

Pretraining Language Models (part 1)

Wei Xu

(many slides from Greg Durrett)

Administrivia

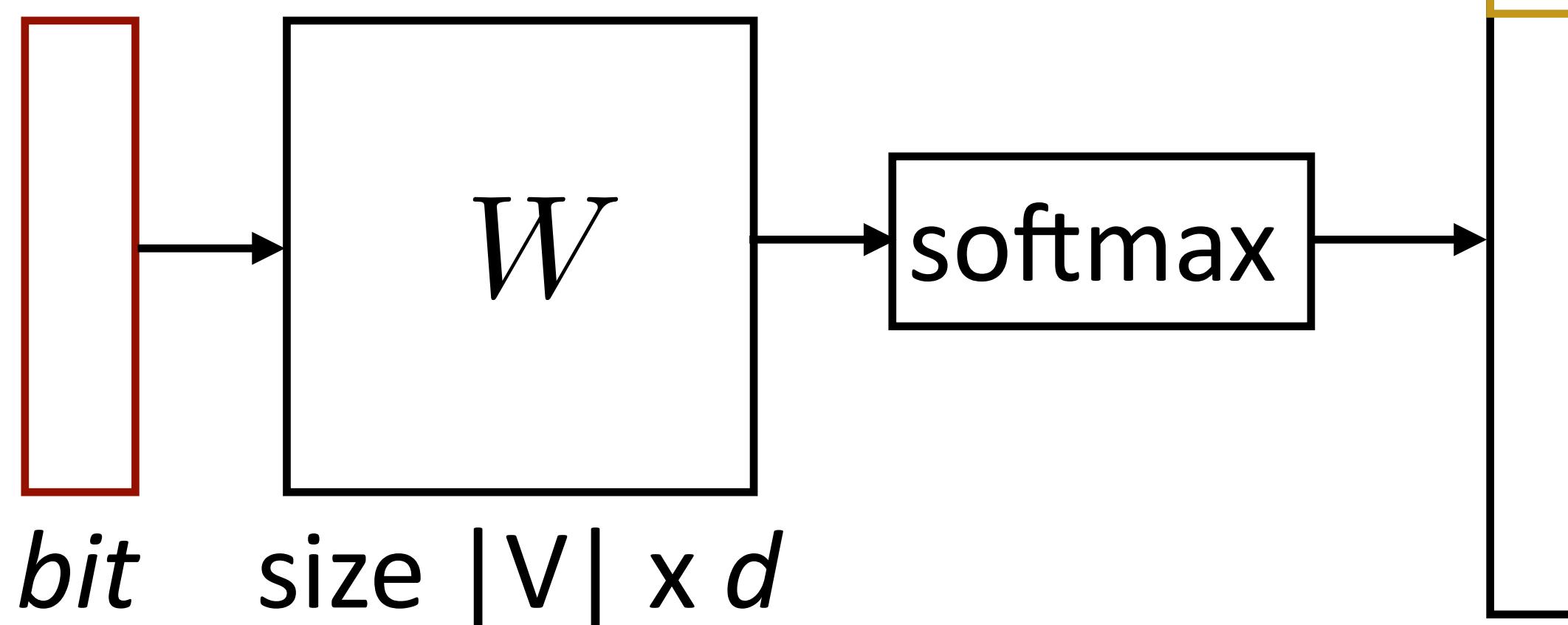
- ▶ Project 3 is released (seq2seq for dialog; can be used for MT)
- ▶ Readings —
 - ▶ ELMo by Peters et al.
<https://aclanthology.org/N18-1202.pdf>
 - ▶ BERT by Devlin et al.
<https://aclanthology.org/N19-1423.pdf>

This Lecture

- ▶ ELMo
- ▶ BERT
- ▶ BERT Results, Extensions
- ▶ Analysis/Visualization of BERT

Recall: word2vec (Skip-Gram)

