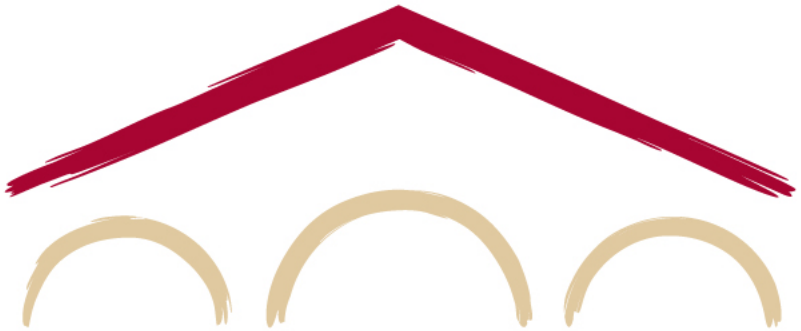
Natural Language Processing has been overrun by large neural language models! What should we make of that?



Christopher Manning

@chrmanning ❀ @stanfordnlp

Departments of Linguistics and Computer Science, Stanford University

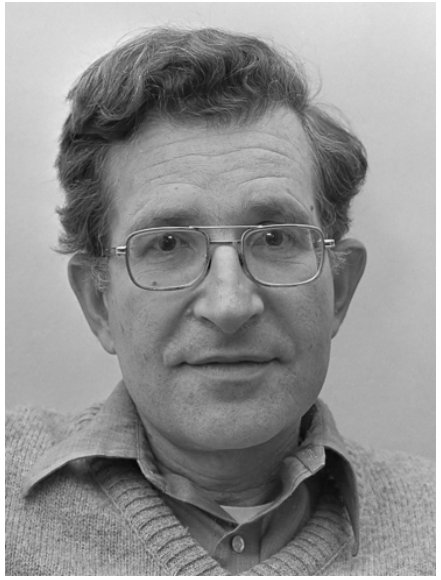
Director, Stanford Artificial Intelligence Laboratory

CUNY 2021

Claude Shannon vs. Noam Chomsky

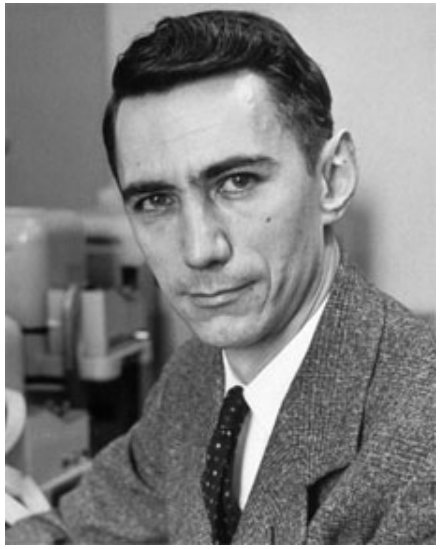
Chomsky (1969):

> “But it must be recognized that the notion of ‘probability of a sentence’ is an entirely useless one, under any known interpretation of this term”



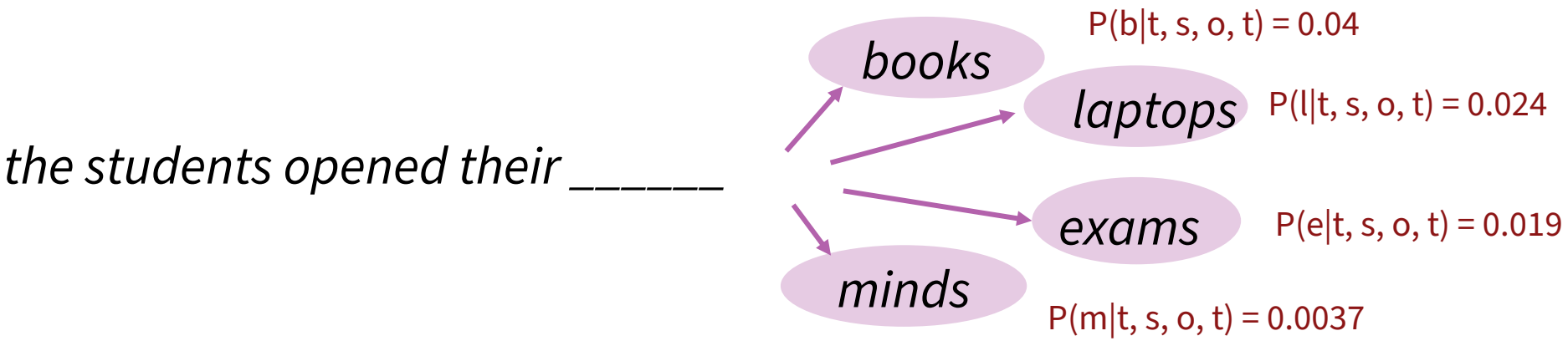
Shannon (1951):

> “anyone speaking a language possesses, implicitly, an enormous knowledge of the statistics of the language [enabling them] to complete an unfinished phrase”
>
> Useful in psycholinguistics as well as engineering!



Language Modeling

A **Language Model (LM)** predicts a word in a context



An LM is a key part of decoding tasks like **speech recognition**, **spelling correction**, and any language generation task, including **machine translation**, **summarization**, and **story generation**

LMs in The Dark Ages (20th C): *n*-gram models

Count how often words follow word sequences; divide to get cond. prob.

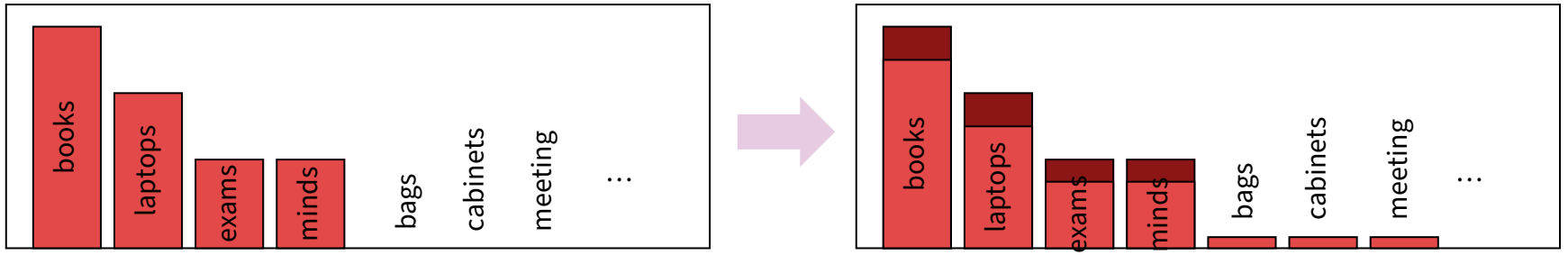
#(t, s, o, t, b) = 11 #(t, s, o, t) = 187 P(b | t, s, o, t) = 11/187 = 0.06

Classic **curse of dimensionality** scenario: zillions of params

Limiting context: Markov assumption:

$$P(x^{(t+1)}|\text{the students opened their}) \approx P(x^{(t+1)}|\text{opened their})$$

Discounting/Smoothing



Mixture/Backoff $P_{bo}(x^{(3)}|x^{(2)},x^{(1)}) \approx \lambda P(x^{(3)}|x^{(2)},x^{(1)}) + (1-\lambda)P(x^{(3)}|x^{(2)})$

N-gram models almost work as *n* increases

1-gram:

To him swallowed confess hear both. Which. Of save on trail for are …

2-gram:

Why dost stand forth thy canopy, forsooth; he is this palpable hit the King

3-gram:

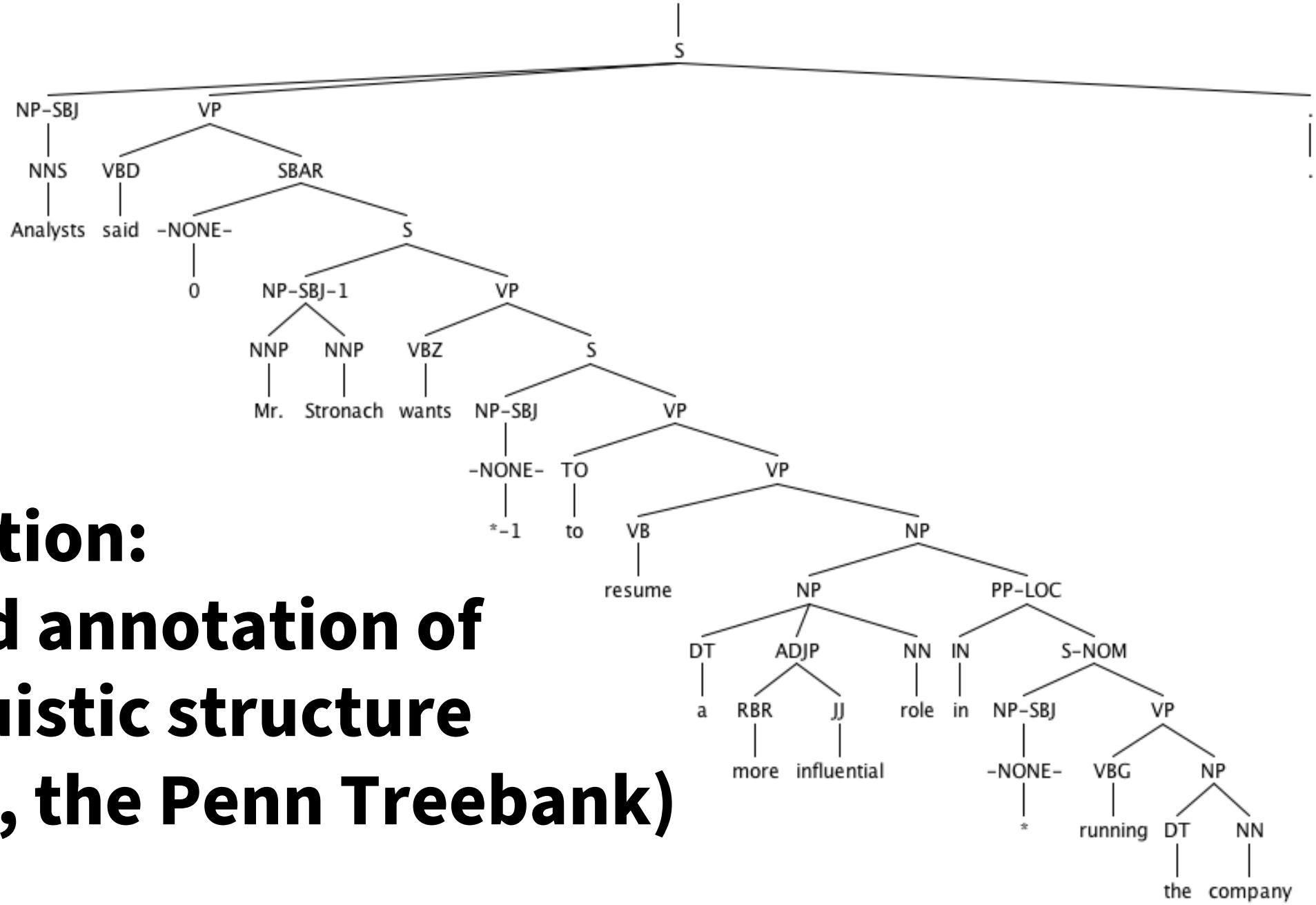
Fly, and will rid me these news of price. Therefore the sadness of parting,

4-gram:

King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch.

How much of the intricate structure of human languages do these language models know?

- (**Passionately argued!**) answer of linguists: **almost none**
  - Though n-gram models know quite a bit of simple world knowledge
    - The ship {sailed, sank, anchored, …}
  - And, in an unaggregated way, they know some low-level syntax
    - They know you tend to get sequences like:
      - preposition – article – noun
      - article – adjective – noun
  - They don't know the concept *noun*, or any sentence structure rules
    - As an abstracted grammar



Solution: Hand annotation of linguistic structure (e.g., the Penn Treebank)

Neural language models (NLMs)

1. **Solve the curse of dimensionality** by sharing of statistical strength via dense, low-dimensionality real-number word vectors $v_1, v_2, \ldots, v_K$ [Bengio, Ducharme, Vincent & Jauvin JMLR 2003, etc.]

$$P\left(x^{(t+1)} \middle| x^{(t)}, x^{(t-1)}\right) = \text{softmax}(\text{FFNN}(v^{(t)}, v^{(t-1)}))$$

2. **Solve the failure to exploit long contexts** via **recurrent NNs**

First, simple RNNs, soon usually LSTMs [Zaremba et al. 2014]; now transformers

*the same **stump** which had impaled the car of many a guest*
*in the past thirty years and which he refused to have **removed***

$$P\left(x^{(t+1)} \middle| x^{(\leq t)}\right) = \text{LSTM}(h^{(t)}, x^{(t)})$$

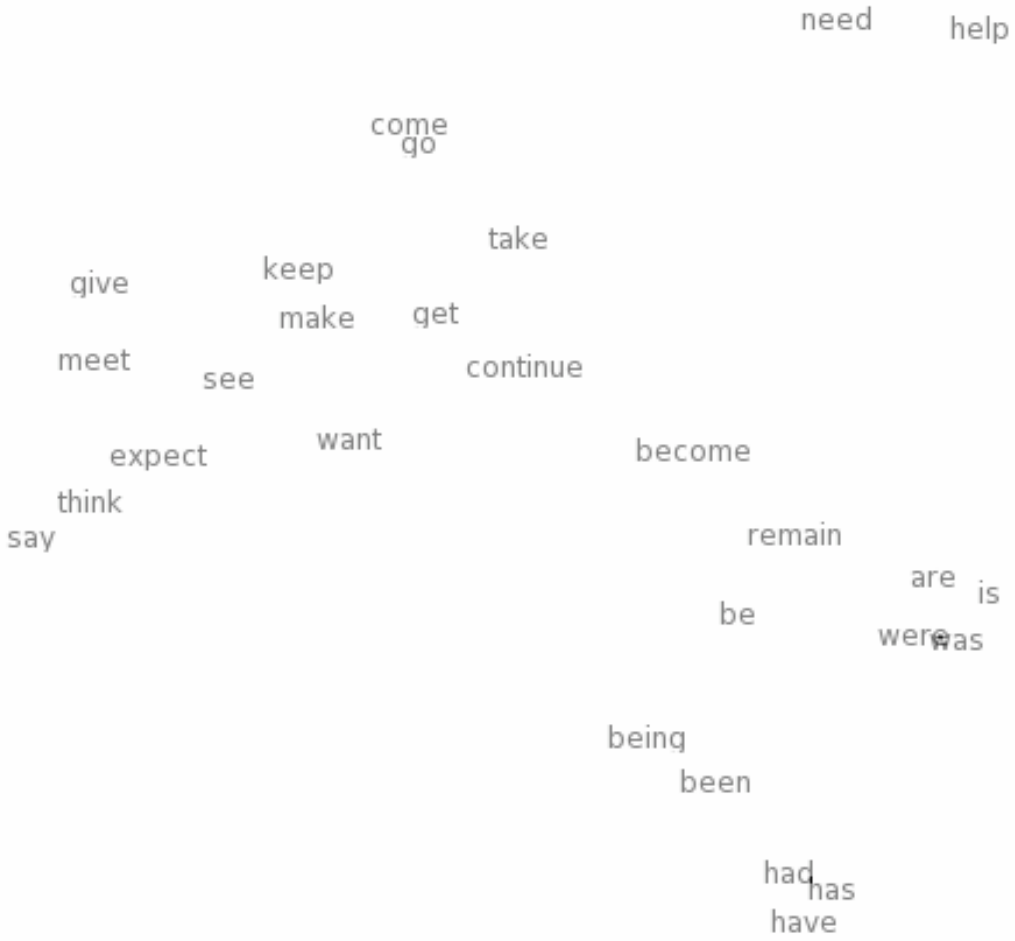
Word meaning as a neural word vector – visualization

expect =

$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

think =

$$\begin{pmatrix} 0.201 \\ 0.312 \\ -0.502 \\ 0.124 \\ 0.189 \\ -0.719 \\ 0.492 \\ 0.311 \\ 0.418 \end{pmatrix}$$



