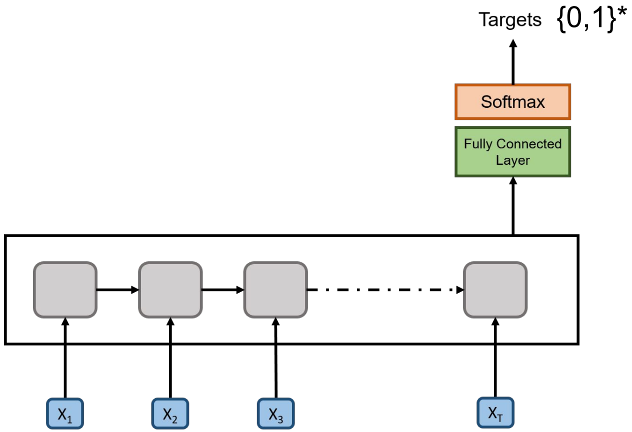
Extracting Automata from Recurrent Neural Networks

Gail Weiss, Yoav Goldberg, Eran Yahav

Goal: Model Distillation

Can we approximate the operations of an RNN using a deterministic finite automaton?

Given: **Oracle RNN (R)**

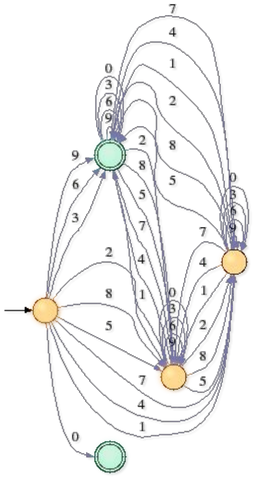


?

≈

As measured by the classification output

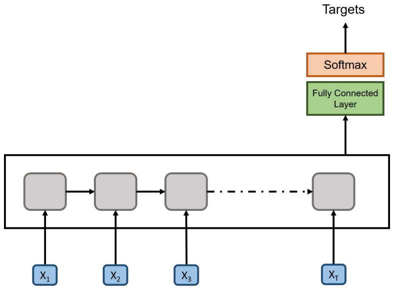
Find: **Minimal DFA (L)**



https://www.arxiv-vanity.com/papers/1801.08322/
https://www.brics.dk/automaton/

Core Contributions

Given: **Oracle RNN (R)**

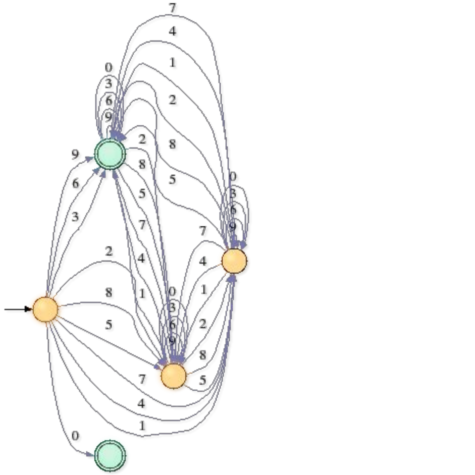


Must answer:

1. **Membership queries** : Label the data point
2. **Equivalence queries** : Is the hypothesis equivalent to me? i.e. accept or reject DFA with counter eg. if reject

Use as functions to call when suggesting new hypotheses

Find: **Minimal DFA (L)**



Approximate using the L* algorithm (black box)

Core Contributions

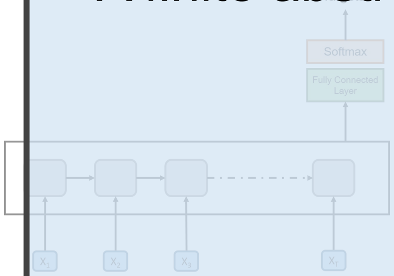
Given: **Oracle RNN**

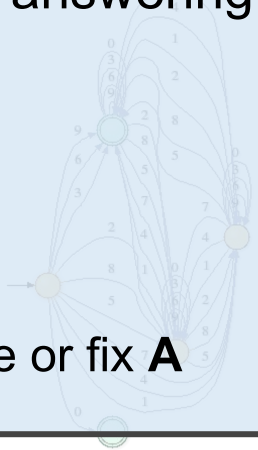
Find: **Minimal DFA**

A finite abstraction to the RNN to allow for answering of equivalence queries:

Finite Abstraction (**A**)
L* DFA (**L**)
RNN (**R**)

L == A if **L = R** else find counterexample or fix **A**





Must answer:

1. **Membership queries** : Label the data point
2. **Equivalence queries** : Is the hypothesis equivalent to me? i.e. accept or reject DFA with counter eg. if reject

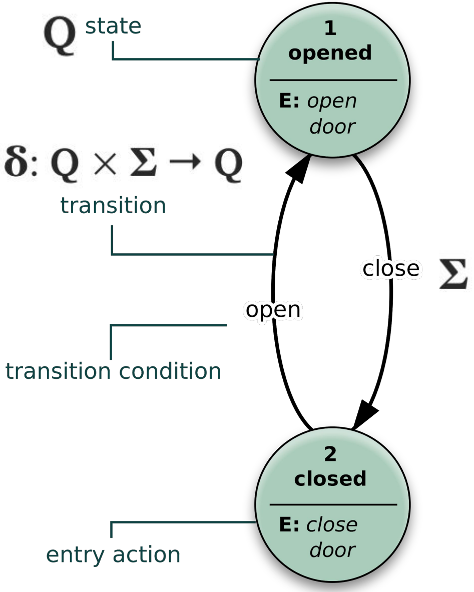
Use as functions to
new hypotheses

Approximate using the L* algorithm (black box)

Brief Recap of Automata Theory

Deterministic Finite State Automata (DFA)

5 tuple $\langle \mathbf{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta}, \mathbf{qo}, \mathbf{F} \rangle$ such that:



1. $\mathbf{Q}$ all states, i.e. {1,2}
2. $\boldsymbol{\Sigma}$ alphabet i.e. {open, close}
3. $\boldsymbol{\delta}: \mathbf{Q} \times \boldsymbol{\Sigma} \rightarrow \mathbf{Q}$ transition function e.g. $\boldsymbol{\delta}$(1, close) = 2
4. $\mathbf{qo} \in \mathbf{Q}$ starting state, assume 1

   "DFA can have only 1 start state"
5. $\mathbf{F} \subseteq \mathbf{Q}$ final/ accept state(s)

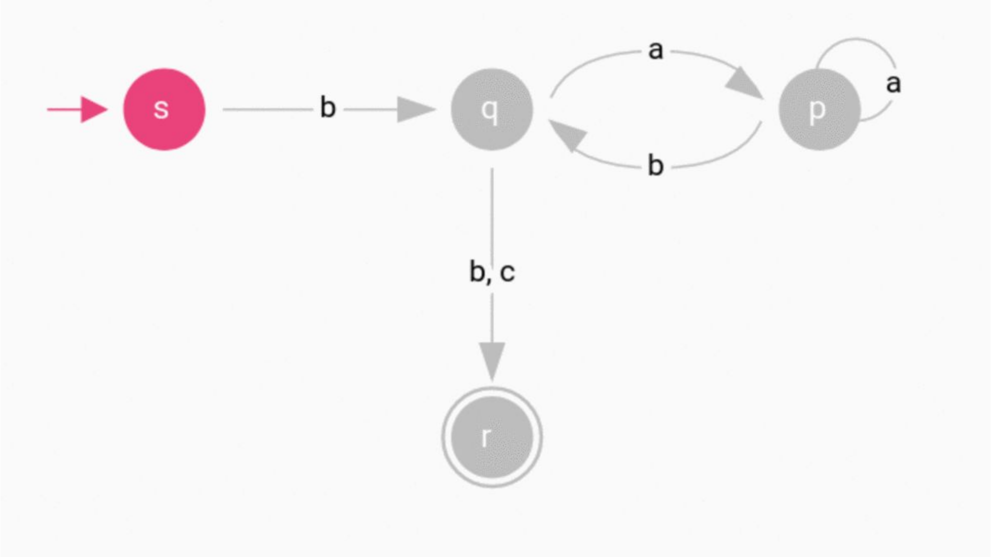
Regular Language: The set of languages that can be accepted by a DFA

https://commons.wikimedia.org/wiki/File:Finite_state_machine_example_with_comments.svg

DFA Running Example

Regular Expressions are commonly represented with DFAs eg. **baabb**

$q_0 = s$ $F = \{r\}$ $Q = \{ s, q, p, r \}$ $\Sigma = \{ b, a, c \}$



In Weiss et al, RNN hidden states are compared to Q

https://levelup.gitconnected.com/an-example-based-introduction-to-finite-state-machines-f908858e450f

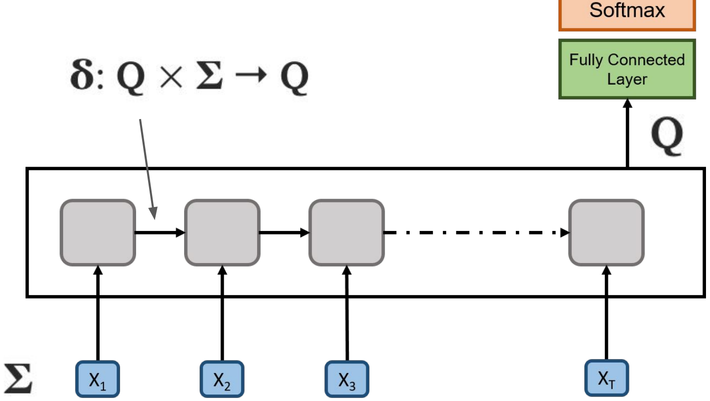
RNN - Automata Notations

Notations

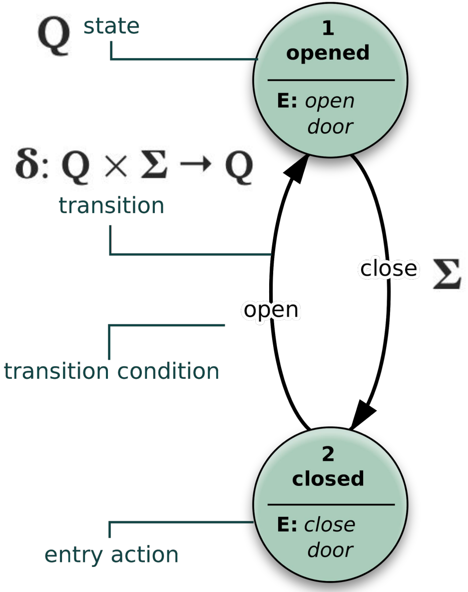
5 tuple $\langle \mathbf{Q}, \mathbf{\Sigma}, \boldsymbol{\delta}, \mathbf{qo}, \mathbf{F} \rangle$

and f(**Q**) --> {Accept, Reject} s.t f(**Q**) == 1 if **Q** in **F**

RNN (R)



DFA (L)