- ▶ Predict one word of context from word
d-dimensional word embeddings



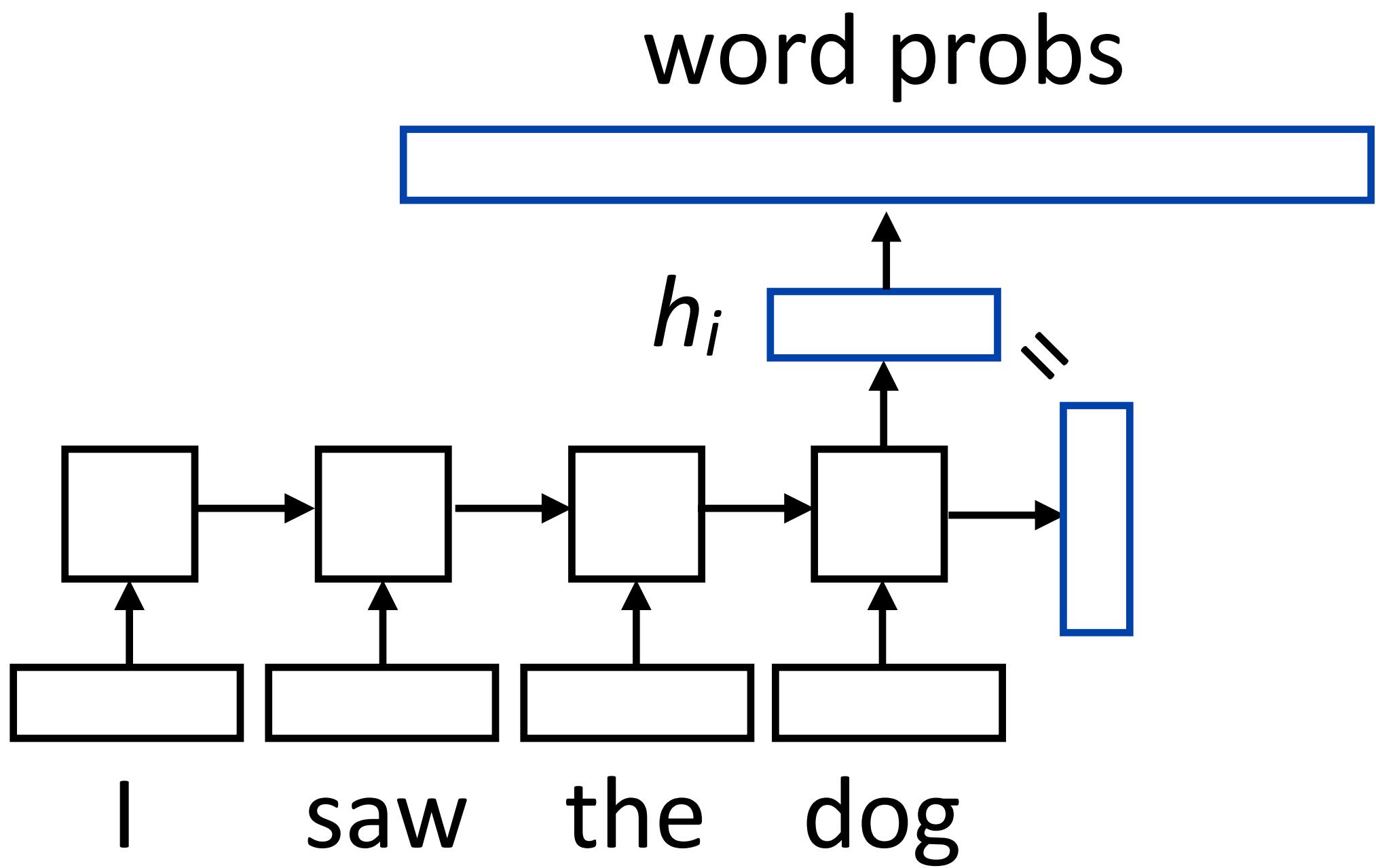
the dog bit the man

gold label = *dog*

$$P(w'|w) = \text{softmax}(We(w))$$

- ▶ Another training example: *bit* -> *the*
- ▶ Parameters: $d \times |V|$ **vectors**, $|V| \times d$ output parameters (W) (also usable as vectors!)