An RNN Language Model

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

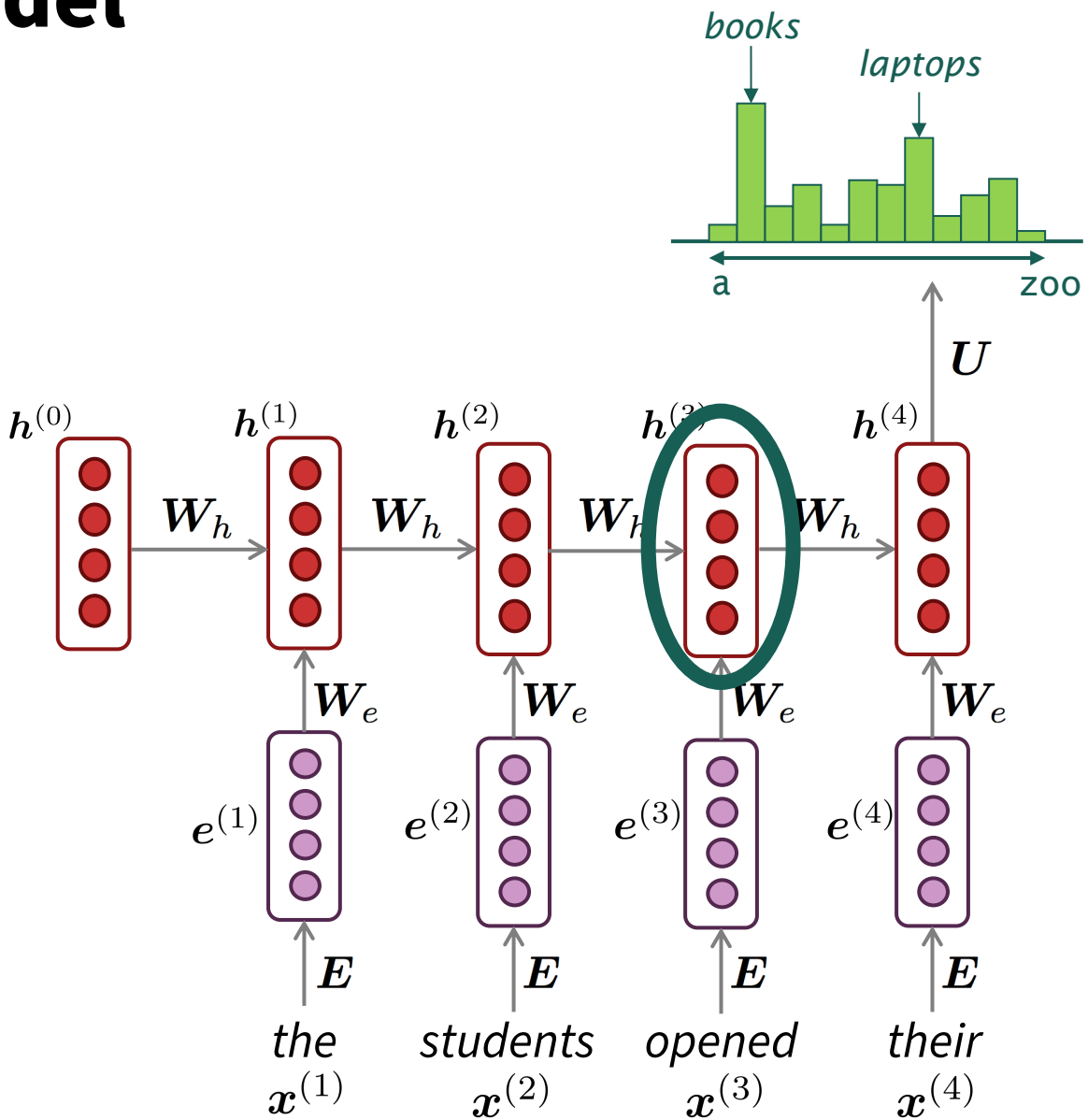
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

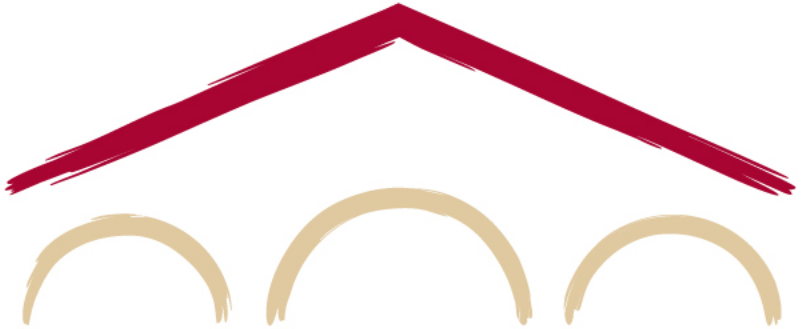
$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



RNNs can generate bounded hierarchical languages with optimal memory



John Hewitt

Michael Hahn

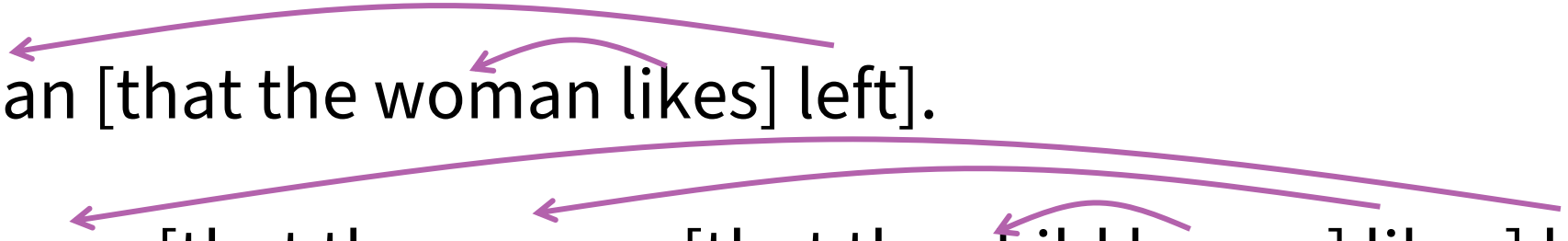
Surya Ganguli

Percy Liang

Chris Manning

Hewitt et al. 2020. RNNs can generate bounded hierarchical languages with optimal memory. EMNLP 2020.

Human languages have a hierarchical or recursive structure with nested dependencies

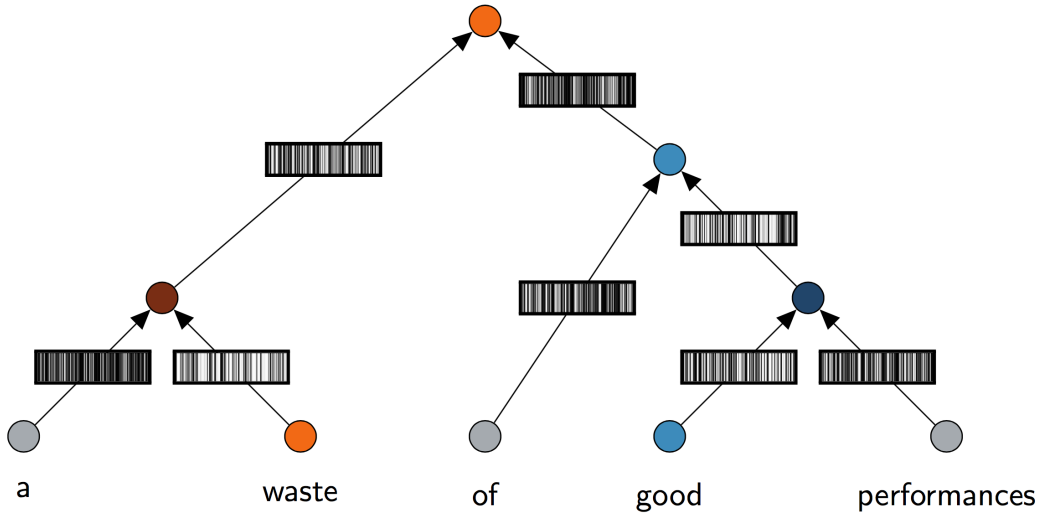


- [The man [that the woman likes] left].
- ???[The man [that the woman [that the child knows] likes] left].
- [The only thing [that the words [that can lose -d] have in common] is, apparently, that they are all quite common words].
  - E.G. Pulleyblank, *Journal of Chinese Linguistics* 10:410 (1982)
- [The odds [that your theory will be in fact right, and that the general thing [that everybody's working on] will be wrong,] are low].
  - Richard Feynman, Nobel Prize address, Stockholm Dec. 11, 1965 quoted from J. Gleick, *Genius*, Abacus Books, 1994, p. 382)

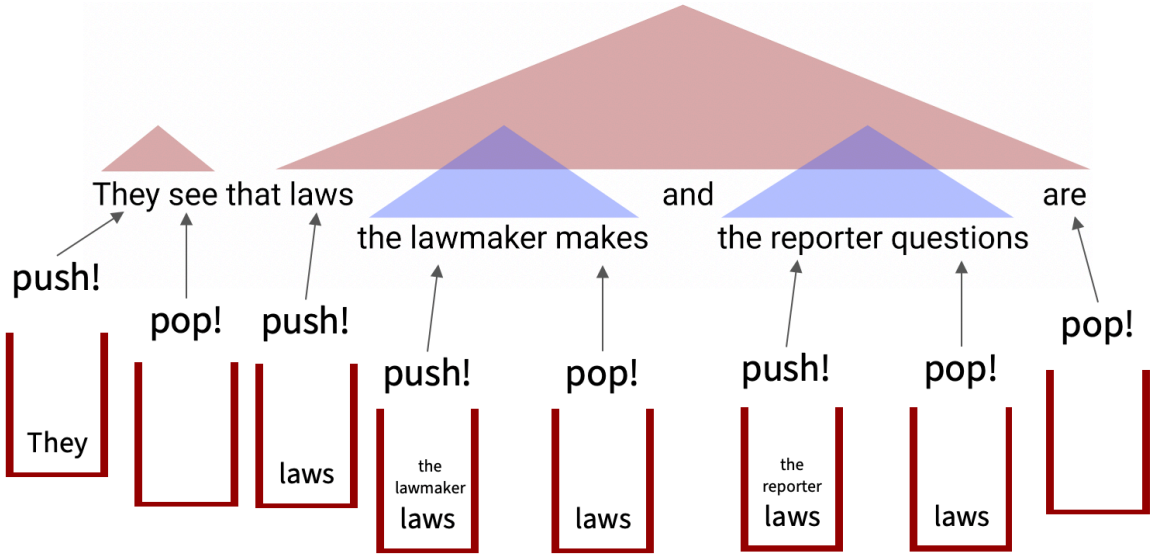
Examples from Geoffrey Sampson

Neural models of human language syntax

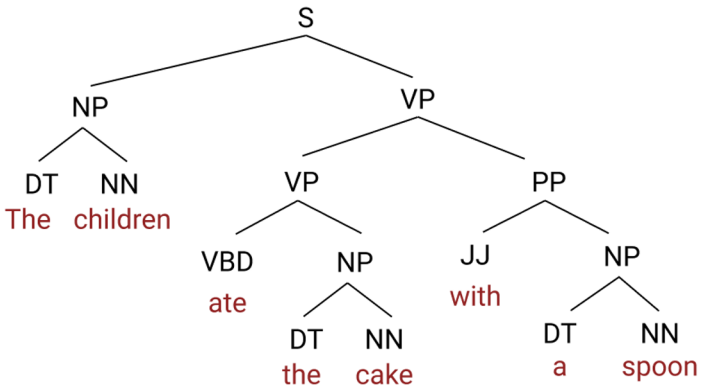
Tree Recursive Neural Networks (Socher et al. 2010ff), e.g., TreeLSTMs (Tai et al. 2015)

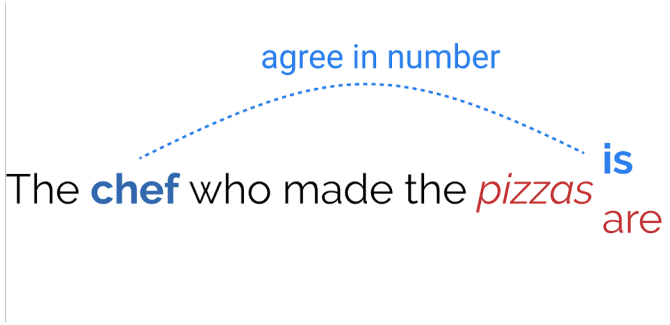


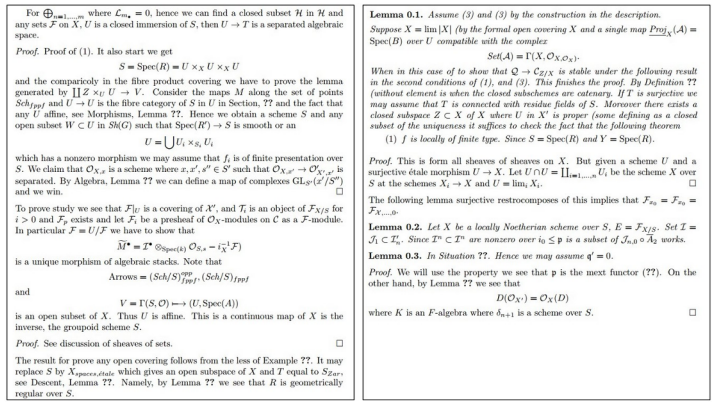
Stack-augmented RNNs (Sun et al. 1995, Joulin and Mikolov 2015, Grefenstette et al. 2015, Bowman et al. 2016, DuSell and Chiang 2020)



Recurrent neural networks and human language syntax







Human languages like English exhibit hierarchical or recursive structure, often with nested dependencies

RNN sequence models perform well on syntax-dependent language model predictions like agreement [1]

RNNs' generations can model hierarchical syntax, like this LaTeX from *The Unreasonable Effectiveness of RNNs*[2]

[1]Kuncoro et al., 2018, [2]Karpathy, 2015

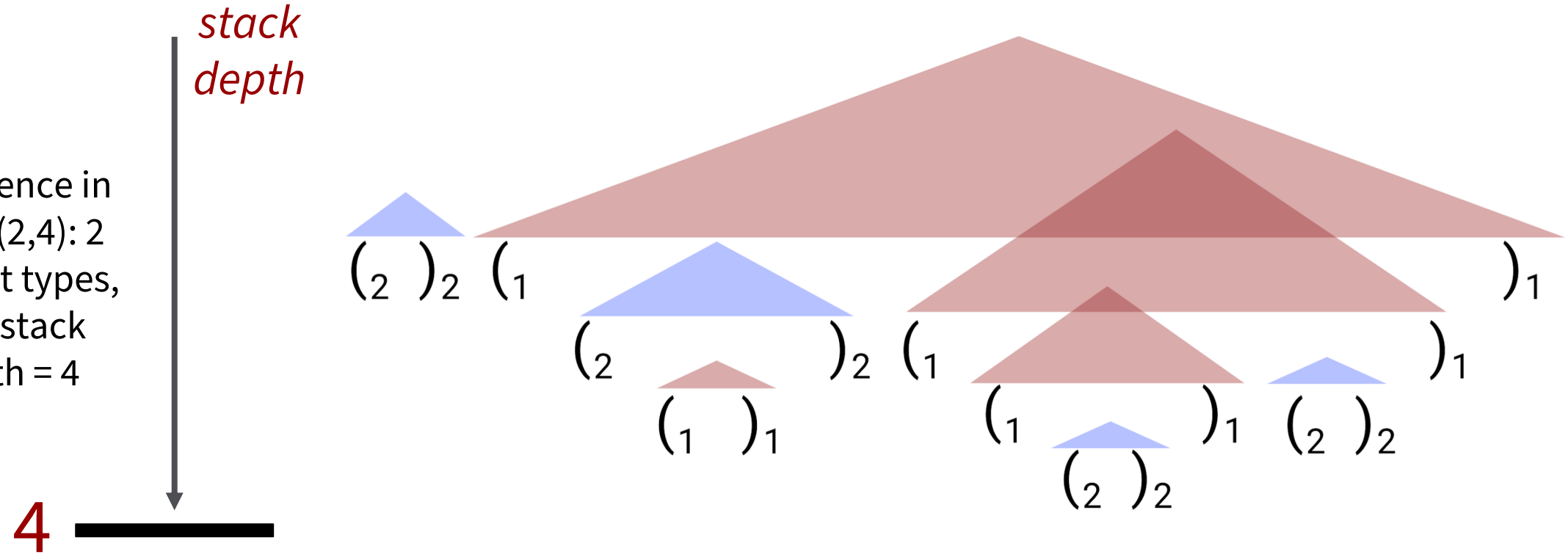
What are the results on the expressivity of RNNs?