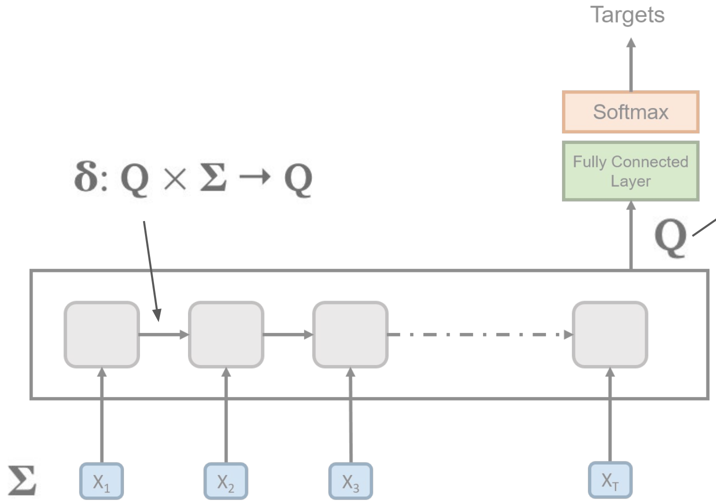
Most importantly, the
hidden state of RNN = each state of DFA

https://commons.wikimedia.org/wiki/File:Finite_state_machine_example_with_comments.svg
https://www.arxiv-vanity.com/papers/1801.08322/

Getting the classification decision

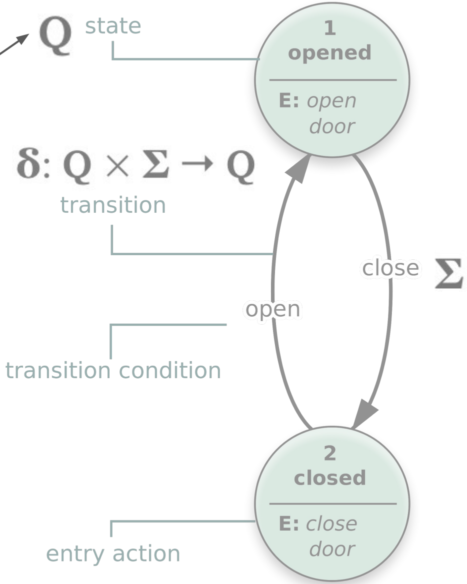
RNN (R)

Each hidden vector:
“Am I the final state?”
f(Q) = {0,1}



DFA (L)

Each discrete state:
“Am I the final state?”
f(Q) = {0,1}



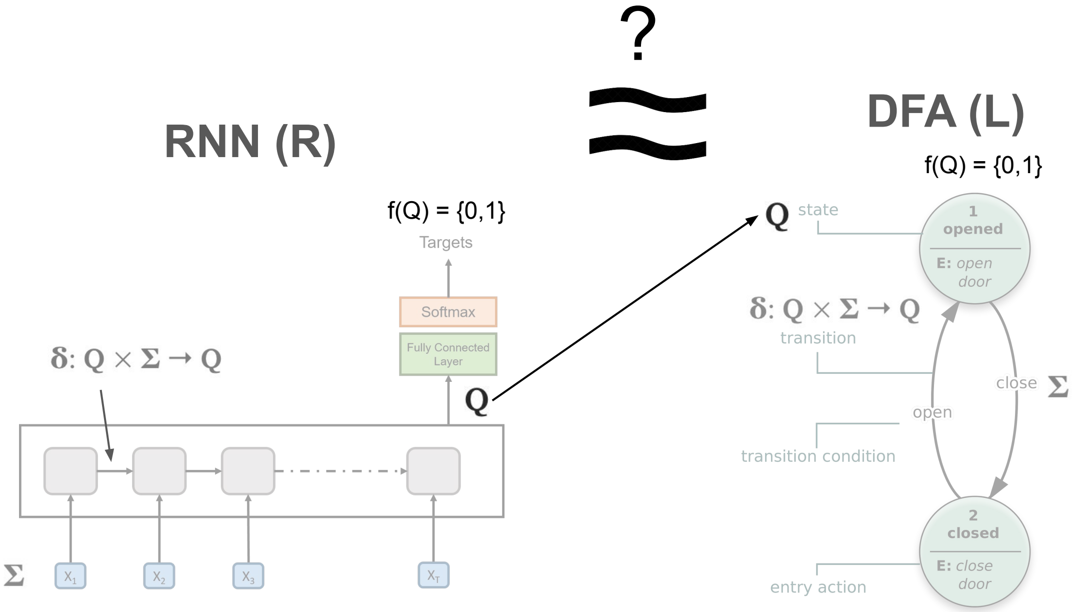
https://commons.wikimedia.org/wiki/File:Finite_state_machine_example_with_comments.svg
https://www.arxiv-vanity.com/papers/1801.08322/

How do we map from R to L?

We need to answer equivalence question based on their classifications:

Go from continuous hidden vectors (R) to discrete states in DFA (L):

We need Abstractions (A) i.e. discretization of states of R.

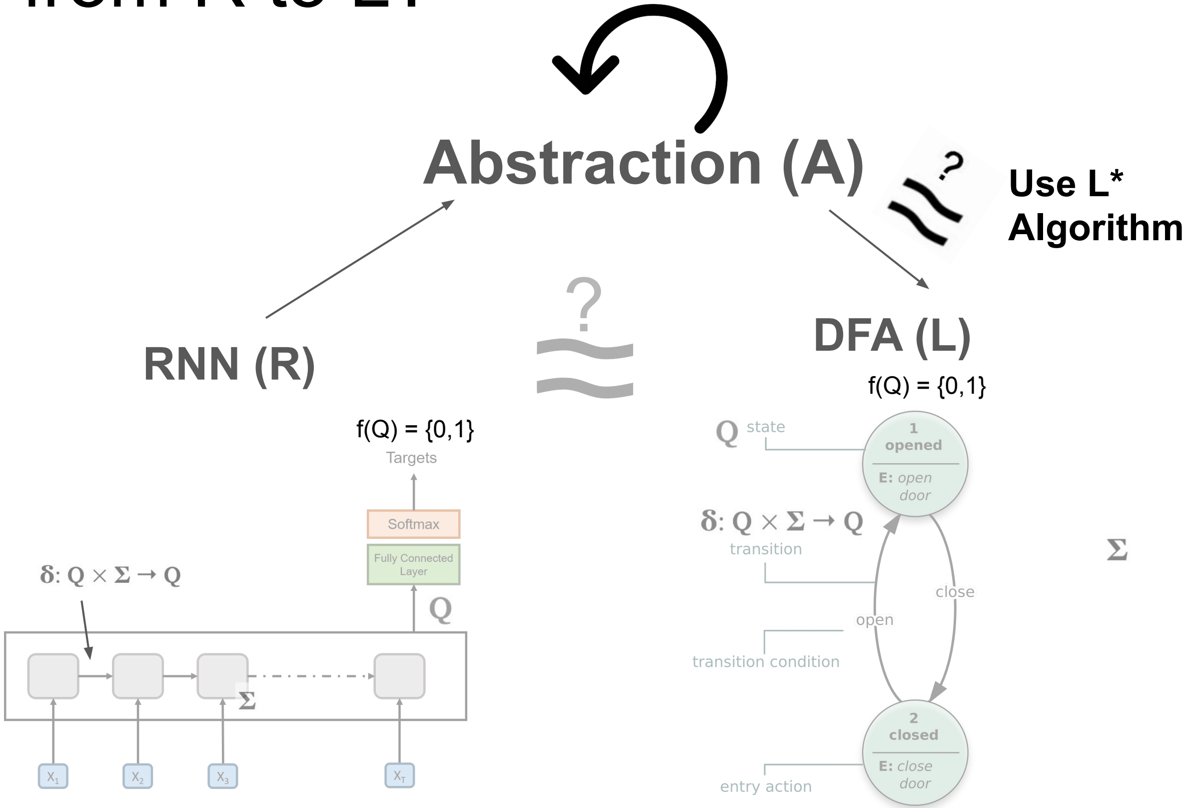


https://commons.wikimedia.org/wiki/File:Finite_state_machine_example_with_comments.svg
https://www.arxiv-vanity.com/papers/1801.08322/

How do we map from R to L?

Approximate R using A and try to answer the simpler question:
is A == L?

This question can be answered using L*



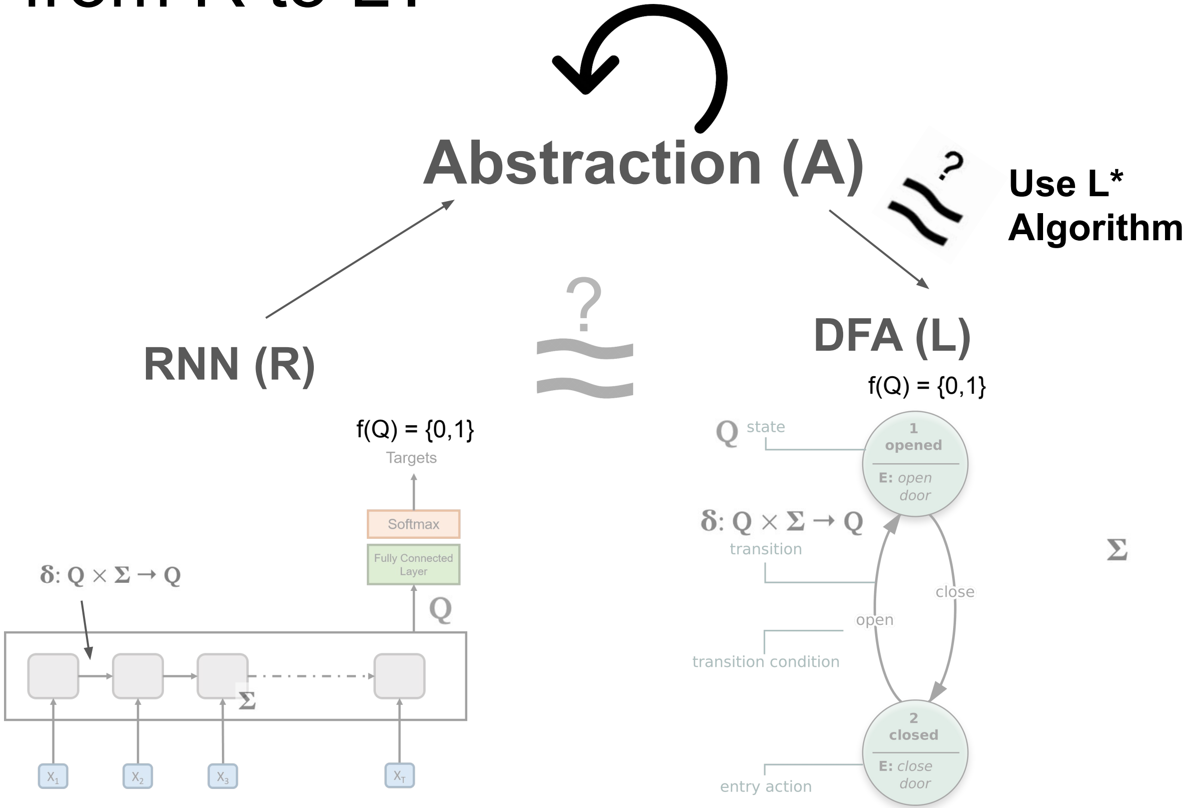
https://commons.wikimedia.org/wiki/File:Finite_state_machine_example_with_comments.svg
https://www.arxiv-vanity.com/papers/1801.08322/

How do we map from R to L?