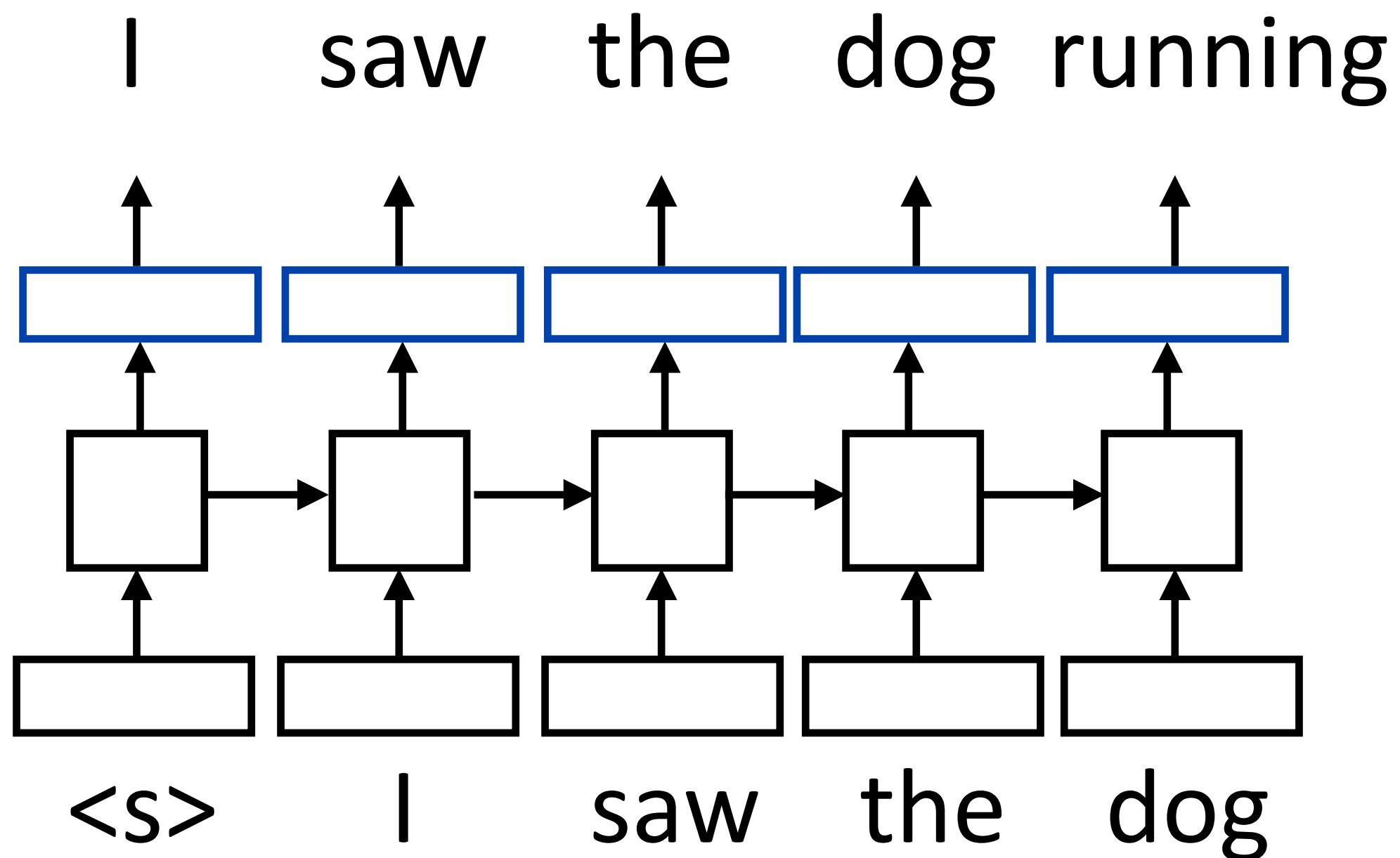
Recall: RNN Language Modeling



$$P(w|\text{context}) = \text{softmax}(W\mathbf{h}_i)$$

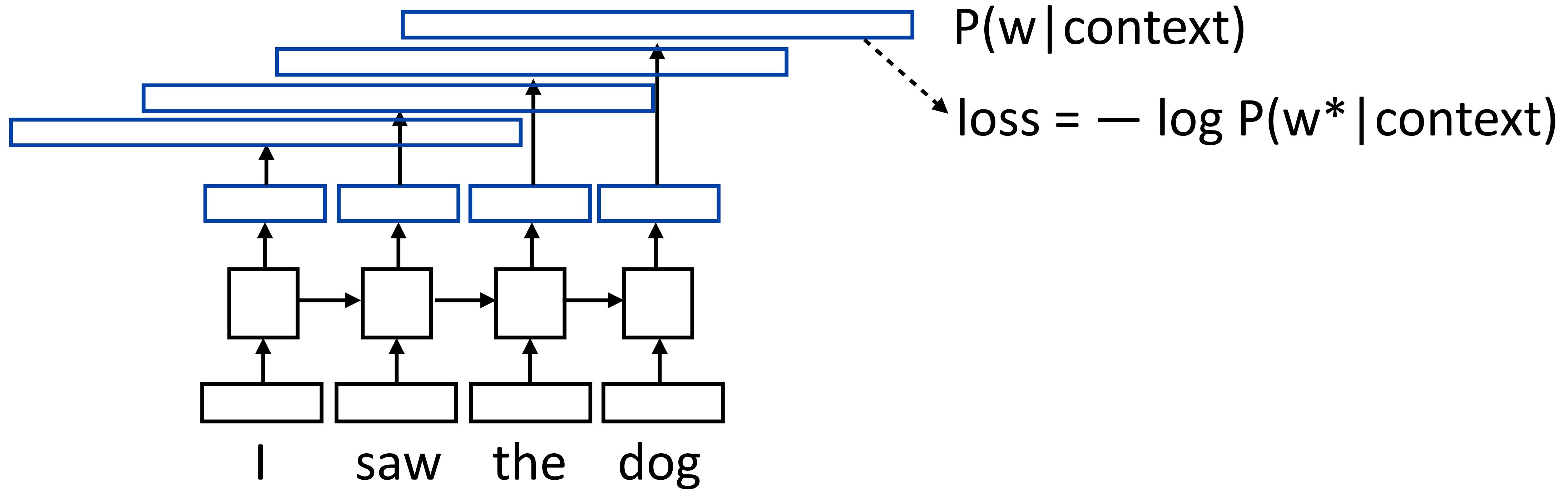
► W is a (vocab size) \times (hidden size) matrix

Recall: Training RNNLMs



- ▶ Input is a sequence of words, output is those words shifted by one,
- ▶ Allows us to efficiently batch up training across time (one run of the RNN)