	Precision	Running time	Expressivity result
[Siegelmann and Sontag, 1992]	unbounded	unbounded	Turing-Complete! ("RNNs can do anything!")
[Weiss et al., 2018]	logarithmic	once-per-token	RNNs can't count, but LSTMs can
[Merrill et al., 2018, 2020]	logarithmic	once-per-token	RNNs and LSTMs can't recognize stack-requiring languages
Our work	finite	once-per-token	???

Humans don’t have infinite-size stacks – Can RNNs handle *bounded* hierarchy?

We introduce Dyck-(*k*,*m*): a language of hierarchical dependencies like Dyck-*k*, the language of nested parentheses, but with at most *m* brackets in the stack at any time

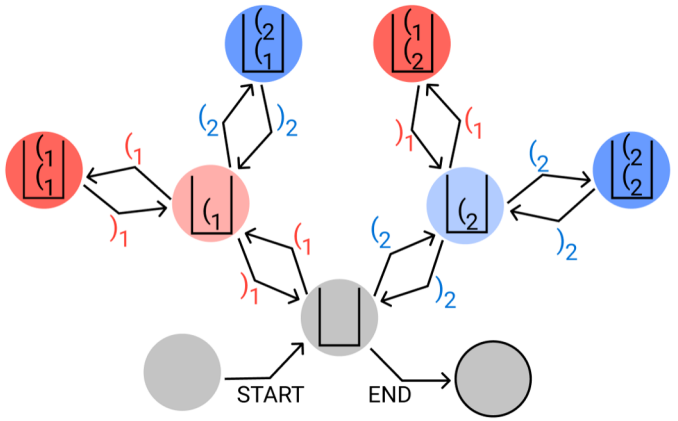
A sentence in Dyck-(2,4): 2 bracket types, max stack depth = 4



RNNs can obviously generate Dyck-(k,m) …

Because Dyck-(k,m) requires finite memory
it's a regular language

It is recognizable by a deterministic finite automaton with k^m states



RNNs can simulate *any* deterministic finite automaton, using roughly as many hidden units as states

So RNNs can generate Dyck-(k,m), but might need exponential memory, i.e., O(k^m)

Results

$k = 100{,}000$ *(vocab size)*
$m = 3$ *(stack depth)*

Previously known: Use around $O(k^m)$ hidden units to generate Dyck-(k,m) → $k^m = 100{,}000^3 = \mathbf{10^{15}}$ hidden units

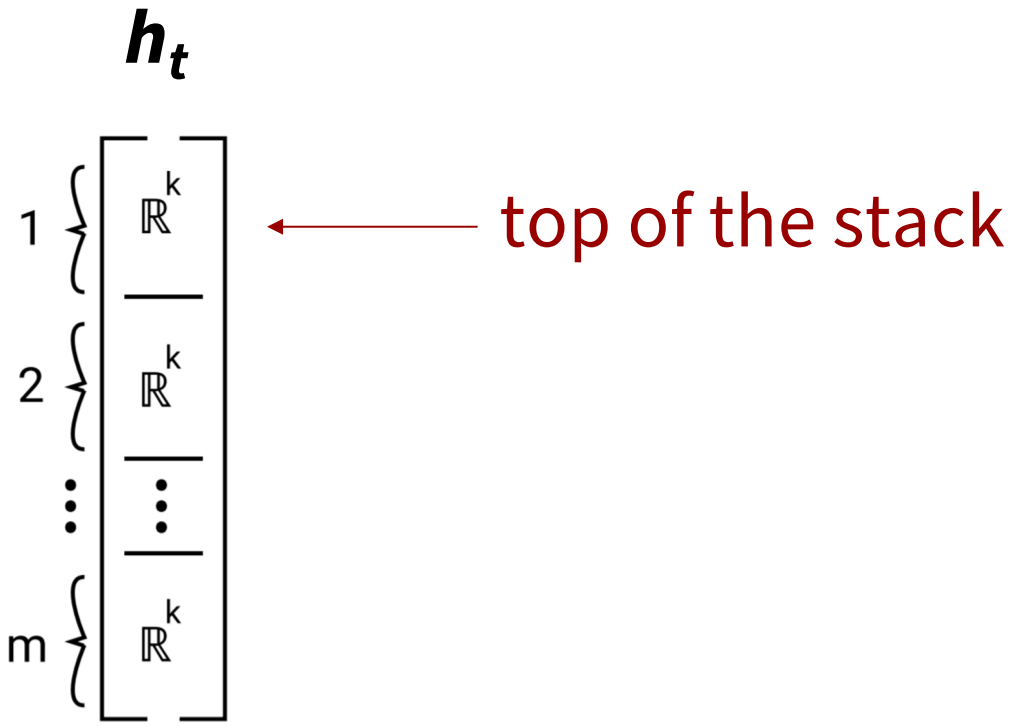
We prove: RNNs need at most $6m(\log k) - 2m = O(m \log k)$ hidden units to generate Dyck-(k,m) → $6m(\log k) - 2m < \mathbf{300}$ hidden units

We prove *this is tight:* it is impossible to use asymptotically fewer units

A vector as a bounded stack

RNNs have vector memories, $\boldsymbol{h_t}$

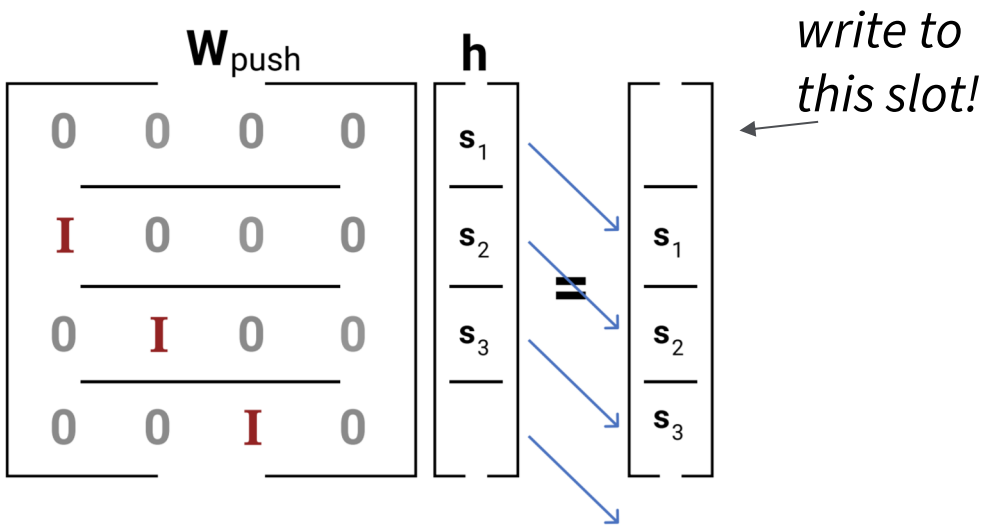
Think of $\boldsymbol{h_t}$ as having km dimensions, split into m stack slots, each of dimension k



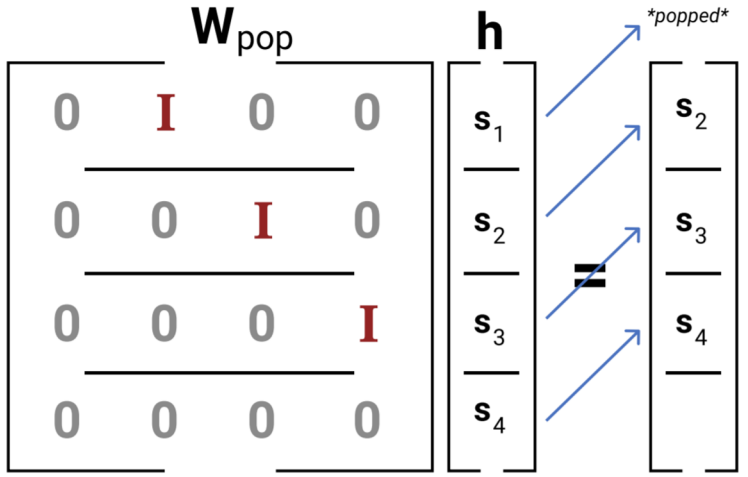
Encode each of the k types of open brackets as a **1-hot vector**

Stack simulation in an (extended) RNN

What if RNNs could push to the vector stack like this:



And RNNs could pop from the vector stack like this:

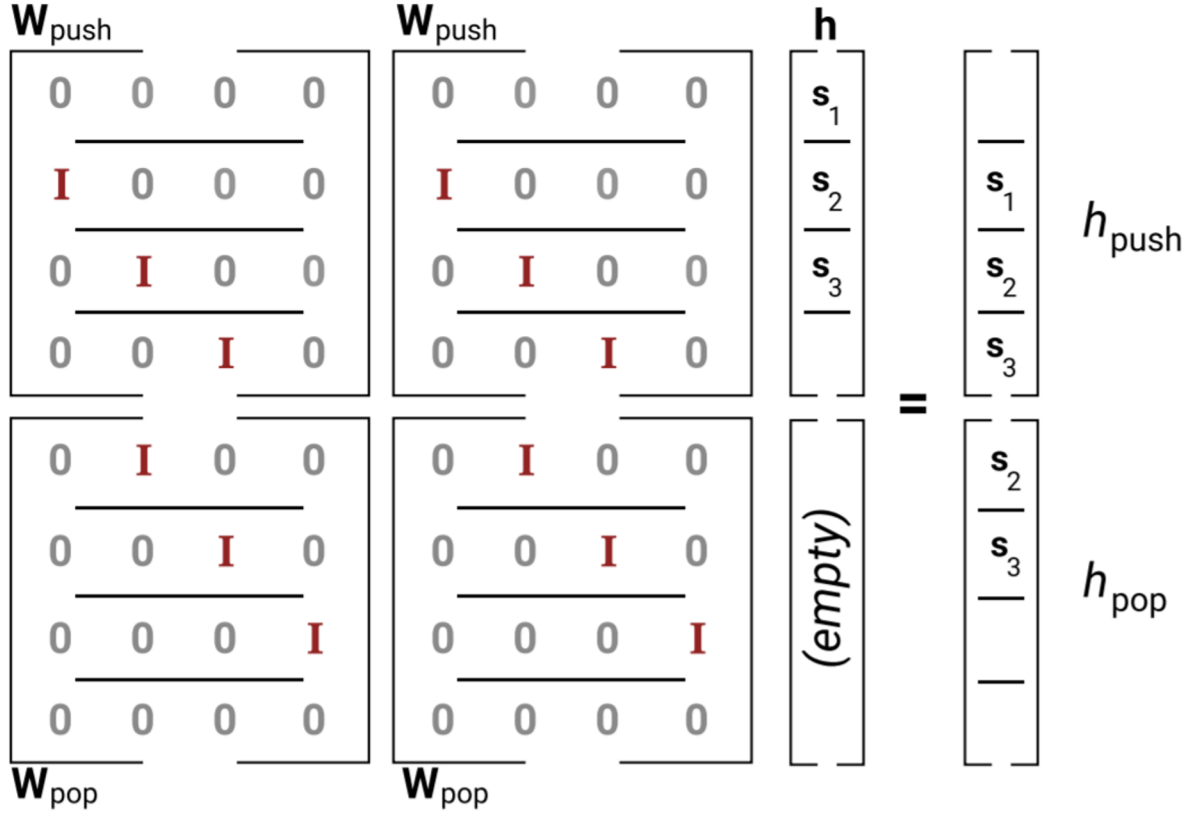


Recall the RNN equation: $h_t = \sigma(Wh_{t-1} + Ux_t + b)$

RNNs can't implement this solution because they only have 1 recurrent matrix

Stack simulation in a simple RNN

- Keep two copies of the stack
- Always keep one copy empty
- Always read from both
- **Always write to both**



Stack simulation in a simple RNN

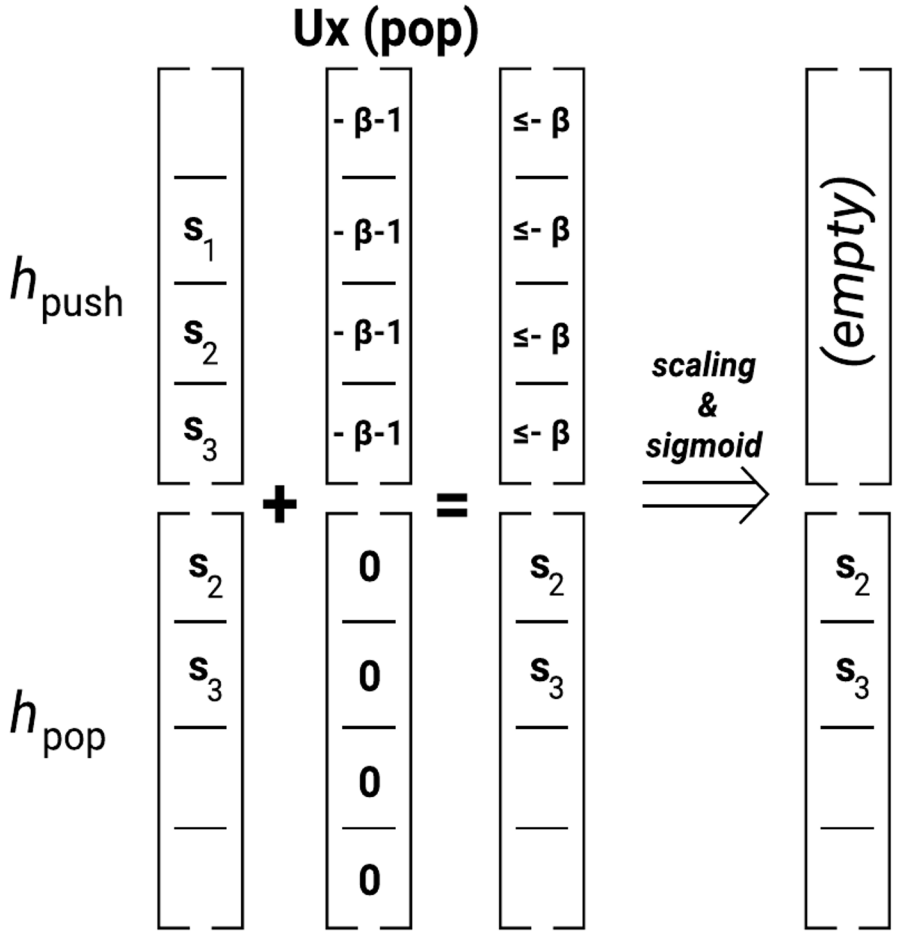
$h_t = \sigma(Wh_{t-1} + Ux_t + b)$

- Keep two copies of the stack

- Always keep one copy empty

- Always read from both

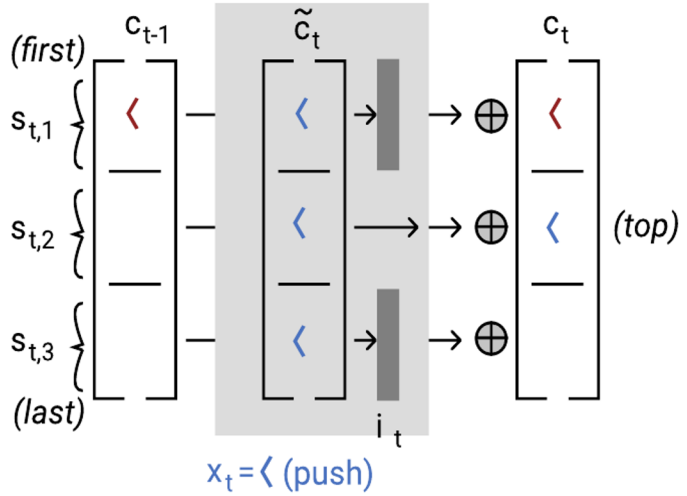
- Always write to both, but use the nonlinearity to zero out one copy or the other depending on whether we pushed or popped

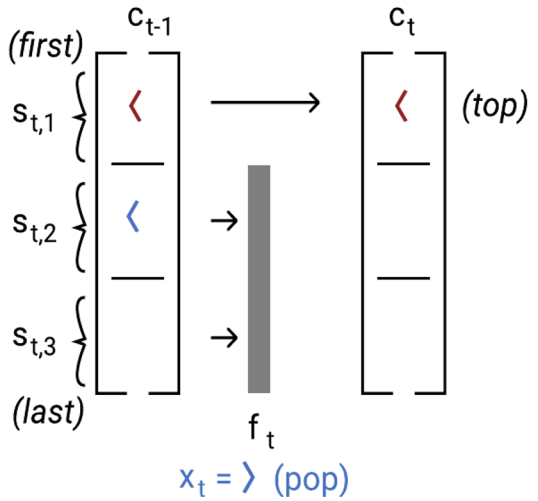


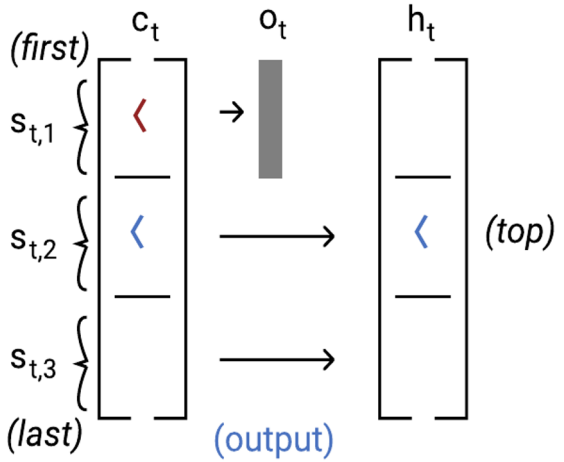
(Imagine β is a very large number)

Stack simulation in an LSTM using *only* gates

LSTMs can handle bounded hierarchy using just their gates (i.e., $W = \mathbf{0}$)







The input gate helps determine where to **push a bracket**

The forget gate helps determine where to **pop a bracket**

The output gate helps determine where the **top of the stack** is

(Every bracket tries to write itself to every slot!)