After comparing classifications, approximation can result in

counter examples i.e. L != A → find new L

or **refinement of abstraction** i.e. L = A after finding new A



https://commons.wikimedia.org/wiki/File:Finite_state_machine_example_with_comments.svg
https://www.arxiv-vanity.com/papers/1801.08322/

Results

Brief Recap of Findings

Classification question: Does the input sequence belong to a Tomita Grammar?

RNN: Binary Classification

DFA: Reached Accept State or Not

1. **Random Regular Languages**: Reference Grammars have 5 state DFA over 2 letter alphabet

   Overall, RNN trained

   to 100% accuracy

Table 1. Accuracy of DFAs extracted from GRU networks representing small regular languages. Single values represent the average of 3 experiments, multiple values list the result for each experiment. Extraction time of 30 seconds is a timeout.

Hidden Size	Time (s)	DFA Size	Average Accuracy on Length 10	50	100	1000	Train
50	30, 30, 30	11,11,155	99.9	99.8	99.9	99.9	99.9
100	11.0	11,10,11	100	99.9	99.9	99.9	100
500	30, 30, 30	10,10,10	100	99.9	100	99.9	100.0

Brief Recap of Findings

2. **Comparison with a-priori Quantization**: Network state space divided into q equal intervals. A different method of network abstraction than that proposed in this paper.

   This paper: extracted small and accurate DFAs in 30s

   A-priori: With quantization of 2, time limit of 1000s was not enough and extracted DFAs were large (60,000 states) and sequences of length 1000 would get 0% accuracy. For others, 99%+

Brief Recap of Findings

3. **Comparison with Random Sampling**: For counterexample generation, their method is superior to random sampling, which could often become intractable.

Table 2. Accuracy and maximum nesting depth of extracted automata for networks trained on BP, using either abstractions ("Abstr") or random sampling ("RS") for equivalence queries. Accuracy is measured with respect to the trained RNN.

	Train Set Accuracy		Max Nest. Depth	
Network	Abstr	RS	Abstr	RS
GRU	99.98	87.12	8	2
LSTM	99.98	94.19	8	3

Brief Recap of Findings

3. **Comparison with Random Sampling**: For counterexample generation, their method is superior to random sampling (RS), which could often become intractable. Their method is also able to find adversarial inputs compared to none for RS.

Table 2. Accuracy and maximum nesting depth of extracted automata for networks trained on BP, using either abstractions ("Abstr") or random sampling ("RS") for equivalence queries. Accuracy is measured with respect to the trained RNN.

	Train Set Accuracy		Max Nest. Depth	
Network	Abstr	RS	Abstr	RS
GRU	99.98	87.12	8	2
LSTM	99.98	94.19	8	3

Brief Recap of Limitations

Due to L* polynomial complexity:

- Extraction can be very slow
- Large DFAs can be returned

When RNN **doesn't generalize well to input,** this method finds various adversarial inputs, builds a **large DFA** and **times out**.

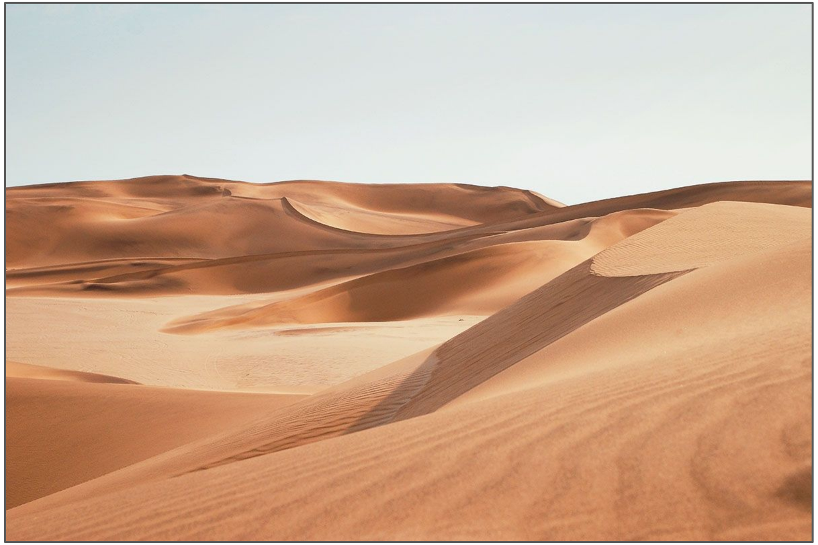
Takeaway? RNNs are brittle and test set performance evidence should be interpreted with extreme caution.

Breakout Room Activity

1. Where does model distillation fit in with the symbolism vs connectionism debate?
2. Were we successfully able to show equivalence between symbolic and connectionist architectures?

What Is One Grain of Sand in the Desert?

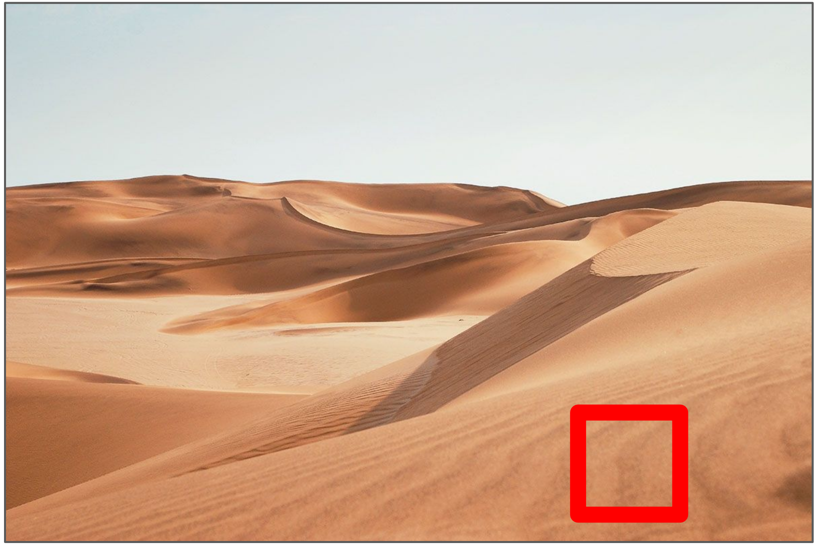
Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, James Glass



Neural networks learn **distributed representations**.

Neural networks learn **distributed representations**.

Many neurons, or “grains of sand,” comprise the meaning, or “the desert.”

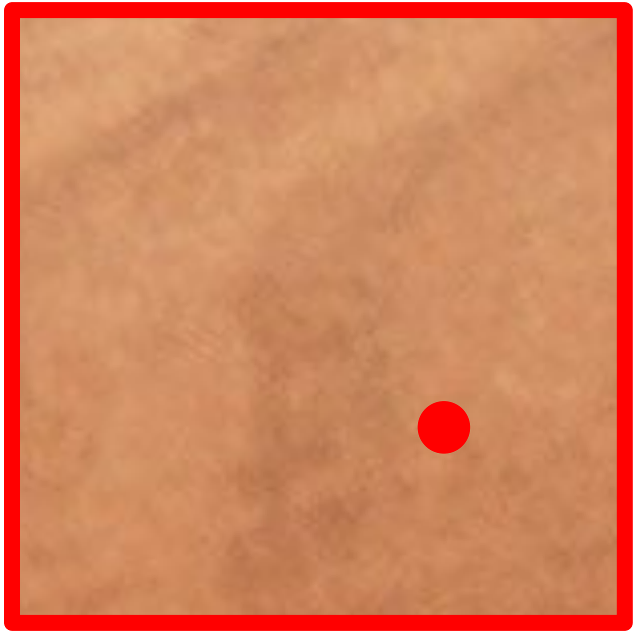


Neural networks learn **distributed representations**.

If we zoom in on a small slice of the representation, what would we find?

Neural networks learn **distributed representations**.

If we zoom in on a small slice of the representation, what would we find?



Neural networks learn **distributed representations**.

If we zoom in on a small slice of the representation, what would we find?

What if we look at only a **single neuron**?

Inside the black box

F&P argue that although neural networks can implement symbolic computation, they need not **explicitly represent** discrete symbols or operations on them.

Inside the black box

F&P argue that although neural networks can implement symbolic computation, they need not **explicitly represent** discrete symbols or operations on them.

However, it might be the case that neural networks **implicitly learn** to represent and manipulate discrete units.

Inside the black box

F&P argue that although neural networks can implement symbolic computation, they need not **explicitly represent** discrete symbols or operations on them.

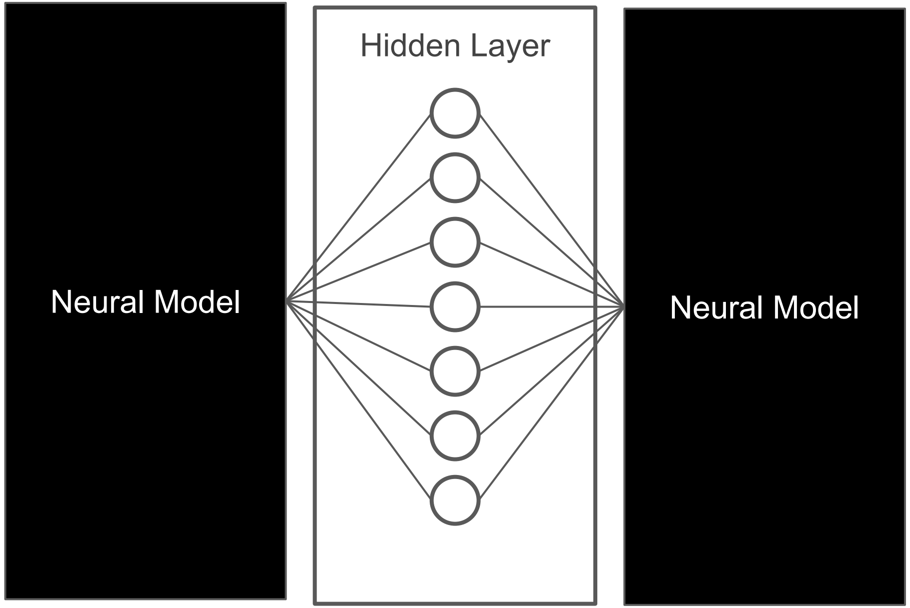
However, it might be the case that neural networks **implicitly learn** to represent and manipulate discrete units.

Here, we investigate whether neurons behave like discrete **concept detectors**, and whether this local representation mechanism determines network behavior.