Recall: Training RNNLMs

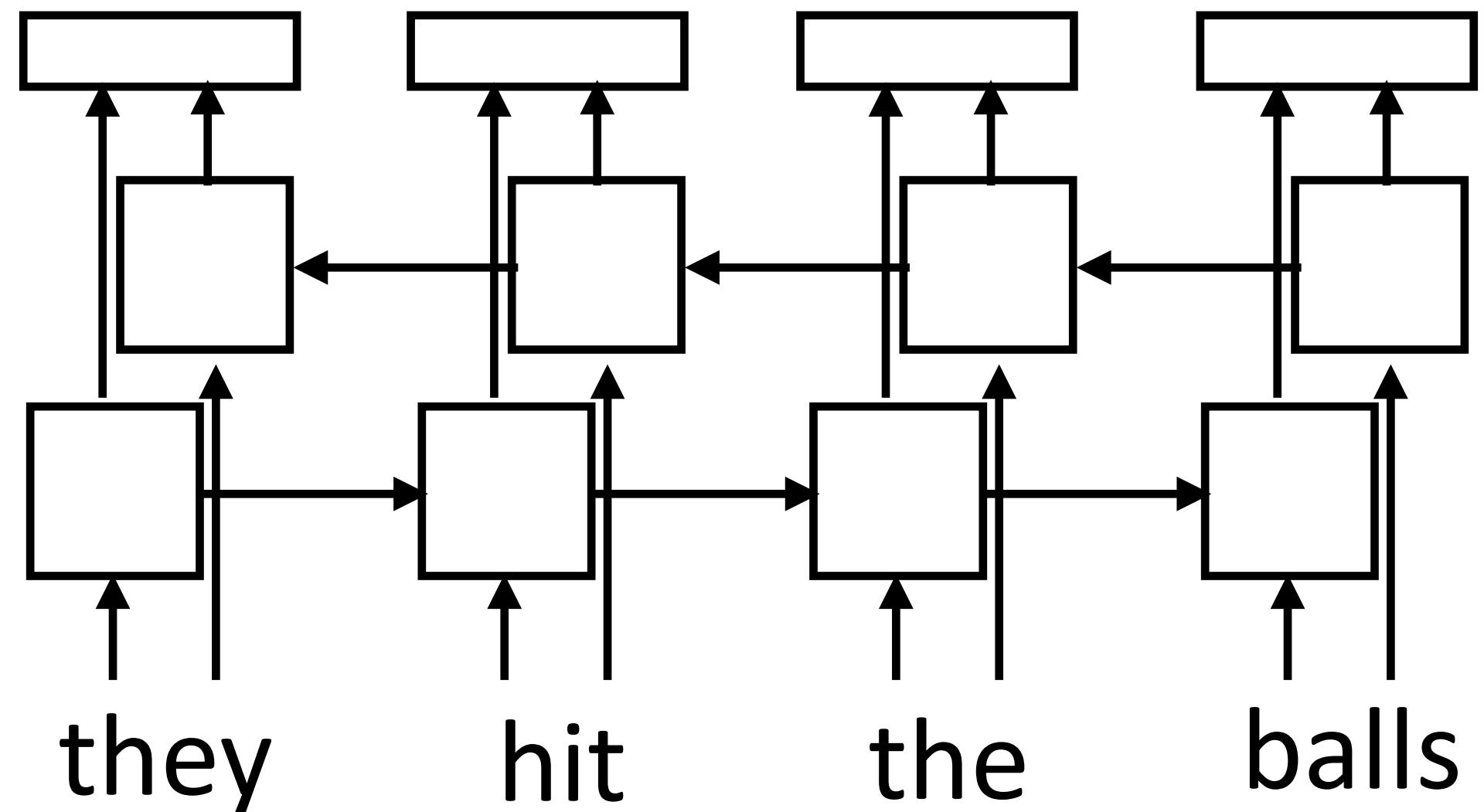