An efficient vocabulary encoding

To improve from **O(*mk*)** to **O(*m* log *k*)**, we need an **O(log *k*) stack slot**

Idea One: give each of the k brackets one of the $2^{\log k} = k$ bitstrings of length $\log k$

Encoding of bracket **45**

00101101

Idea Two: ensure the bracket identity encoded by a stack slot can be read out linearly by concatenating the binary negation

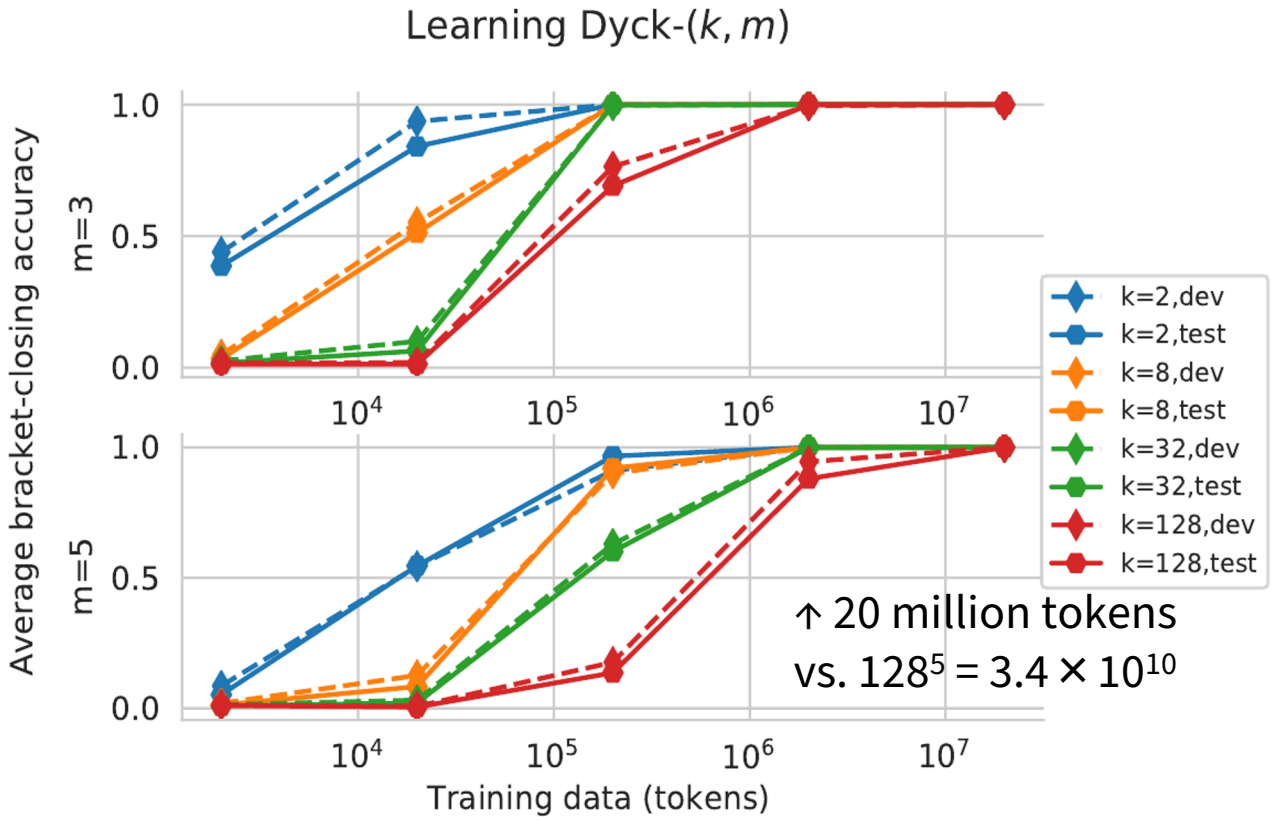
Encoding of bracket **45**

00101101 11010010

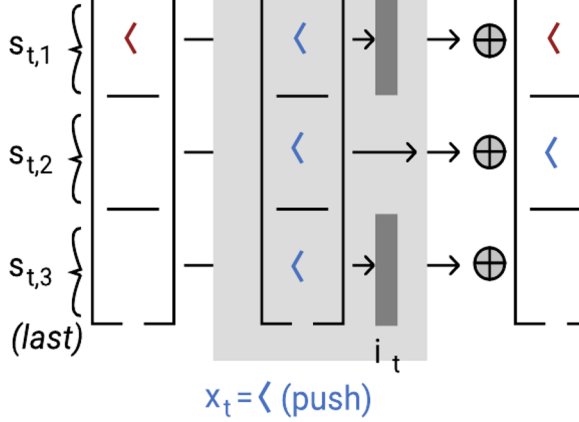
Experiments: Dyck-(*k*,*m*) is learned in practice

Dyck-*k*, with its unbounded memory requirements, isn't learned by RNNs in practice (Suzgun et al., 2019)

Dyck-(*k*,*m*) is learnable by an RNN of the size our theory suggests, even when only a small percentage of stack configurations are seen at training time, suggesting stack-like behavior

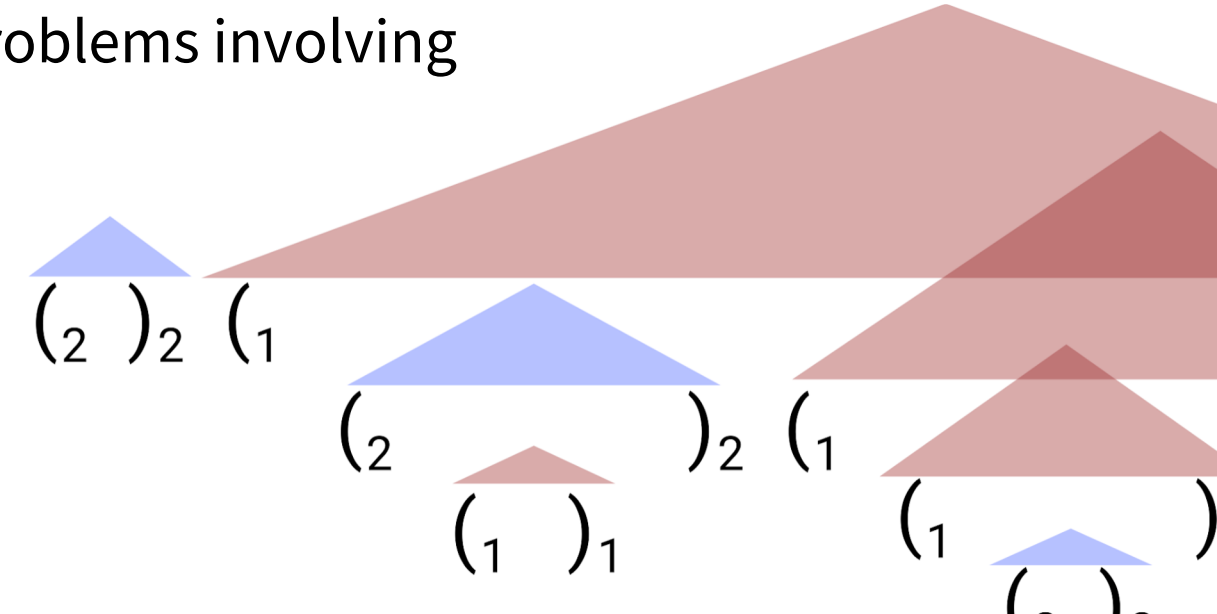


Takeaway



The Chomsky Hierarchy places all finite-memory languages in the same class, but most neural networks we use have finite memory (as do humans!), so we need finer-grained theoretical tools, like memory efficiency

We give constructive proofs, which we hope provide intuition and inspiration for how RNNs can represent problems involving bounded hierarchy



2018 NLP breakthrough with big language models

All these models are Transformer models

ELMo,
ULMfit
Jan 2018
Training:
103M words
1 GPU day

GPT
June 2018
Training
800M words
240 GPU days

BERT
Oct 2018
Training
3.3B words
256 TPU days
~320–560 GPU days

GPT-2
Feb 2019
Training
40B words
~2048 TPU v3 days according to a reddit thread

XL-Net, ERNIE,
Grover, AlBERT,
Megatron-LM, T5,
RoBERTa, GPT-3
July 2019–

GPT-2 language model (cherry-picked) output

SYSTEM PROMPT (HUMAN-WRITTEN)

MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”

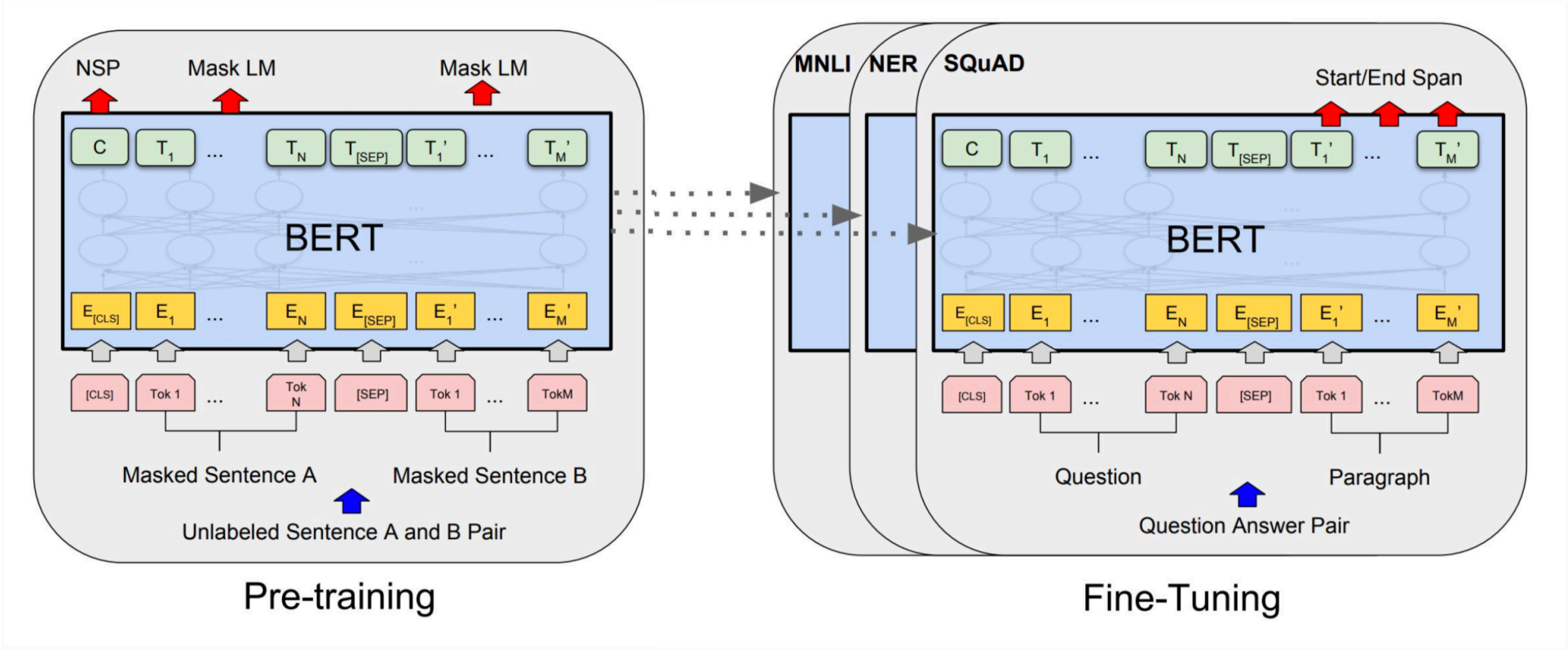
The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.

BERT model

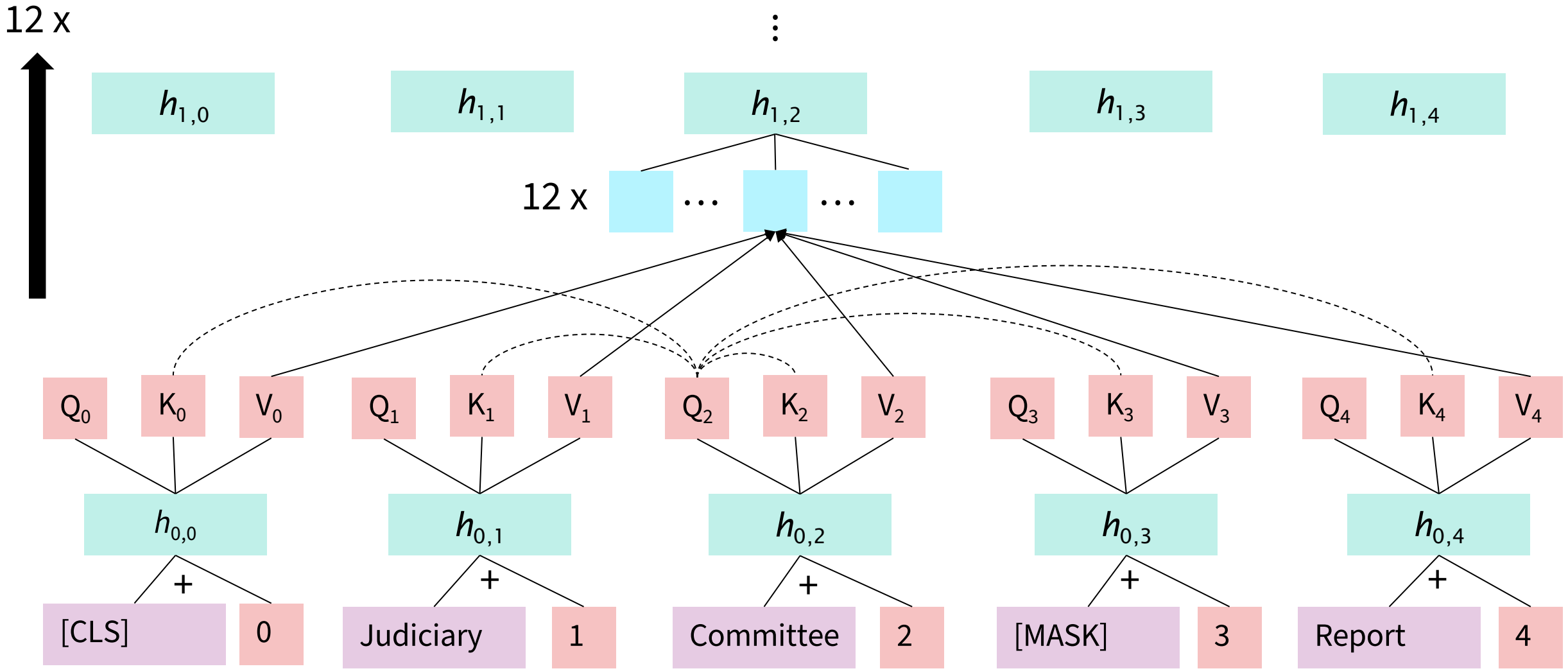
(Devlin, Chang, Lee, & Toutanova 2018)

Pre-train contextual word vectors in a LM-like way with transformers

Learn a classifier built on the top layer for each task that you fine tune for



Transformer (Vaswani et al. 2017)
BERT (Devlin et al. 2018) + descendants



SQuAD Question Answering leaderboard 2017-02-07

Passage

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California.