Neurons as concept detectors

Consider a hidden layer in some neural network.

```
the
large
dog
ran
through
green
grass
```



Neurons as concept detectors

Consider a hidden layer in some neural network.

the

large

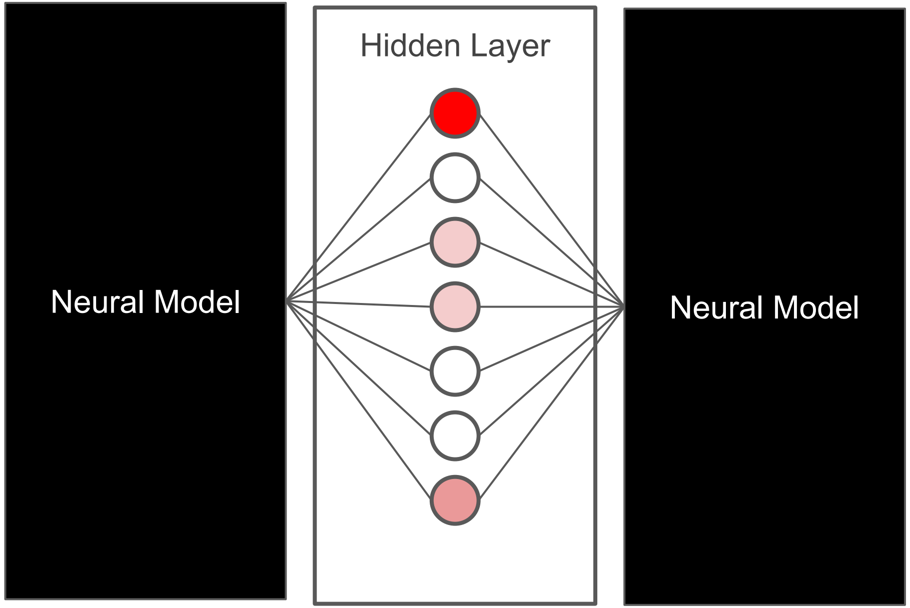
dog

ran

through

green

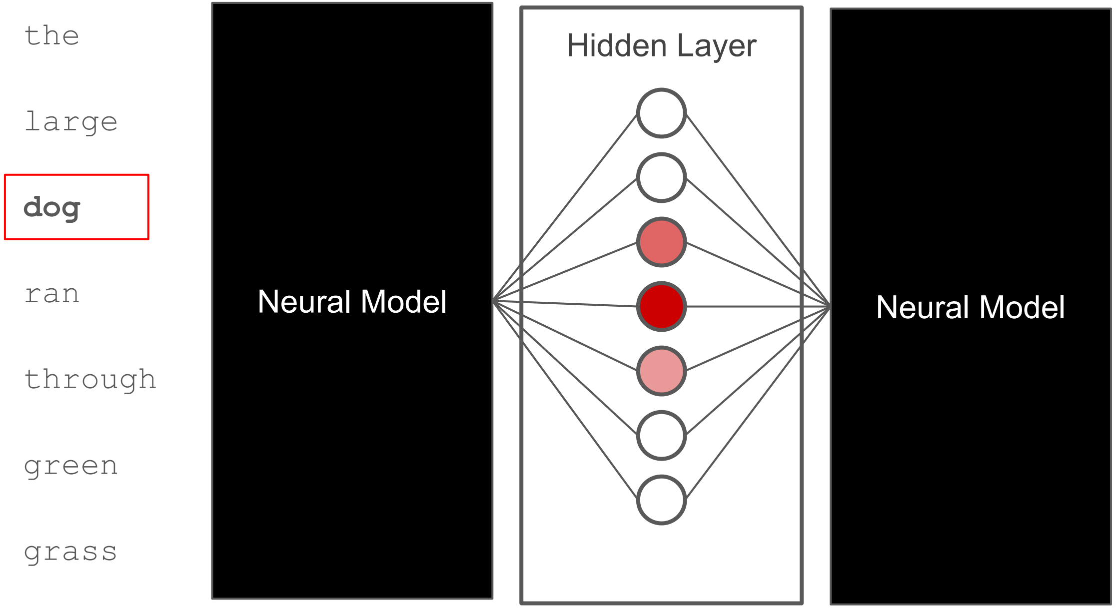
grass



In response to a stimulus (e.g. a word), it either does not fire or it fires with some magnitude.

Neurons as concept detectors

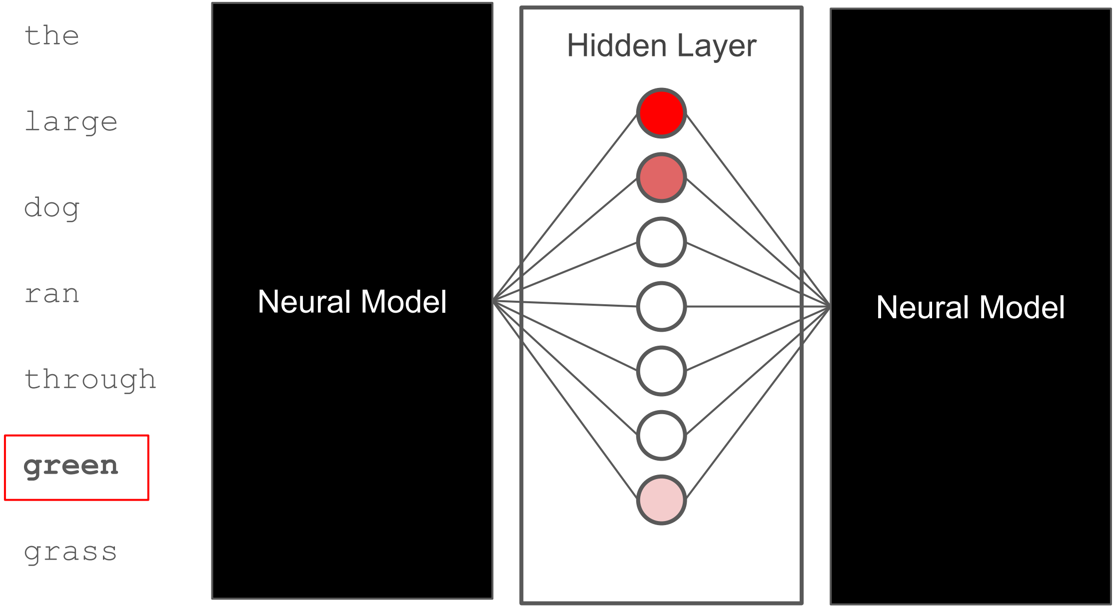
Consider a hidden layer in some neural network.



In response to a stimulus (e.g. a word), it either does not fire or it fires with some magnitude.

Neurons as concept detectors

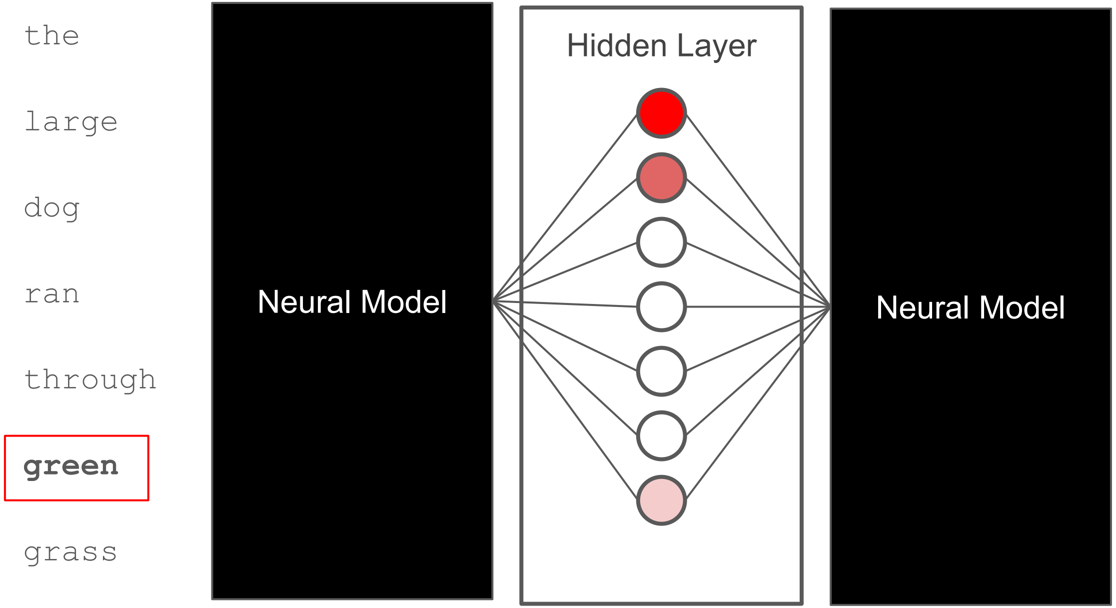
Consider a hidden layer in some neural network.



In response to a stimulus (e.g. a word), it either does not fire or it fires with some magnitude.

Neurons as concept detectors

Consider a hidden layer in some neural network.

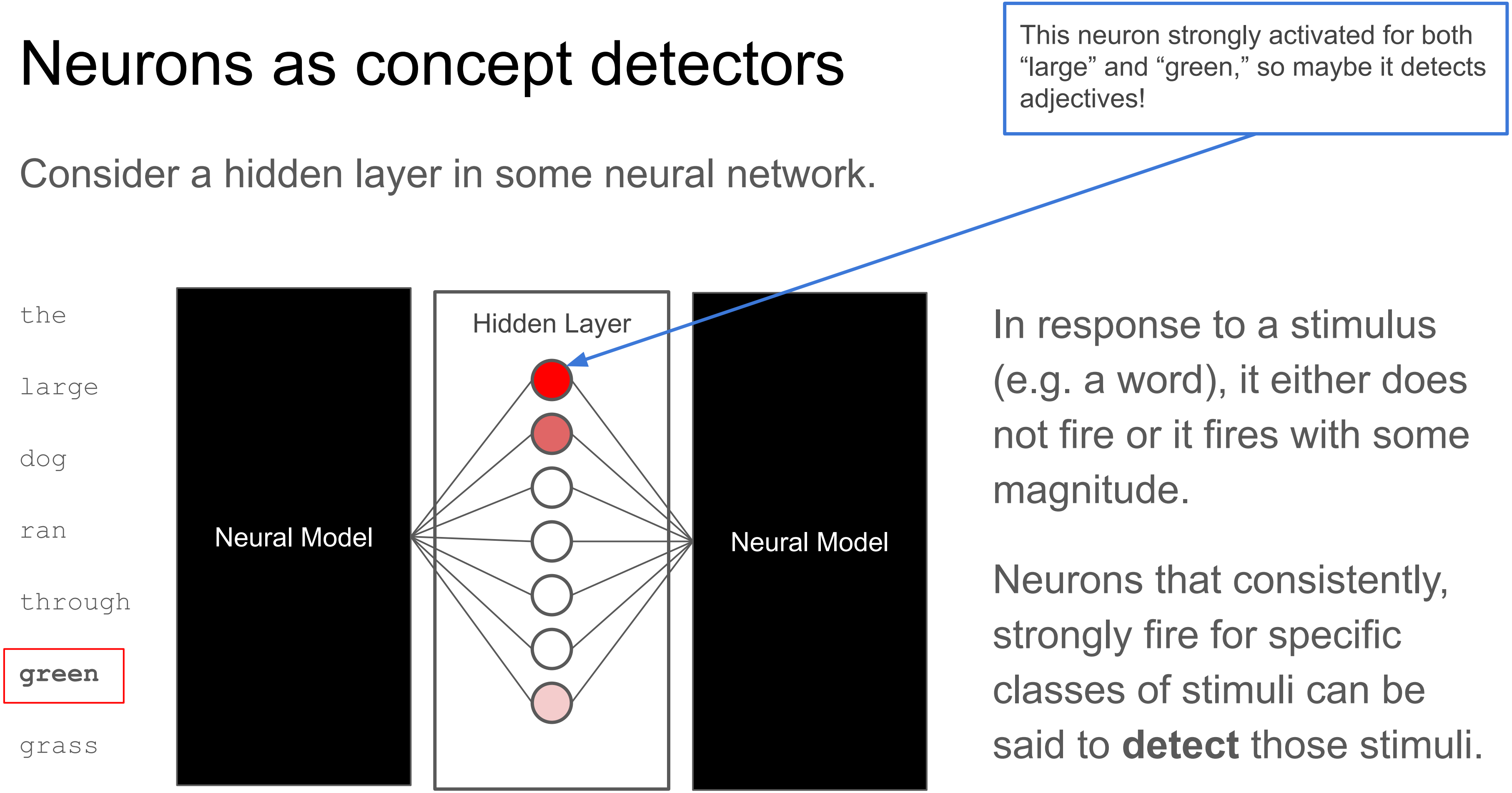


In response to a stimulus (e.g. a word), it either does not fire or it fires with some magnitude.