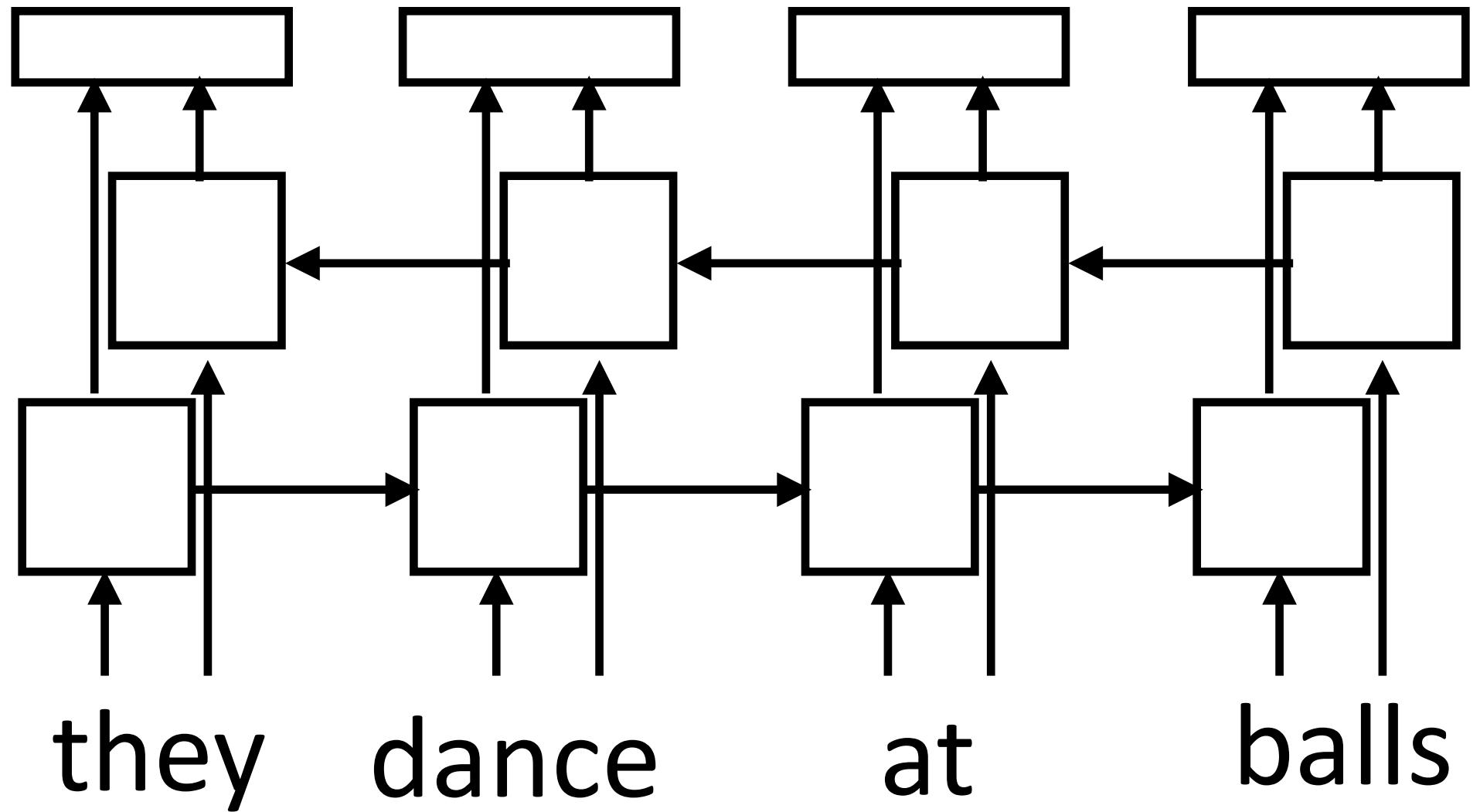