Question: Which team won Super Bowl 50?

System	F1
Human performance	91.2
r-net (MSR Asia) [Wang et al., ACL 2017]	79.7
DrQA (Chen et al. 2017)	79.4
Multi-Perspective Matching (IBM)	78.7
BiDAF (UW & Allen Institute)	77.3
Fine-Grained Gating (Carnegie Mellon U)	73.3
Logistic regression	51.0

SQuAD 2.0 Question Answering leaderboard 2019-02-07

Passage

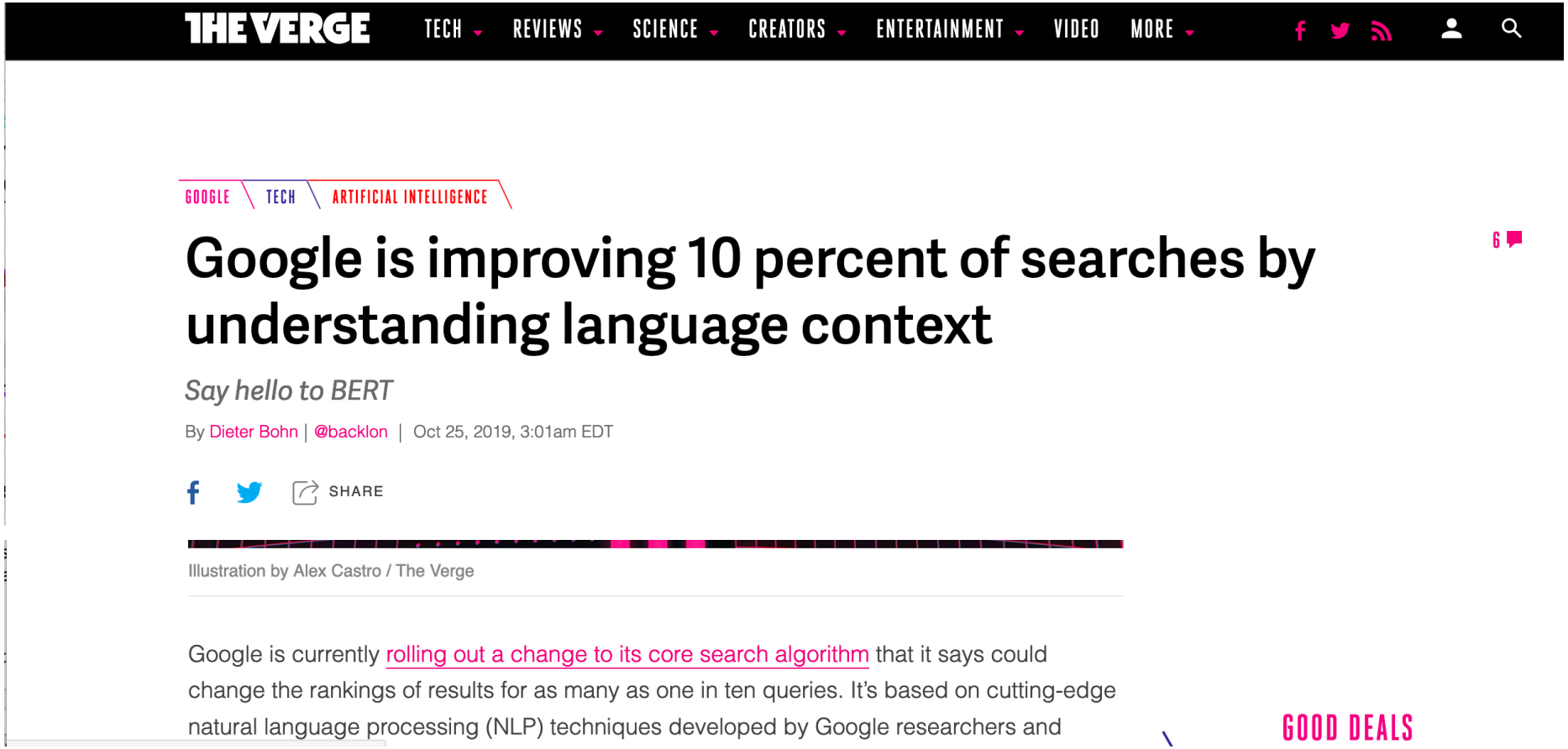
Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California.

Question: Which team won Super Bowl 50?

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Jan 15, 2019	BERT + MMFT + ADA (ensemble) *Microsoft Research Asia*	**85.082**	**87.615**
2 Jan 10, 2019	BERT + Synthetic Self-Training (ensemble) *Google AI Language* https://github.com/google-research/bert	84.292	86.967
3 Dec 13, 2018	BERT finetune baseline (ensemble) *Anonymous*	83.536	86.096
4 Dec 16, 2018	Lunet + Verifier + BERT (ensemble) *Layer 6 AI NLP Team*	83.469	86.043
4 Dec 21, 2018	PAML+BERT (ensemble model) *PINGAN GammaLab*	83.457	86.122
5 Dec 15, 2018	Lunet + Verifier + BERT (single model) *Layer 6 AI NLP Team*	82.995	86.035

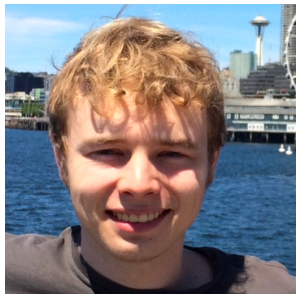
Google web search

BERT brings big gains to web search



Emergent linguistic structure in artificial neural networks trained by self-supervision


Chris Manning


Kevin Clark


John Hewitt

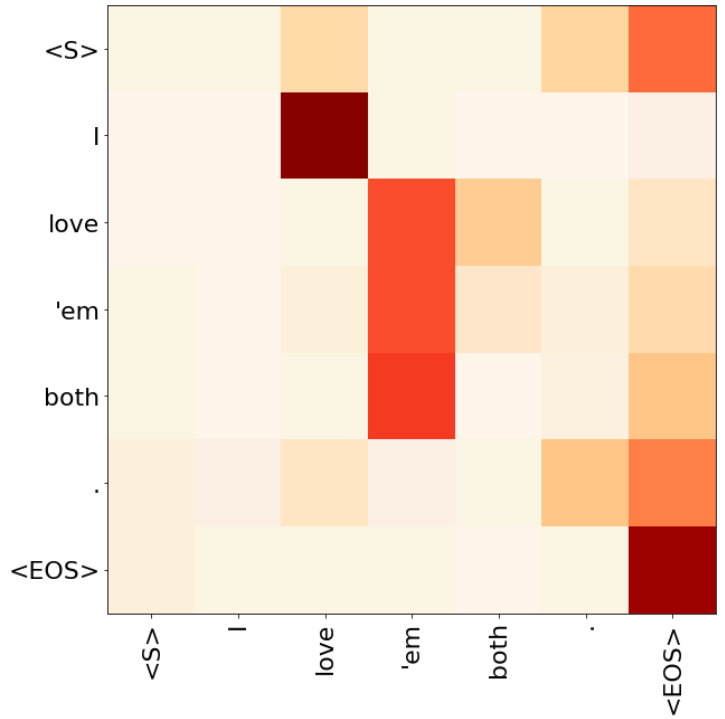

Urvashi Khandelwal


Omer Levy

Manning et al. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *PNAS.*

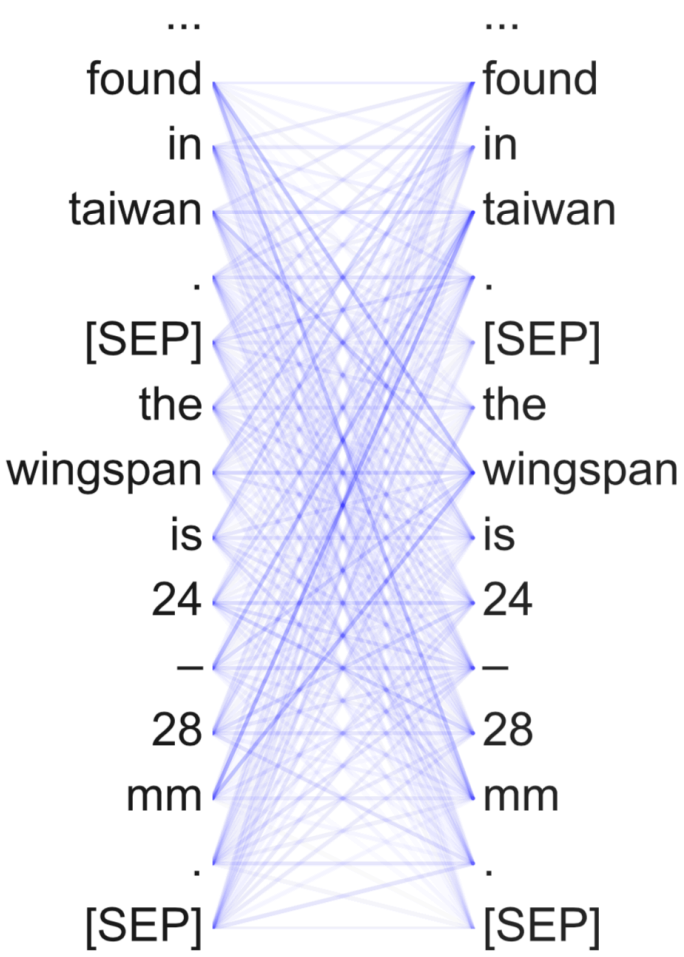
What does BERT know? Observational evidence

- BERT calculates useful context-dependent word representations
- For each of many attention heads, for each word position, we directly observe where BERT pays attention
- Look at the most-attended-to word for each head
- How does what BERT attends to correspond to postulated linguistic structure?
- We find that BERT induces a structure similar to conventional linguistic structure, because it helps predict word occurrence in context



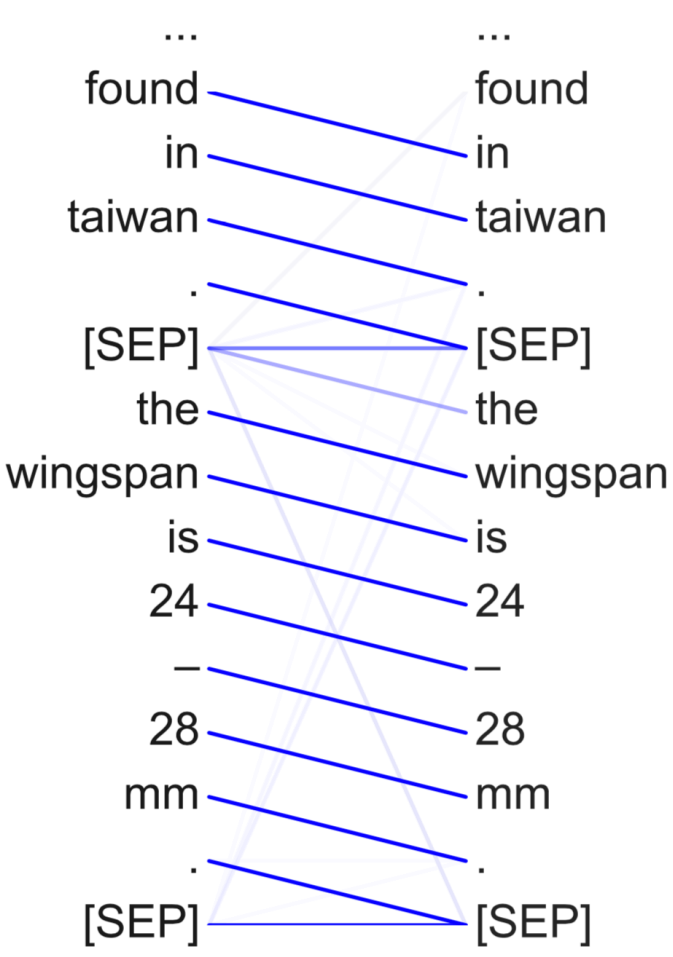
What do BERT attention heads do?

1-1: Attend broadly (“BoW head”)



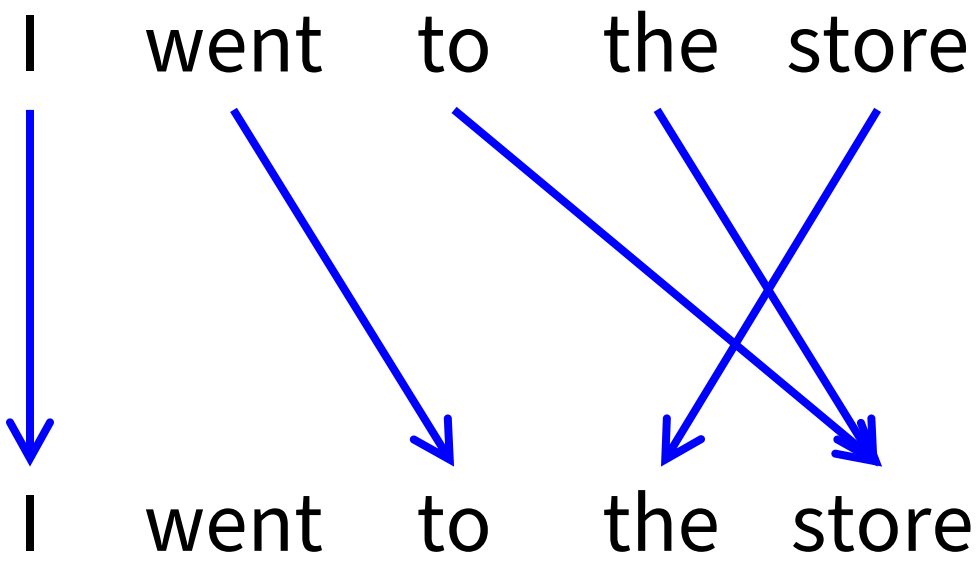
Word attention target

3-1: Attend to next (or prev) word

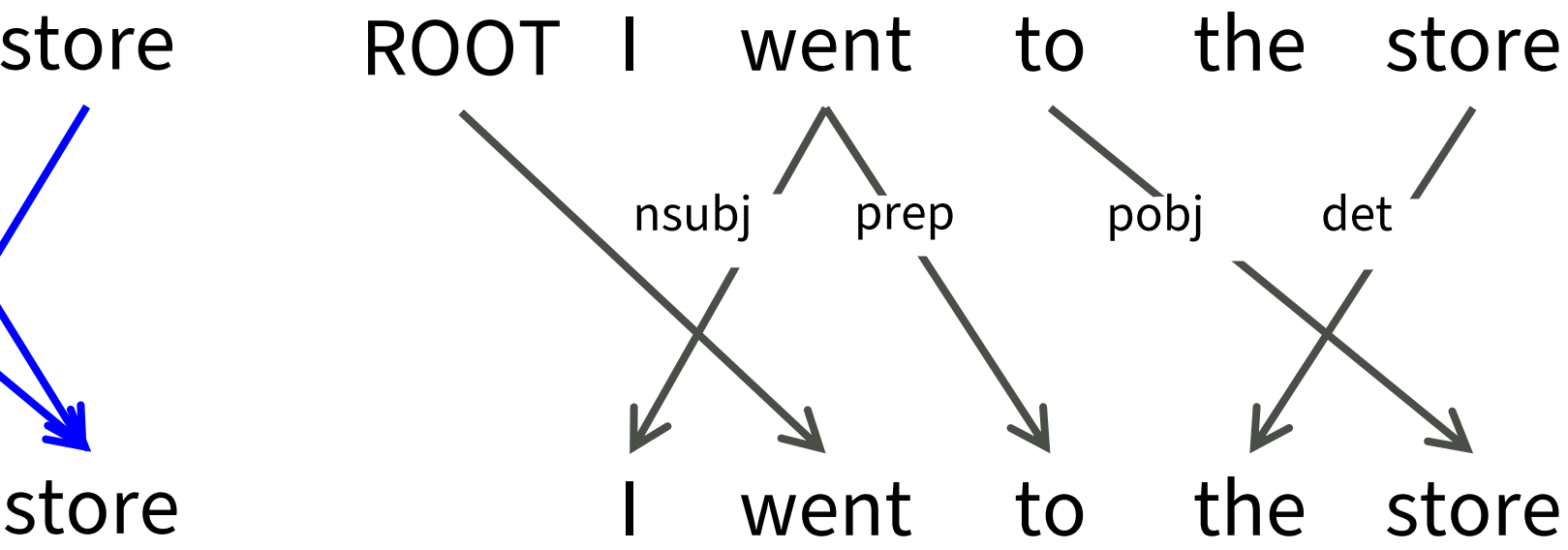


Word attention target

Does some of BERT attention resemble syntactic dependencies?