Neurons that consistently, strongly fire for specific classes of stimuli can be said to **detect** those stimuli.

Neurons as concept detectors

Consider a hidden layer in some neural network.



In response to a stimulus (e.g. a word), it either does not fire or it fires with some magnitude.

Neurons that consistently, strongly fire for specific classes of stimuli can be said to **detect** those stimuli.

Neurons as concept detectors

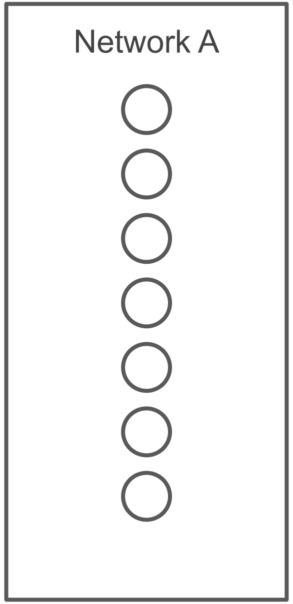
In the previous example, we saw neurons that detect specific parts of speech.

What if we don’t know what concepts to look for?

Neurons as concept detectors

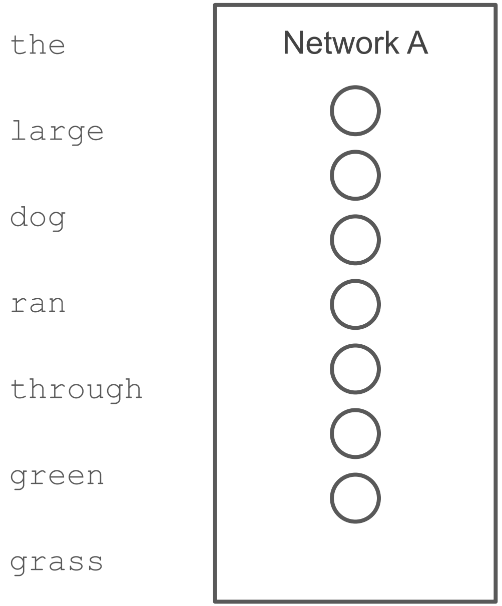
In the previous example, we saw neurons that detect specific parts of speech.
What if we don't know what concepts to look for?

```
the
large
dog
ran
through
green
grass
```



Neurons as concept detectors

In the previous example, we saw neurons that detect specific parts of speech.
What if we don't know what concepts to look for?

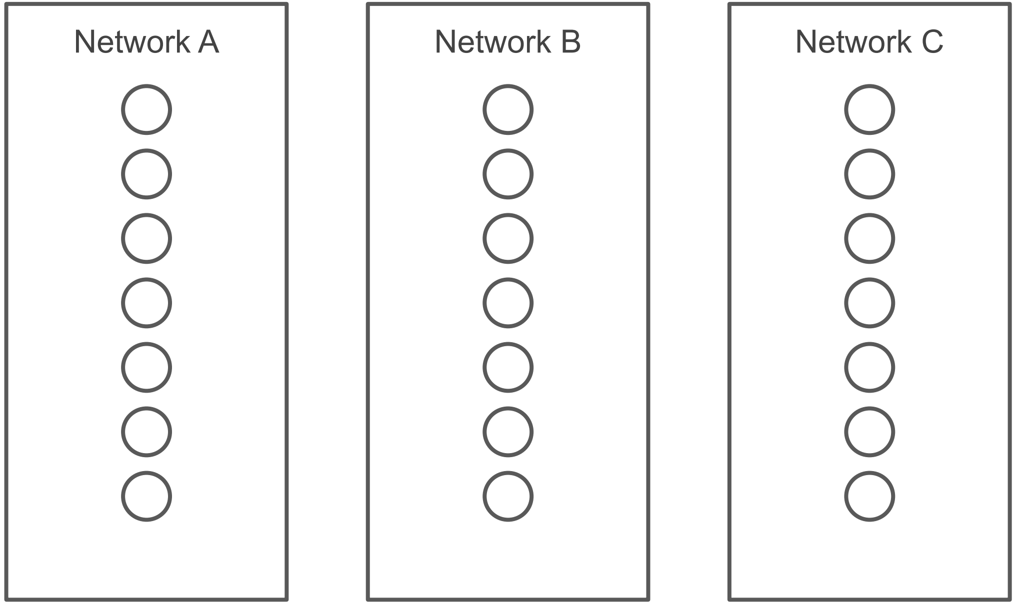


Idea: If the concept is important for the task, then any neural network solving the task should encode the concept.

Neurons as concept detectors

In the previous example, we saw neurons that detect specific parts of speech.
What if we don't know what concepts to look for?

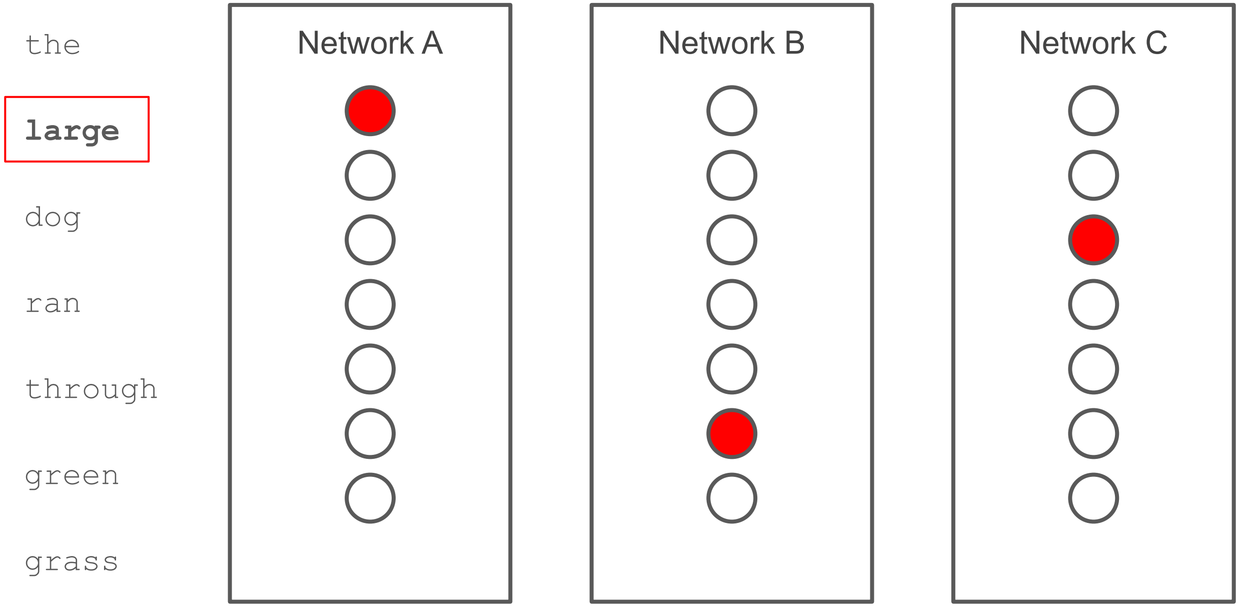
```
the
large
dog
ran
through
green
grass
```



Idea: If the concept is important for the task, then any neural network solving the task should encode the concept.

Neurons as concept detectors

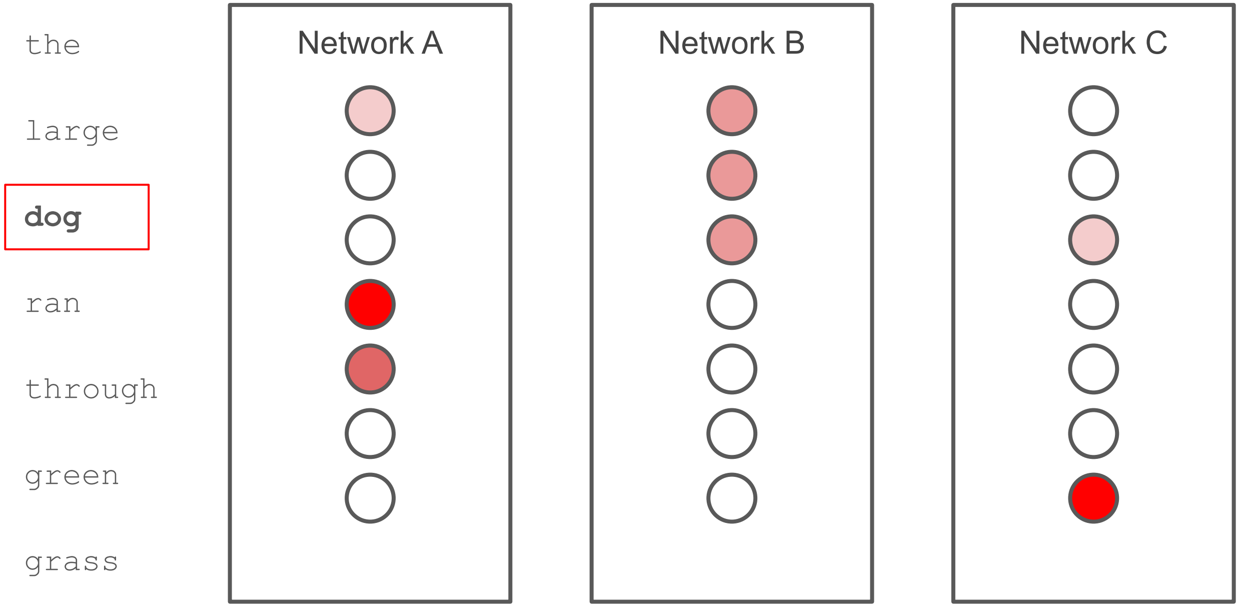
In the previous example, we saw neurons that detect specific parts of speech.
What if we don’t know what concepts to look for?



Idea: If the concept is important for the task, then any neural network solving the task should encode the concept.