- ▶ Total loss = sum of negative log likelihoods at each position
- ▶ Backpropagate through the network to simultaneously learn to predict next word given previous words at all positions

ELMo

Context-dependent Embeddings

- ▶ How to handle different word senses? One vector for *balls*



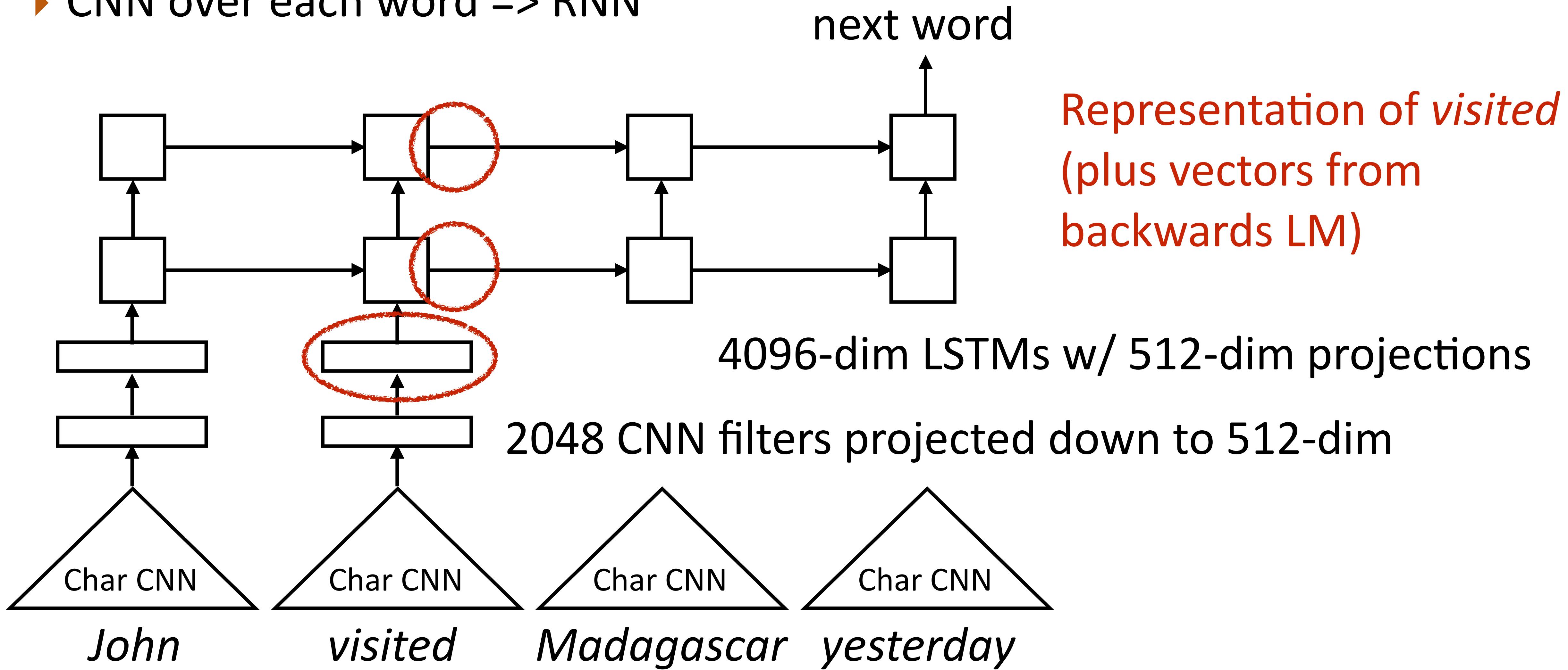
- ▶ Train a neural language model to predict the next word given previous words in the sentence, use its internal representations as word vectors

ELMo

- ▶ Key idea: language models can allow us to form useful word representations in the same way word2vec did
- ▶ Take a powerful language model, train it on large amounts of data, then use those representations in downstream tasks
 - ▶ Data: Wikipedia, books, crawled stuff from the web, ...
- ▶ What do we want our LM to look like?