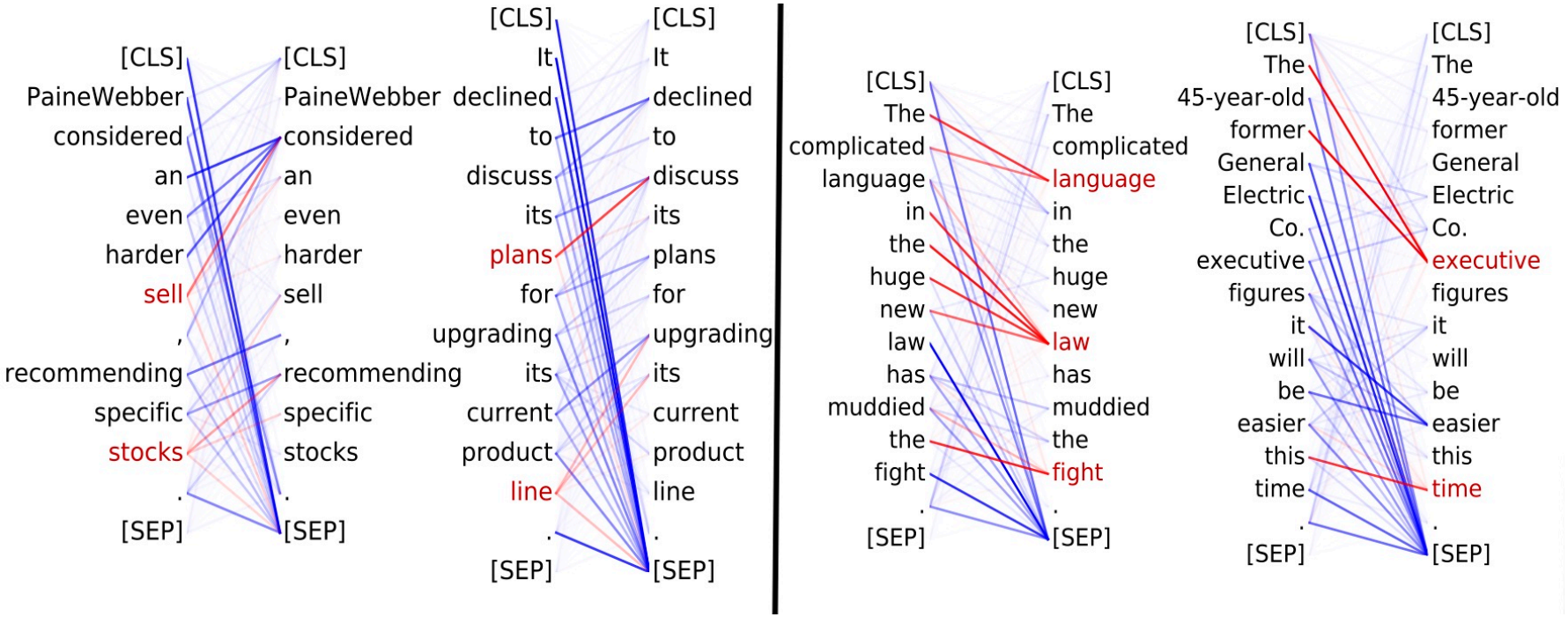


Take the most-attended-to words



Compare with dependency tree

A bunch of heads specialize on a syntactic relation (!)

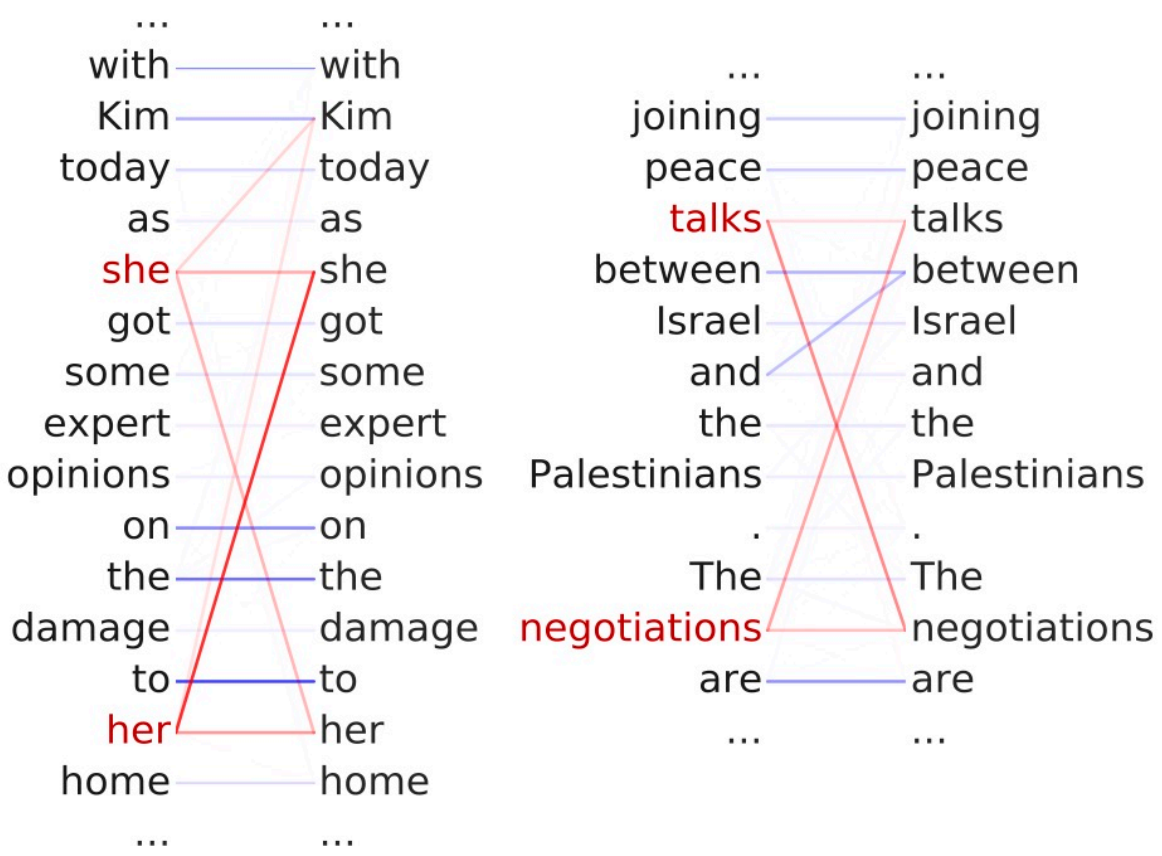


Head 8-10
Direct objects attend to verbs
86.8% on dobj relation

Head 8-11
Noun modifiers (det, adj) attend to head noun. 94.3% on det relation

Overall, a combination of these heads can give an okay dependency parser: 77 UAS (Cf. 26 from right branching, 58 from GloVe word vecs + distance)

There’s a coreference head (!)

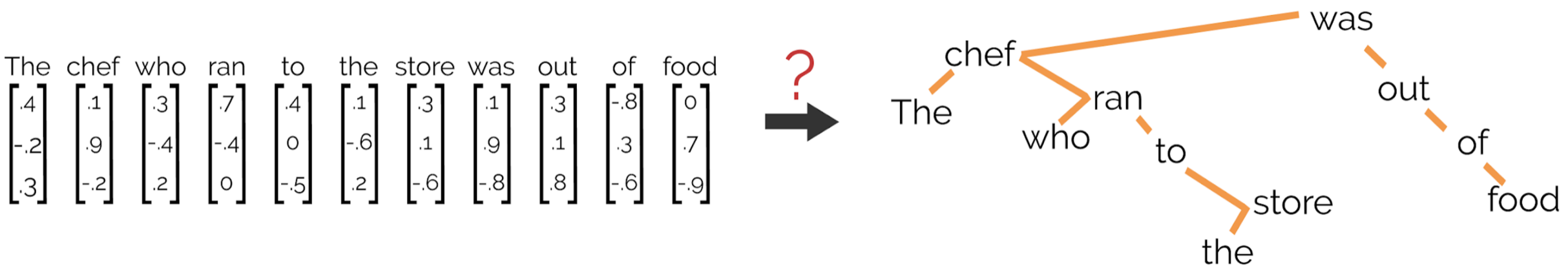


Coreferent mentions attend to their antecedent; for not a mention words: no-op attention 85% on [SEP]
Head 5-4: **65.1%** accuracy at linking to head of antecedent
Cf. vs. 69% for a 4-sieve, rule-based system (cf. Lee et al. 2011)
choosing nearest {full string, headword, PNG match; any NP}

A structural probe to connect vector spaces and trees

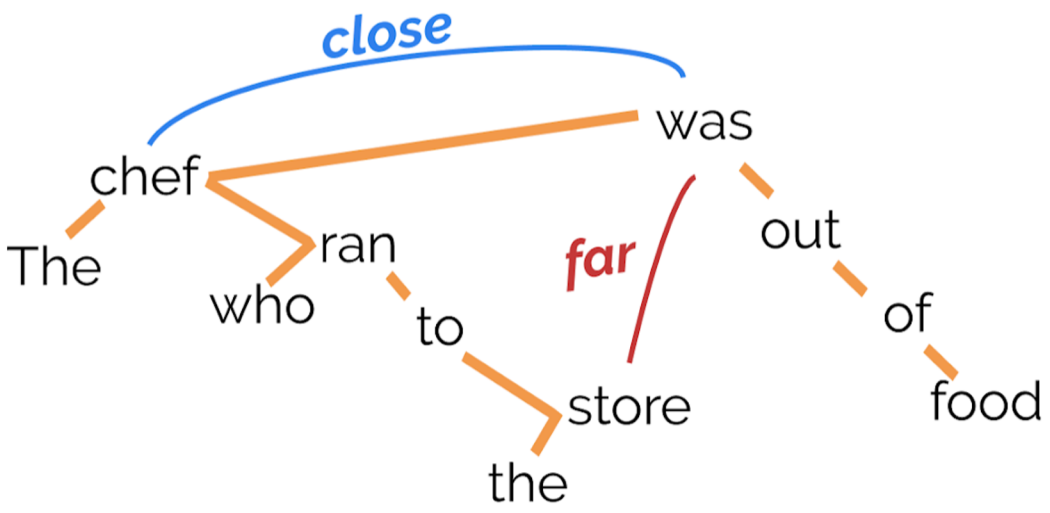
Are the vector space representations in NLP reconcilable with the discrete syntactic tree structures hypothesized for language?

- Here we are looking for dependency syntax structures



Distance metrics unify trees and vectors

An **undirected tree** defines a **distance metric** on pairs of words, the path metric: the number of edges in the path between the words.



The — chef $d_{path} = 1$

...

chef — ran $d_{path} = 1$

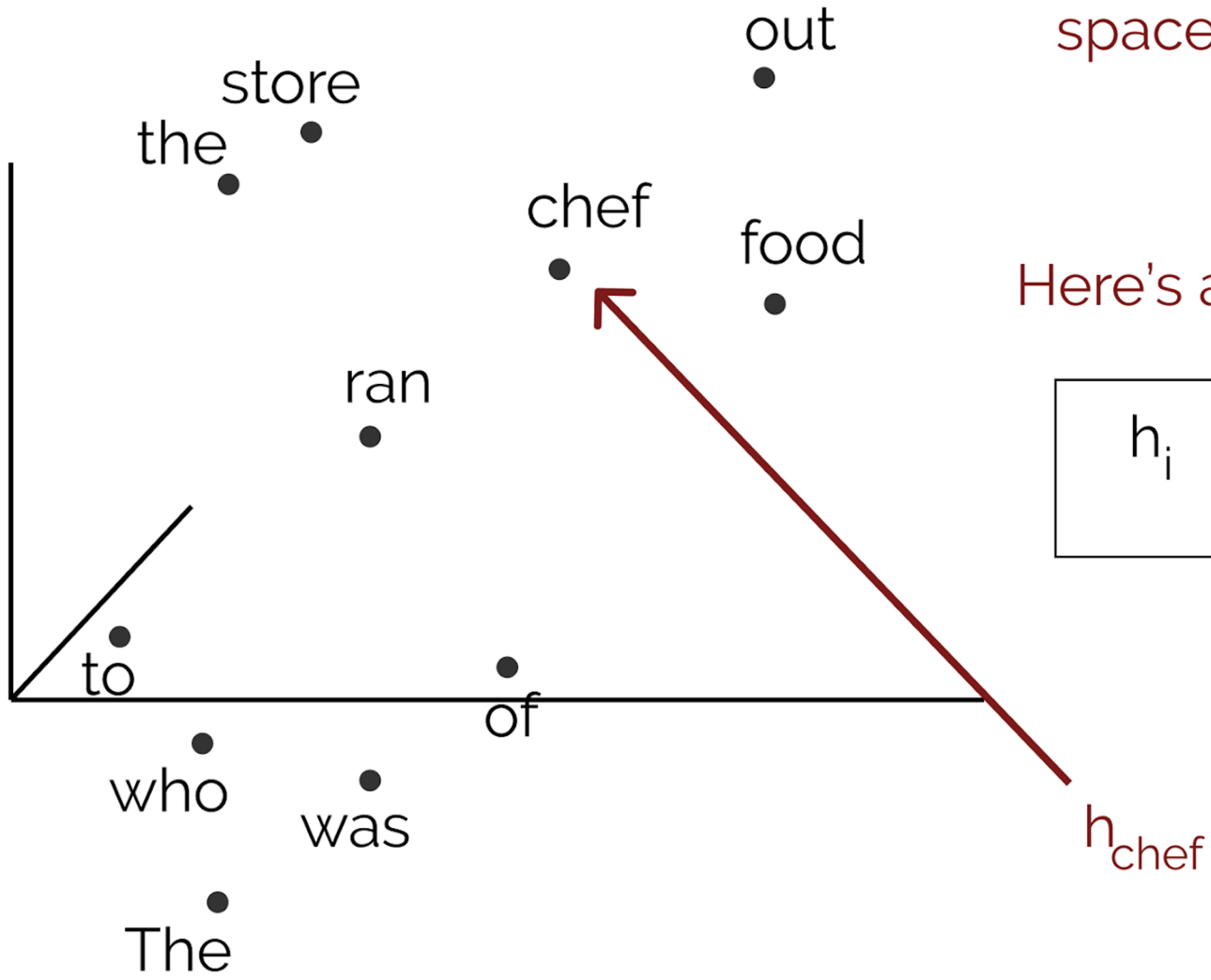
chef — was $d_{path} = 1$

...

was — — — — store $d_{path} = 4$

The edges of the tree can be recovered by looking at all distance=1 pairs.