Neurons as concept detectors

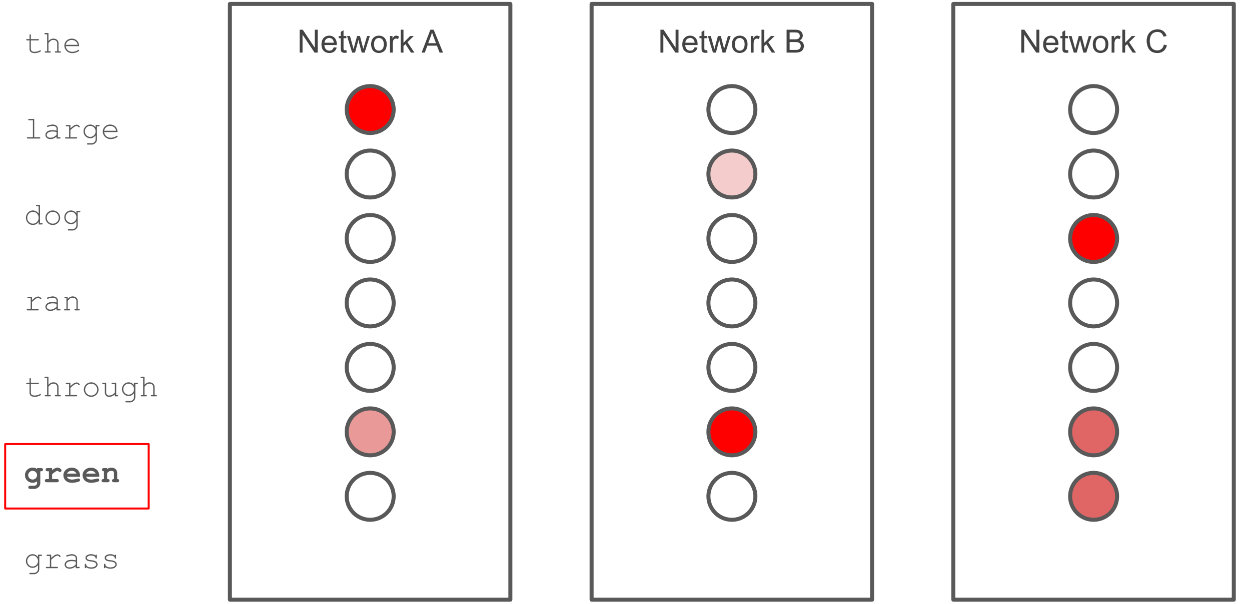
In the previous example, we saw neurons that detect specific parts of speech.
What if we don't know what concepts to look for?



Idea: If the concept is important for the task, then any neural network solving the task should encode the concept.

Neurons as concept detectors

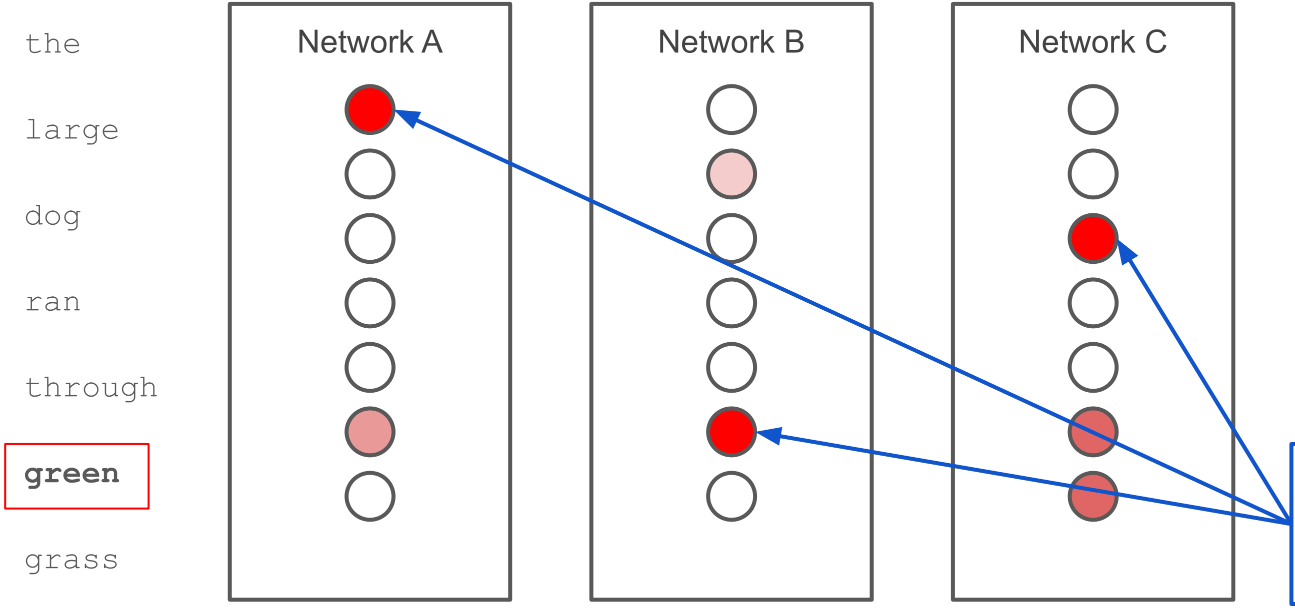
In the previous example, we saw neurons that detect specific parts of speech.
What if we don't know what concepts to look for?



Idea: If the concept is important for the task, then any neural network solving the task should encode the concept.

Neurons as concept detectors

In the previous example, we saw neurons that detect specific parts of speech.
What if we don't know what concepts to look for?



Idea: If the concept is important for the task, then any neural network solving the task should encode the concept.

These neurons tend to fire together, so they probably encode the same (important) thing.

Discussion

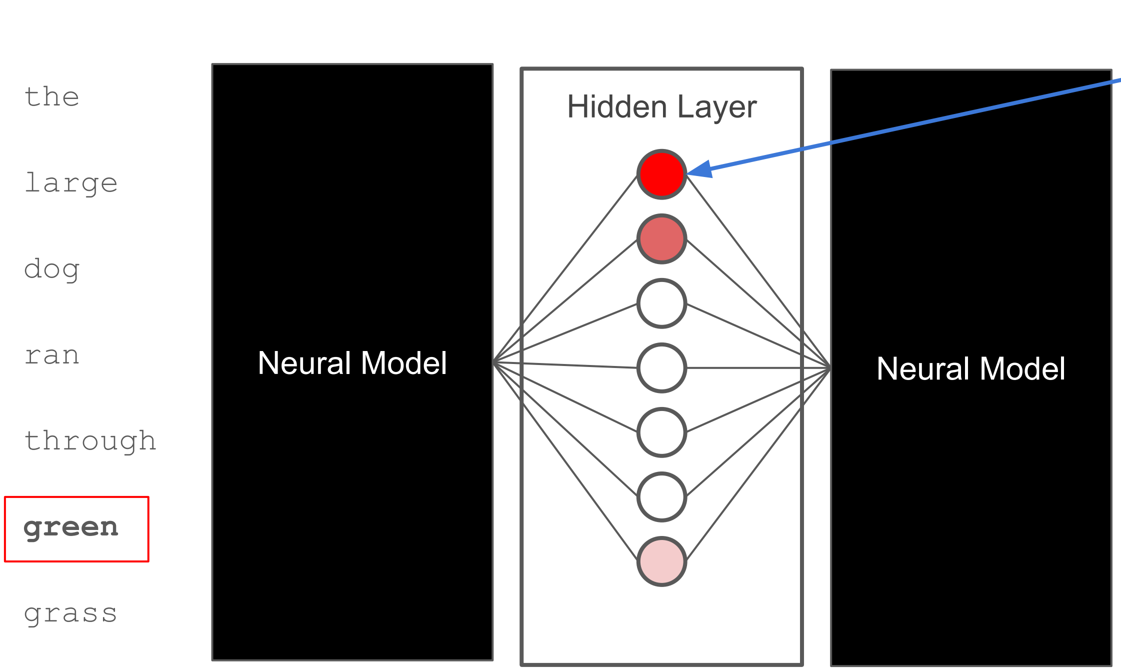
10 minutes

Before we dive into experiments:

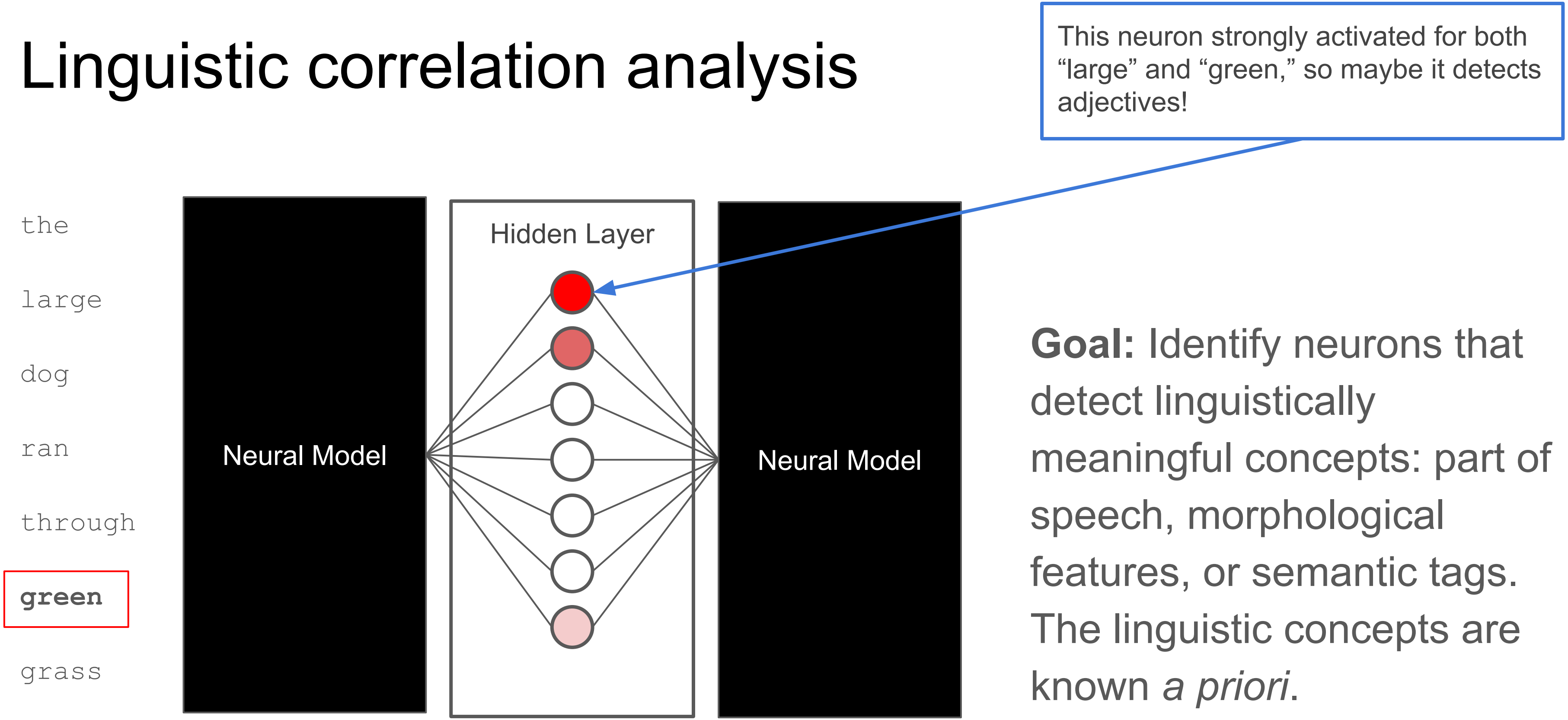
- Is this a reasonable way to interpret neuron activations?
- We've described a sort of local representation; can we call it "symbolic"?

Linguistic correlation analysis

This neuron strongly activated for both “large” and “green,” so maybe it detects adjectives!



Linguistic correlation analysis



Goal: Identify neurons that detect linguistically meaningful concepts: part of speech, morphological features, or semantic tags. The linguistic concepts are known *a priori*.

Setup

Sequence of words $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$

Setup

Sequence of words $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$

Set of word and label tuples $(\mathbf{x}_i, \mathbf{l}_i)$

Setup

Sequence of words $(\mathbf{x_1, \ldots, x_n})$

Set of word and label tuples $(\mathbf{x_i, l_i})$

E.g., ("green", JJ) for POS. The authors experiment with **POS** and **semantic** tags.

Setup

Sequence of words $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$

Set of word and label tuples $(\mathbf{x}_i, \mathbf{l}_i)$

E.g., ("green", JJ) for POS. The authors experiment with **POS** and **semantic** tags.

Model **f** mapping words to vector representations $\mathbf{f}(\mathbf{x}_i) = \mathbf{z}_i$

Setup

Sequence of words $(\mathbf{x_1, \ldots, x_n})$

Set of word and label tuples $(\mathbf{x_i, l_i})$

Model **f** mapping words to vector representations $\mathbf{f(x_i) = z_i}$

E.g., ("green", JJ) for POS. The authors experiment with **POS** and **semantic** tags.

E.g., the hidden state of an RNN after the **i**-th input. The authors use the hidden states of RNNs trained on **MT** (EN → FR, DE → EN) and **LM**.

Method

Train logistic regression classifier on **(z_i, l_i)** pairs

Method

Train logistic regression classifier on **(z_i, l_i)** pairs

Minimize regularized cross entropy:

$$\mathcal{L}(\theta) = -\sum_i \log P_\theta(\mathbf{l}_i|\mathbf{x}_i) + \lambda_1\|\theta\|_1 + \lambda_2\|\theta\|_2^2$$

Method

Train logistic regression classifier on **(z_i, l_i)** pairs

Minimize regularized cross entropy:

$$\mathcal{L}(\theta) = -\sum_i \log P_\theta(\mathbf{l}_i|\mathbf{x}_i) + \boxed{\lambda_1\|\theta\|_1} + \lambda_2\|\theta\|_2^2$$

Encourages sparsity, i.e.
selection of only a few neurons

Results: classifier accuracy

	French		English		German	
	POS	Morph	POS	SEM	POS	Morph
MAJ	92.8	89.5	91.6	84.2	89.3	83.7
NMT	93.2	88.0	93.5	90.1	93.6	87.3
NLM	92.4	90.1	92.9	86.0	92.3	86.5

Table 1: Classifier accuracy when trained on activations of NMT and NLM models. MAJ: local majority baseline.

Takeaway: The neural representations do contain (potentially distributed) signal about part of speech, morphology, and semantic tags.

Results: ablating important neurons

			Masking-out					
Task		ALL	10%		15%		20%	
			Top	Bot	Top	Bot	Top	Bot
NMT	FR (POS)	93.2	63.2	23.8	73.0	24.8	79.4	24.9
	EN (POS)	93.5	69.8	15.8	78.3	17.9	84.1	21.5
	EN (SEM)	90.1	51.5	16.3	65.3	18.9	74.2	20.7
	DE (POS)	93.6	65.9	15.7	78.0	15.6	88.2	15.7
NLM	FR (POS)	92.4	41.6	23.8	53.6	23.8	59.6	24.0
	EN (POS)	92.9	54.2	18.4	66.1	20.4	72.4	24.7
	EN (SEM)	86.0	49.7	21.9	56.8	22.3	65.2	25.1
	DE (POS)	92.3	39.7	16.7	51.7	16.7	67.2	16.9

Table 2: Classification accuracy on different tasks using all neurons (ALL). Masking-out: all except top/bottom N% of neurons are masked when testing the trained classifier.

Takeaway 1: The MT and LM systems do distribute information across neurons.

Results: ablating important neurons

	Task	ALL	Masking-out					
			10%		15%		20%	
			Top	Bot	Top	Bot	Top	Bot
NMT	FR (POS)	93.2	63.2	23.8	73.0	24.8	79.4	24.9
	EN (POS)	93.5	69.8	15.8	78.3	17.9	84.1	21.5
	EN (SEM)	90.1	51.5	16.3	65.3	18.9	74.2	20.7
	DE (POS)	93.6	65.9	15.7	78.0	15.6	88.2	15.7
NLM	FR (POS)	92.4	41.6	23.8	53.6	23.8	59.6	24.0
	EN (POS)	92.9	54.2	18.4	66.1	20.4	72.4	24.7
	EN (SEM)	86.0	49.7	21.9	56.8	22.3	65.2	25.1
	DE (POS)	92.3	39.7	16.7	51.7	16.7	67.2	16.9

Table 2: Classification accuracy on different tasks using all neurons (ALL). Masking-out: all except top/bottom N% of neurons are masked when testing the trained classifier.

Takeaway 2: ...but the systems rely more on neurons that detect linguistically meaningful symbols.

Examples of linguistically meaningful neurons

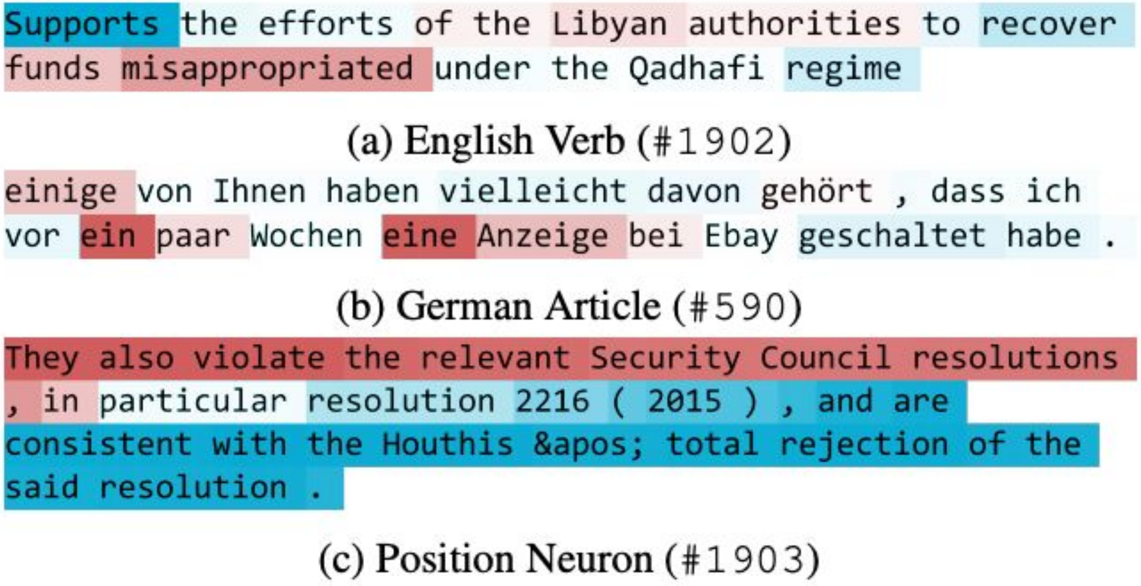
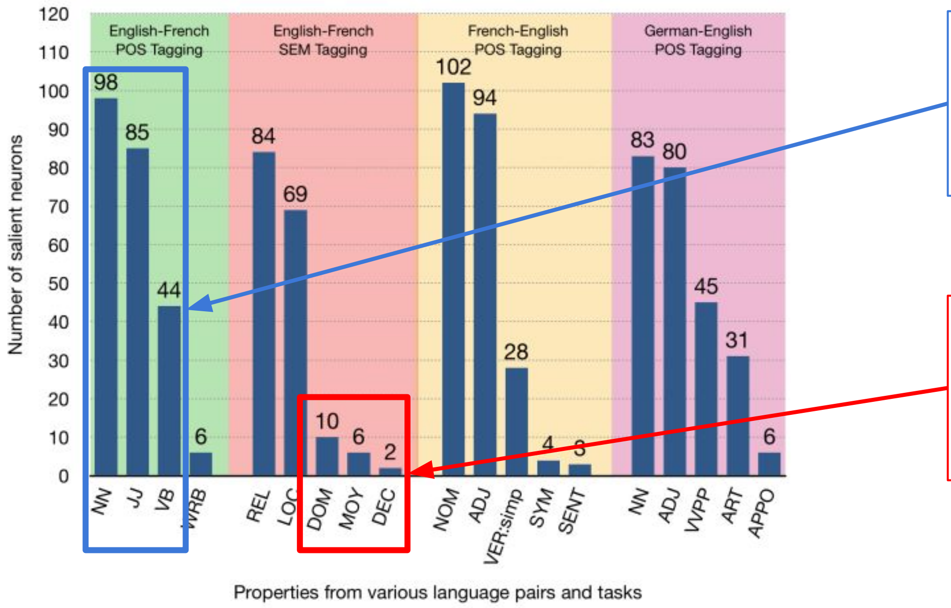


Figure 3: Activations of top neurons for specific properties

Which linguistic concepts are most distributed?



Information about **open-class** categories (e.g. noun and verb parts of speech) is highly distributed.

Information about **closed-class** categories (e.g. month of year, end of sentence) is local to a few neurons.