Finding trees in vector spaces

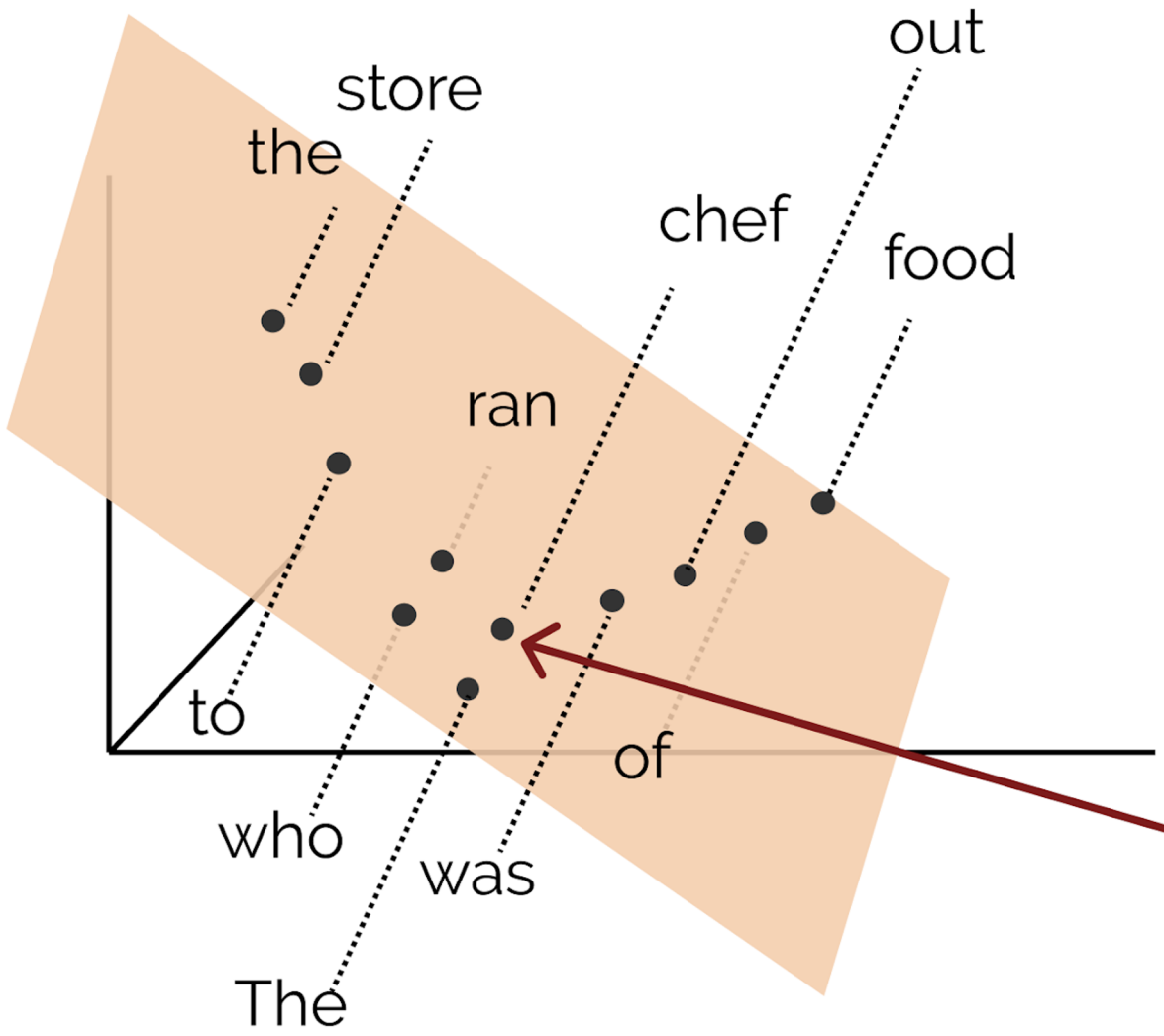


We can look for trees in the vector space by looking for their **distances** and **norms** in the space.

Here's a sentence embedded by a NN!

h_i h_j : vector representation of words *i* and *j*.

Finding trees in vector spaces



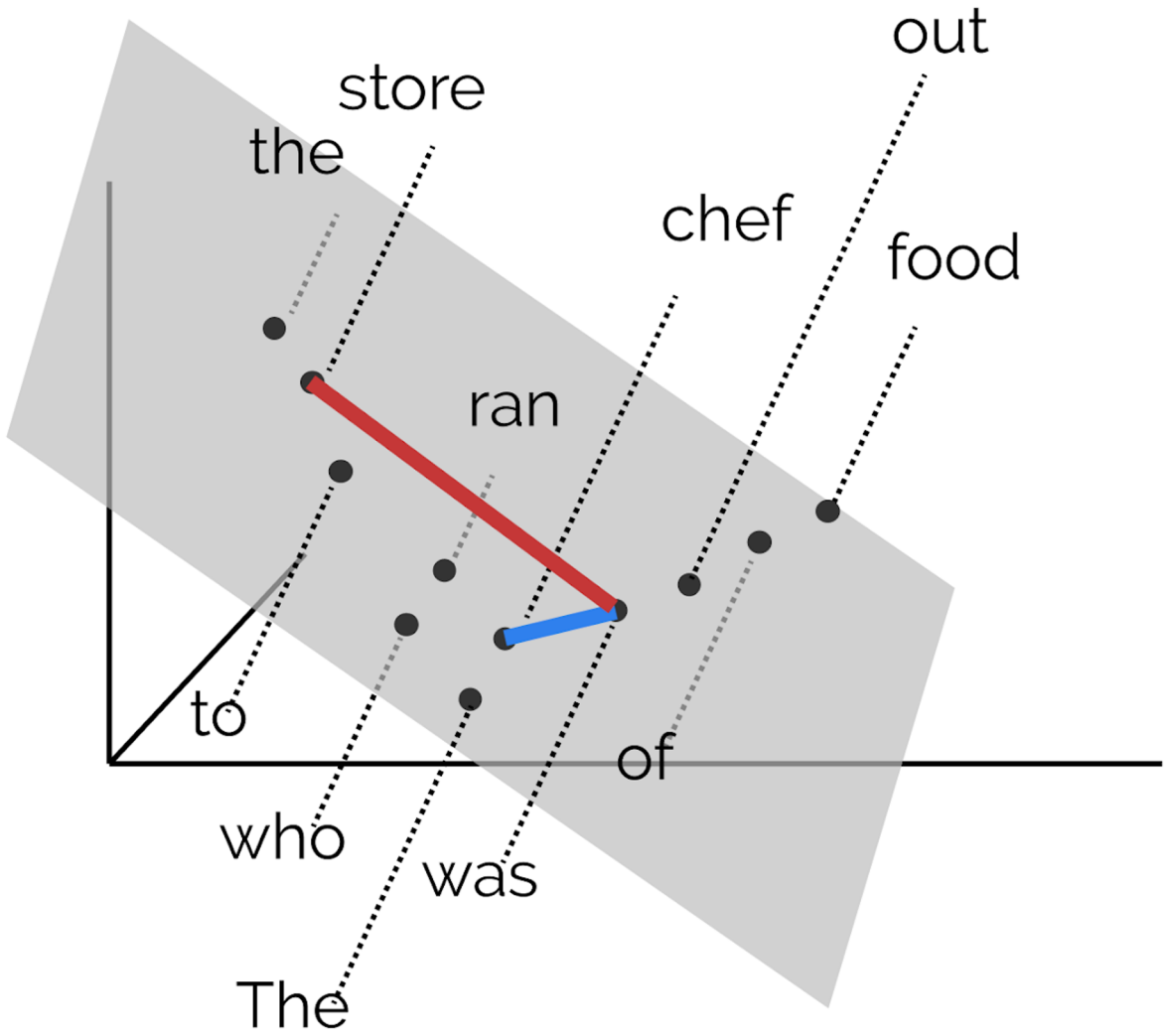
We don't expect all dimensions of the vector space to encode syntax -- NNs have a lot to encode!

We find the linear transformation that encodes syntax best.

B : The syntax transformation matrix

Bh_i : Syntax-transformed vector word representation

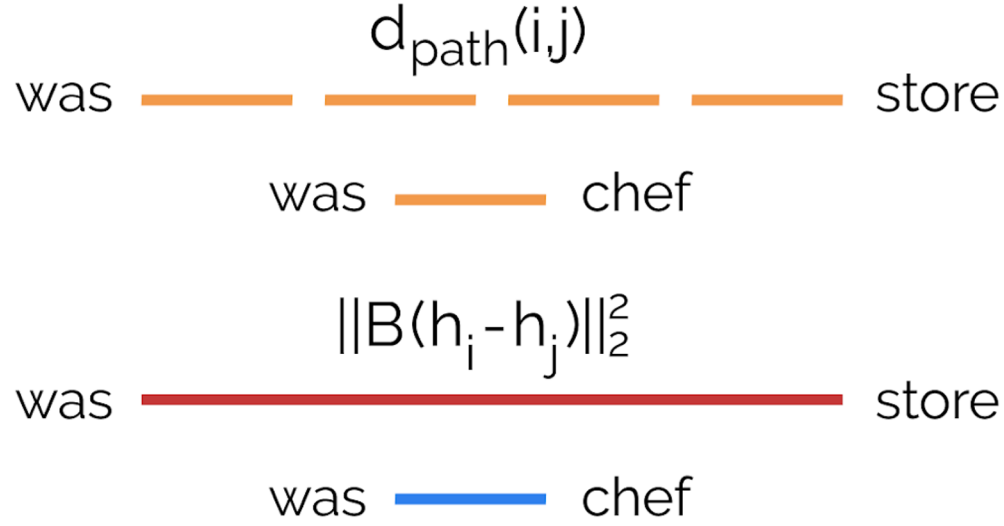
Finding trees in vector spaces



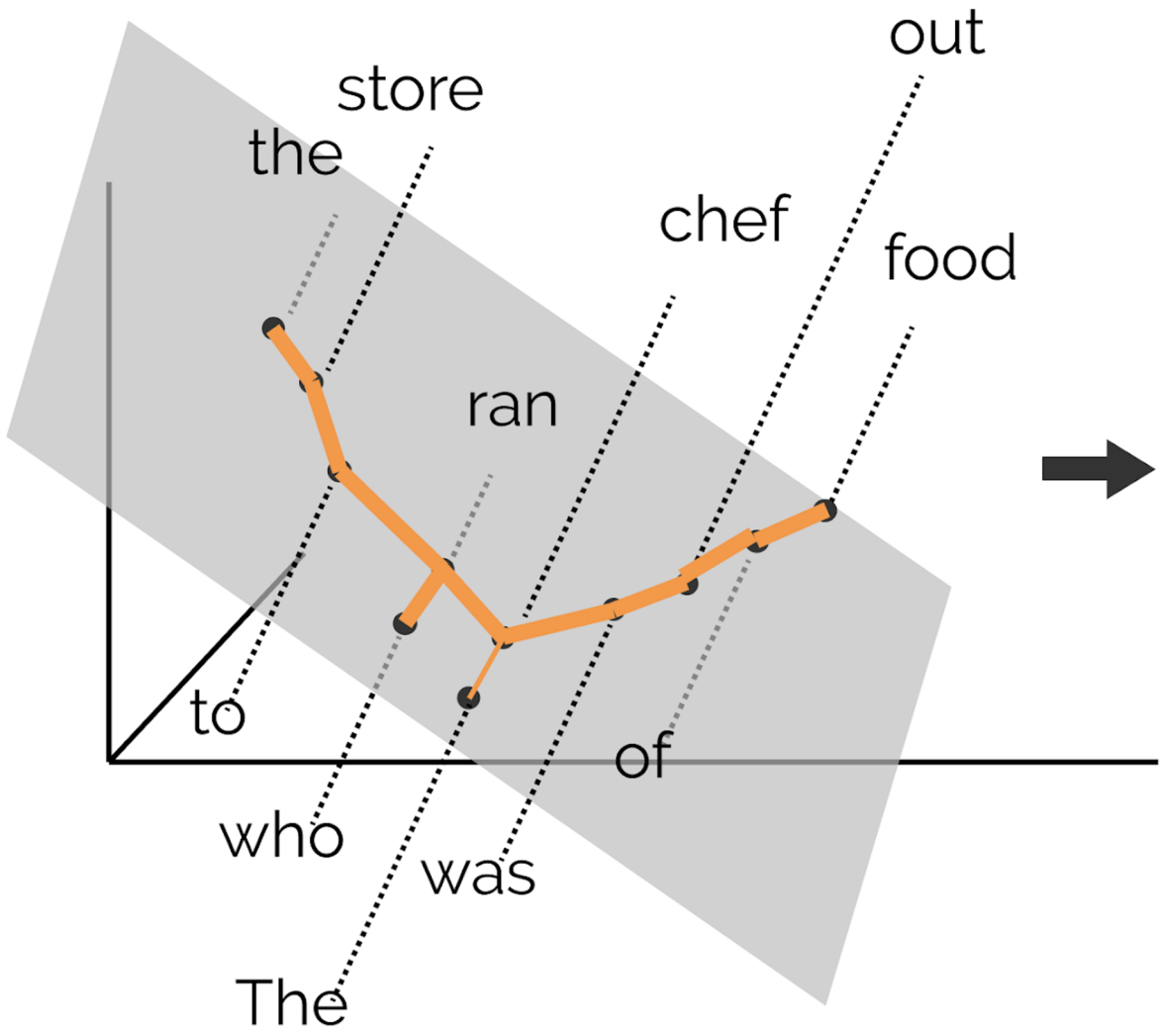
In the transformed space, (squared) L2 distance approximates tree distance.

$d_{path}(i,j)$: Tree path distance

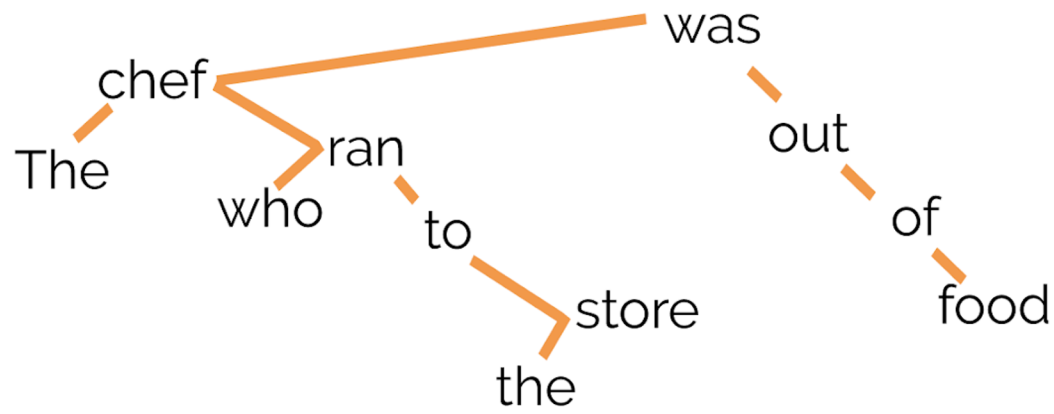
$||B(h_i - h_j)||_2^2$: Squared Vector space distance ($||h_i - h_j||_B^2$)



Finding trees in vector spaces



With this property, a minimum spanning tree in the vector space distance recovers the tree.



Does BERT encode undirected parse trees -> does there exist a *distance* transformation?

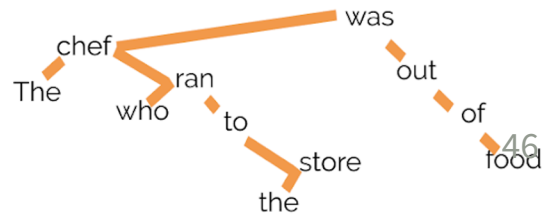
$$\arg\min_{B} \sum_{\ell \in \text{PTB}} \frac{1}{|s^\ell|^2} \sum_{i,j} \left| d^\ell_{\text{path}}(i,j) - \|B(h^\ell_i - h^\ell_j)\|_2^2 \right|$$

Find a single transformation *B*

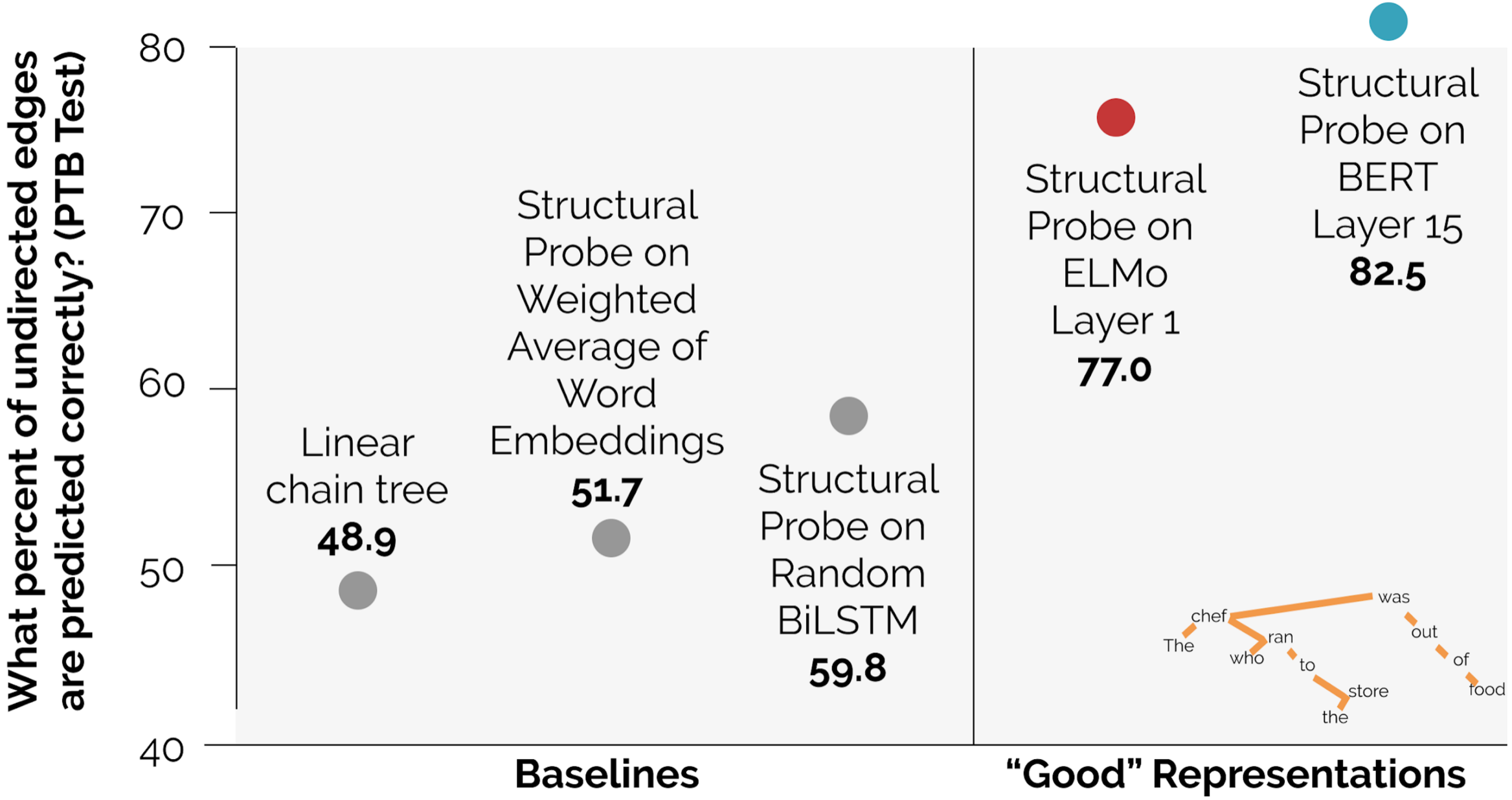
such that over all sentences in PTB training

Over all word pairs in each sentence

The difference between **tree distance** and **squared vector distance** is *minimized*

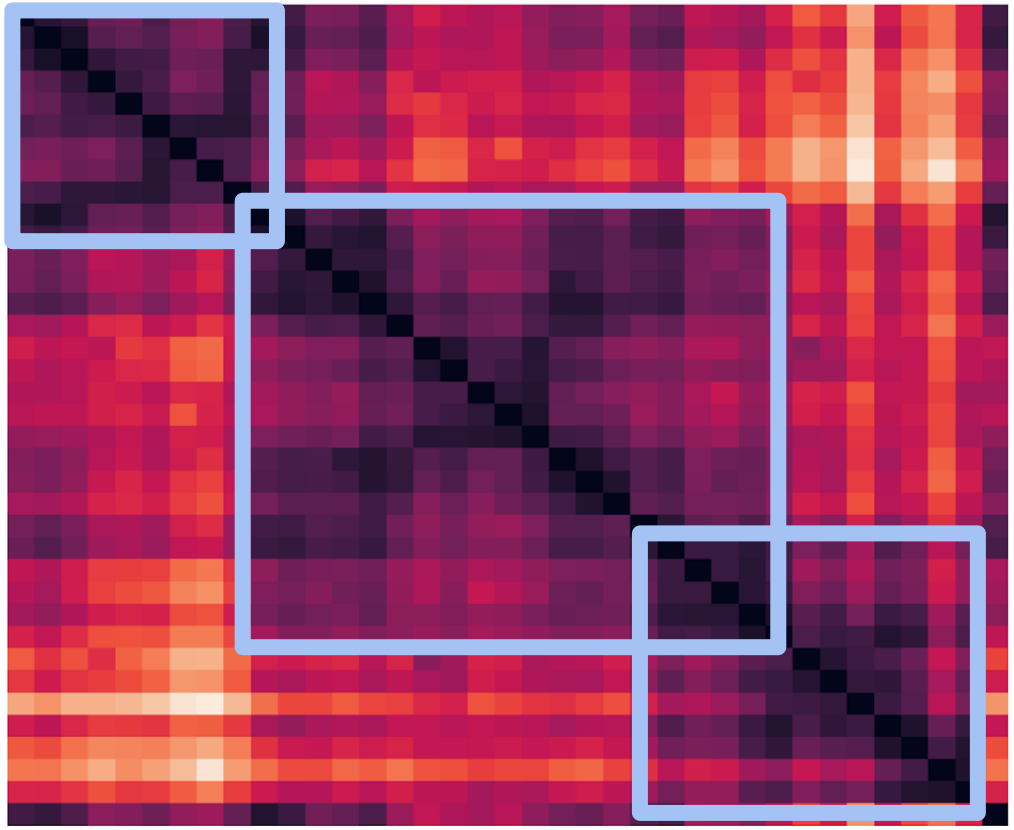


Trees are encoded well in these representations

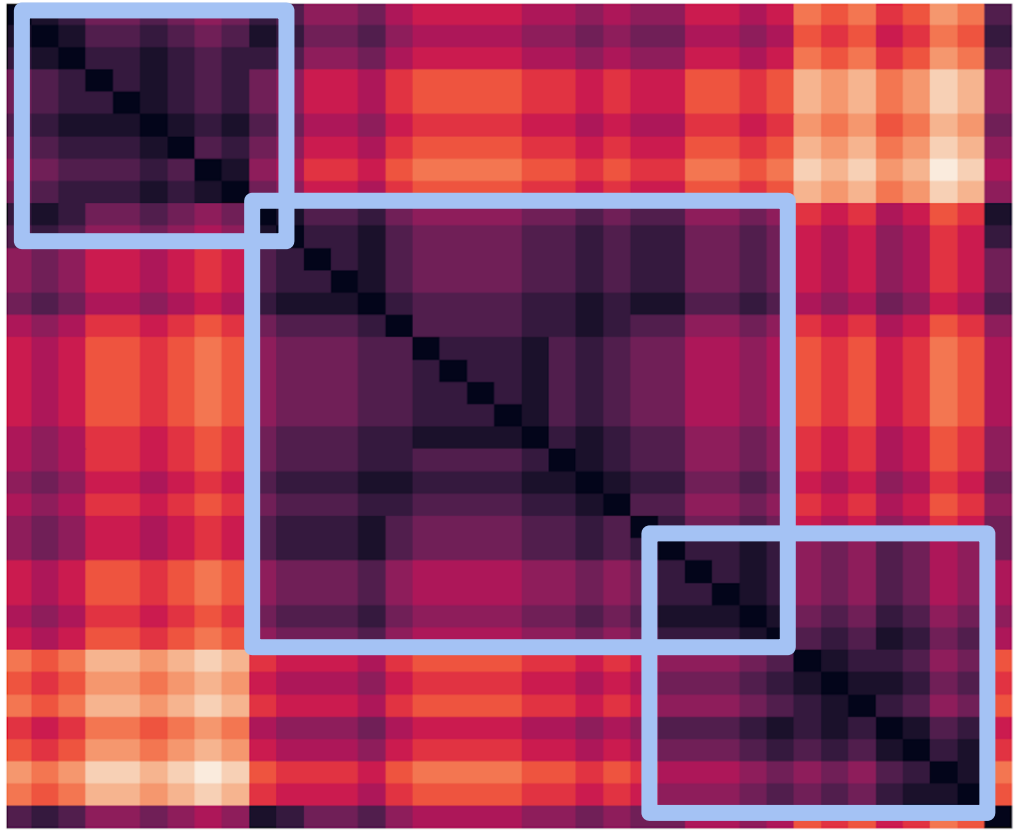


Legend: far close

BERT structural probe



Gold parse tree

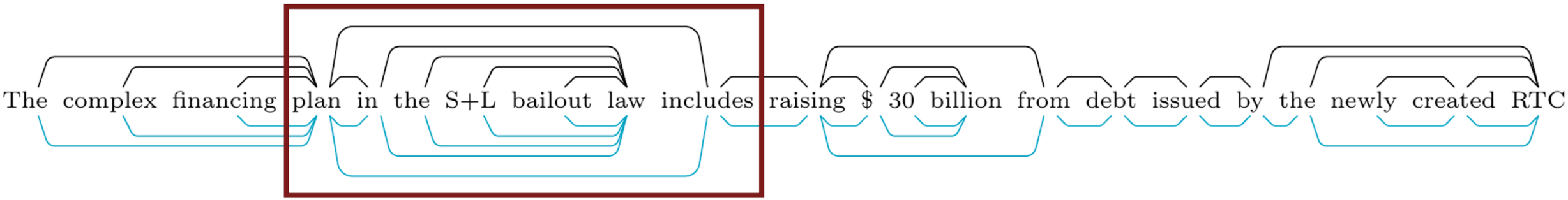


words

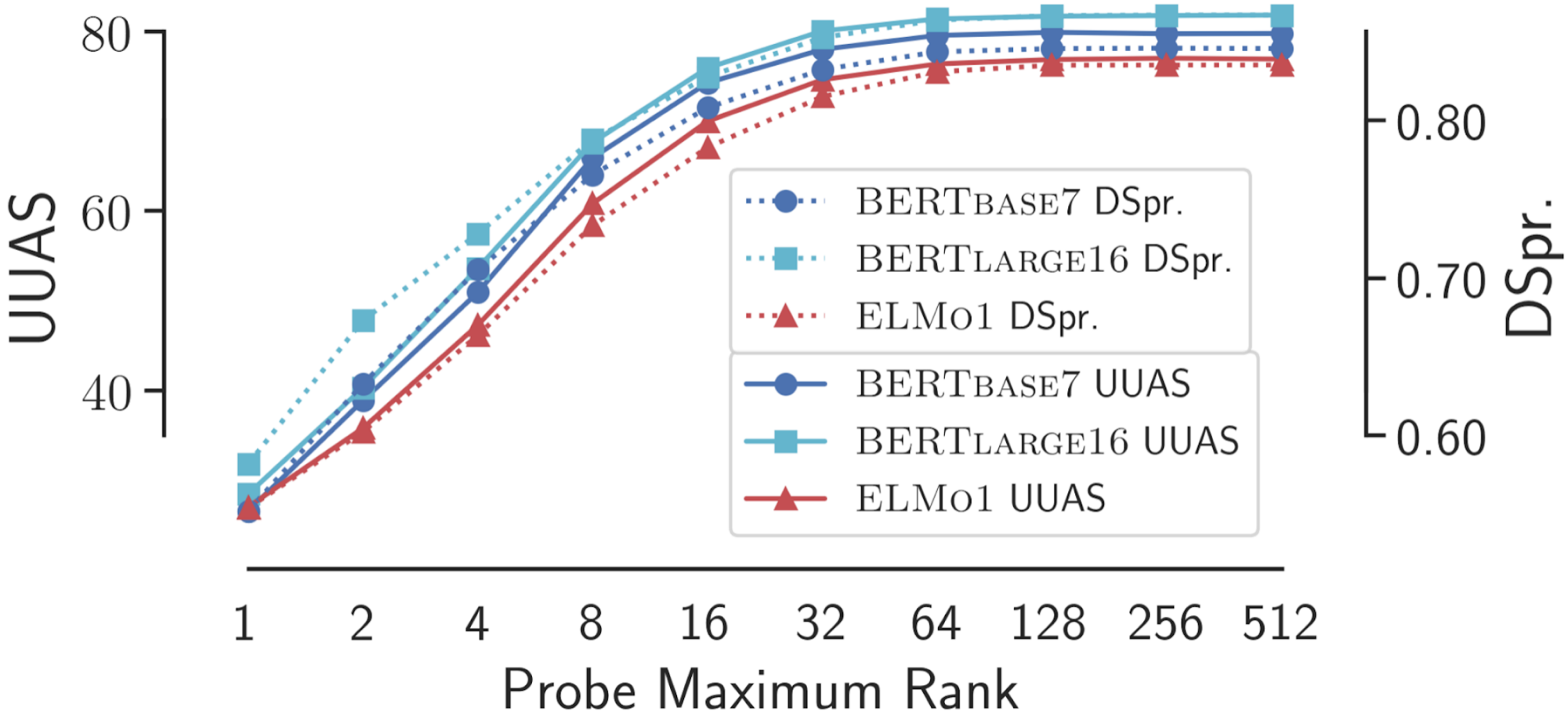
Trees from structural probe parse distances approximate parse trees pretty well!

Black (above sentence): Human-annotated parse tree

Teal (below sentence): Minimum spanning tree, structural probe on BERT



Syntax geometry is quite low rank



Takeaways

We posed the question “Does BERT learn syntax?” as “Does the geometry of syntax trees consistently exist in the hidden states of BERT?”

This allows us to evaluate with a structural probe whether entire parse trees are embedded by a neural network

We find, surprisingly, that entire parse trees are quite accurately embedded in BERT representations, while also confirming that our method isn’t powerful enough to simply “find” parse trees in naive baselines

Lots of exciting work on understanding RNNs and BERT, which connects to psycholinguistics – and to which I can't do justice!

- Futrell, Wilcox, Morita, Qian, Ballesteros, & Levy (2019): LSTMs trained on large datasets represent syntactic state over large text spans
- Linzen & Baroni (2020): "DNNs are capable of high accuracy in challenging syntactic tasks"
- Hu, Gauthier, Qian, Wilcox, & Levy (2020): All neural models are imperfect at syntactic generalization, but Transformer models perform strongly
- Ettinger (2020): BERT good on hypernymy and good/bad sentence completions; struggles with contextual impacts of negation
- Rogers, Kovaleva & Rumshisky (2020): "BERT representations are hierarchical rather than linear"

Final thoughts

Unsupervised or self-supervised learning of linguistic structure can now be done **very** successfully

Deep contextual word representations have phase-shifted from statistical association learners to **language discovery devices**!

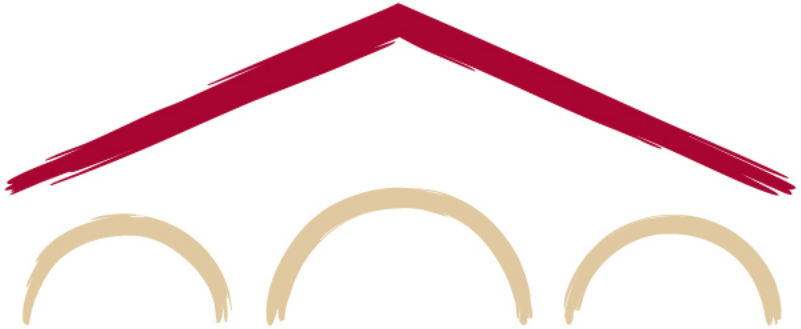
- Syntax emerges (approximately) in the geometry of BERT representations

American structuralists vs. Noam Chomsky

- **Distributional learning** in modern NLP connects with the ideas of late **American structuralism**, sometimes referred to as ***distributionalism***
  - *A school of thought that was rather distinct from European structuralism*

- For the first time now, we are within sight of having **mechanical discovery procedures** for syntax that work from a corpus
  - "The strongest requirement [which C. rejects] … is that the theory must provide a practical and mechanical method for actually constructing the grammar, given a corpus of utterances. Let us say that such a theory provides us with a discovery procedure for grammars." — Chomsky (1957)

Natural Language Processing has been overrun by large neural language models! What should we make of that?



Christopher Manning

@chrmanning ❀ @stanfordnlp

Departments of Linguistics and Computer Science, Stanford University

Director, Stanford Artificial Intelligence Laboratory

CUNY 2021