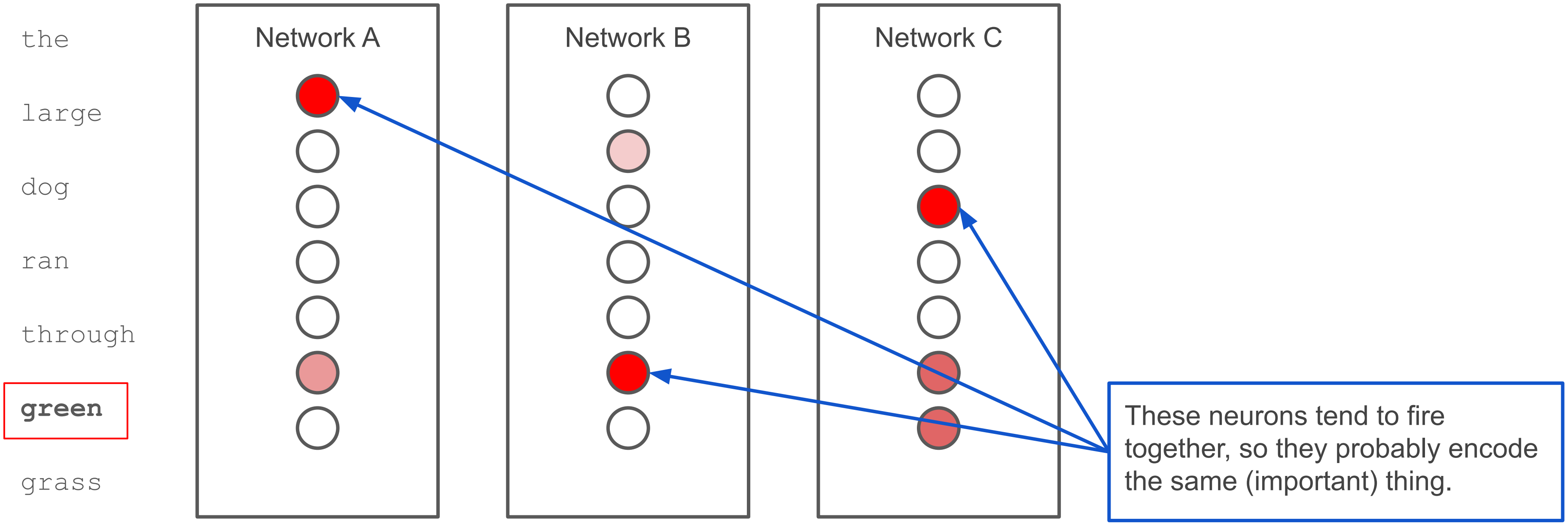
Discussion

10 minutes

Model performance still drops substantially when the least salient neurons are ablated. What can we conclude?

Why should open class concepts (e.g. noun/verb POS) be more distributed than closed class concepts?

Cross-model correlations



Method

Train the same architecture on the original task with multiple random seeds.

Method

Train the same architecture on the original task with multiple random seeds.

In each model, look for neurons whose activations are highly correlated with a neuron from a different initialization.

$$score(\mathbb{M}_{ij}) = \max_{\substack{1 \leq i' \leq N \\ 1 \leq j' \leq D \\ i \neq i'}} \rho(\mathbb{M}_{ij}, \mathbb{M}_{i'j'})$$

Method

Train the same architecture on the original task with multiple random seeds.

In each model, look for neurons whose activations are highly correlated with a neuron from a different initialization.

Activation values for **i**-th model, **j**-ith neuron

$$score(\mathbb{M}_{ij}) = \max_{\substack{1 \le i' \le N \\ 1 \le j' \le D \\ i \ne i'}} \rho(\boxed{\mathbb{M}_{ij}}, \mathbb{M}_{i'j'})$$

Method

Train the same architecture on the original task with multiple random seeds.

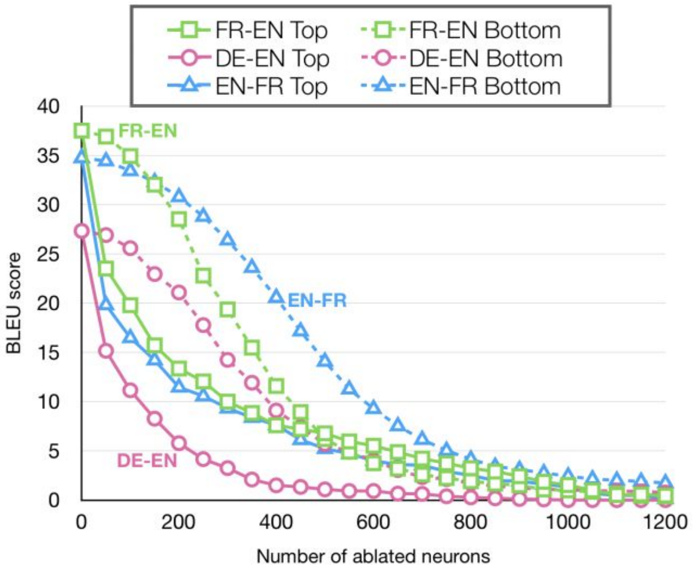
In each model, look for neurons whose activations are highly correlated with a neuron from a different initialization.

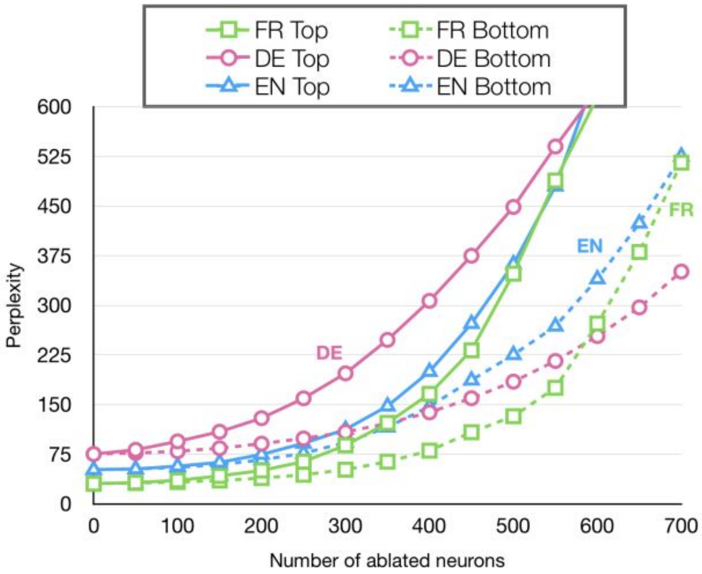
Activation values for **i**-th model, **j**-ith neuron

$$score(\mathbb{M}_{ij}) = \max_{\substack{1 \leq i' \leq N \\ 1 \leq j' \leq D \\ i \neq i'}} \rho(\boxed{\mathbb{M}_{ij}}, \mathbb{M}_{i'j'})$$

Same architectures (RNNs) and tasks (LM/MT) as before.

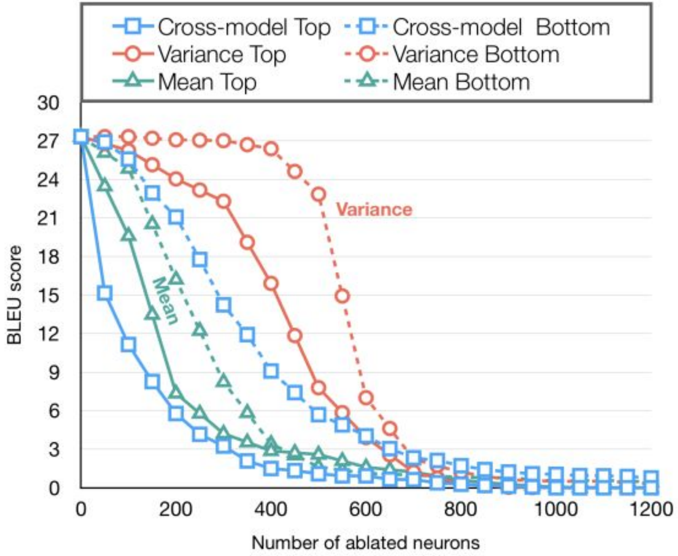
Results: ablating correlated neurons





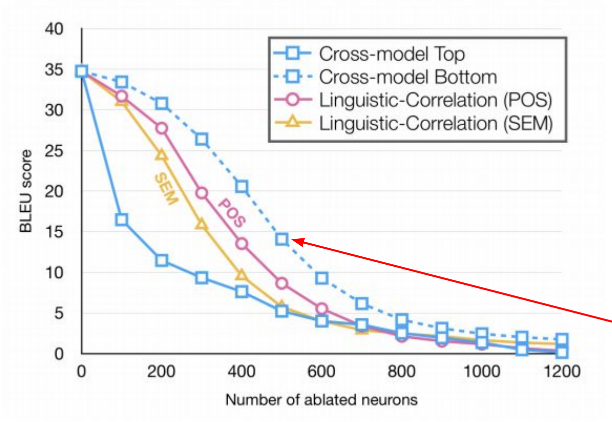
Takeaway: Cross-model correlations select for salient neurons, and the network is most sensitive to the most correlated neurons. These neurons likely select for task-essential concepts.

Results: comparison to single-model correlations



Takeaway: We're not hallucinating. Neurons with cross-model correlation select for more task-essential concepts than e.g. the highest variance neurons.

Results: comparison to linguistic correlations



Takeaway: Some classes of neurons are more essential for NMT than others.

In particular, the model relies most neurons with cross-model correlations. These probably select for concepts essential to MT.

Breakout Rooms

For the remaining time...

Is it fair to assume different initializations of an NN will learn similar concept detectors?

How does this method for identifying symbolic computation compare to the method used in [Weiss et al., 2018]?

These results are somewhat noisy; can we conclude these models are learning discrete structures?

Appendix

	Task	ALL	Re-training					
			10%		15%		20%	
			Top	Bot	Top	Bot	Top	Bot
NMT	FR (POS)	93.2	88.4	72.1	90.0	77.8	91.1	81.8
	EN (POS)	93.5	89.1	80.6	90.5	84.8	91.2	87.2
	EN (SEM)	90.1	85.6	73.4	87.0	77.8	87.8	80.8
	DE (POS)	93.6	91.4	77.1	92.3	81.9	92.8	85.3
NLM	FR (POS)	92.4	83.7	61.8	86.2	71.7	87.8	77.4
	EN (POS)	92.9	85.8	62.4	88.2	72.5	89.4	79.2
	EN (SEM)	86.0	78.9	67.8	81.4	74.1	82.7	77.6
	DE (POS)	92.3	87.2	41.7	89.6	67.0	90.4	76.5

Table 4: Classification accuracy on different tasks using all neurons (ALL). Re-training: only top/bottom N% of neurons are kept and the classifier is retrained

	Task	ALL	Masking-out					
			10%		15%		20%	
			Top	Bot	Top	Bot	Top	Bot
NMT	FR (Morph)	88.0	25.2	17.3	39.0	20.3	56.3	24.3
	DE (Morph)	87.3	21.8	15.7	33.3	20.8	53.2	29.3
NLM	FR (Morph)	90.1	36.3	13.9	45.1	15.5	58.4	19.0
	DE (Morph)	86.5	24.2	10.7	40.7	13.0	52.8	19.2

Table 5: Classification accuracy on morphological tags for French and German using all neurons (ALL). Masking-out: all except top/bottom N% of neurons are masked when testing the trained classifier.

	Task	ALL	Retraining 10% Top	Retraining 10% Bot	Retraining 15% Top	Retraining 15% Bot	Retraining 20% Top	Retraining 20% Bot
NMT	FR (Morph)	88.0	73.5	65.8	78.0	71.6	80.6	75.1
NMT	DE (Morph)	87.3	79.3	75.4	82.1	78.9	83.5	80.5
NLM	FR (Morph)	90.1	79.5	61.6	82.5	70.3	84.9	75.7
NLM	DE (Morph)	86.5	78.3	66.1	81.6	72.4	83.0	77.1

Table 6: Classification accuracy on morphological tags for French and German using all neurons (ALL). Re-training: only top/bottom N% of neurons are kept and the classifier is retrained