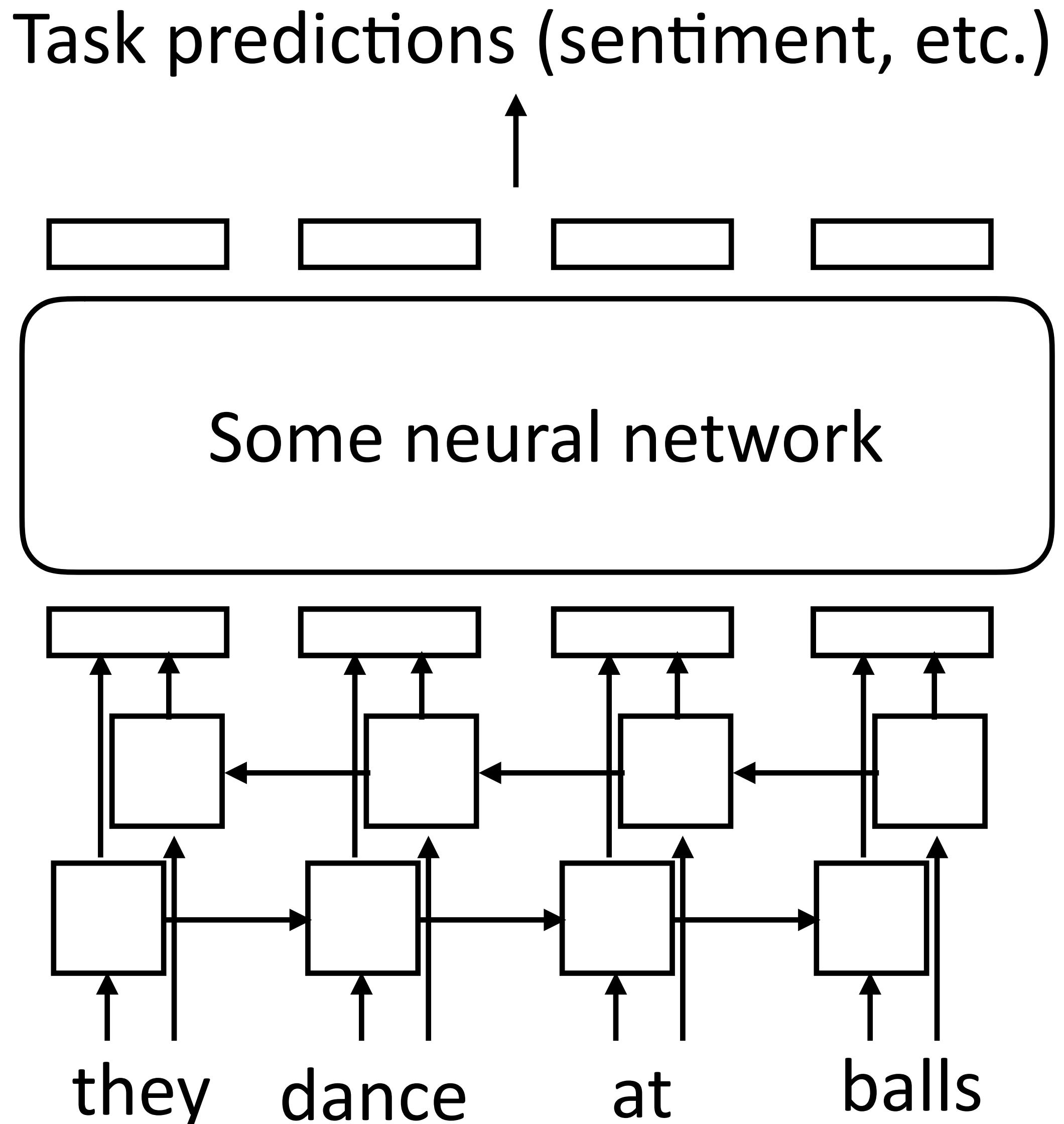
ELMo

- ▶ CNN over each word => RNN



How to apply ELMo?

- ▶ Take those embeddings and feed them into whatever architecture you want to use for your task
- ▶ *Frozen* embeddings: update the weights of your network but keep ELMo's parameters frozen
- ▶ *Fine-tuning*: backpropagate all the way into ELMo when training your model



Results: Frozen ELMo

TASK	PREVIOUS SOTA		OUR BASELINE	ELMo + BASELINE	INCREASE (ABSOLUTE/RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

- ▶ Massive improvements across 5 benchmark datasets: question answering, natural language inference, semantic role labeling (discussed later in the course), coreference resolution, named entity recognition, and sentiment analysis

Peters et al. (2018)

How to apply ELMo?

Pretraining	Adaptation	NER	SA	Nat. lang. inference		Semantic textual similarity		
		CoNLL 2003	SST-2	MNLI	SICK-E	SICK-R	MRPC	STS-B
Skip-thoughts	❄️	-	81.8	62.9	-	86.6	75.8	71.8
ELMo	❄️	91.7	91.8	79.6	86.3	86.1	76.0	75.9
	🔥	91.9	91.2	76.4	83.3	83.3	74.7	75.5
	Δ=🔥-❄️	0.2	-0.6	-3.2	-3.3	-2.8	-1.3	-0.4

- ▶ How does frozen (❄️) vs. fine-tuned (🔥) compare?

- ▶ Recommendations:

Conditions			Guidelines
Pretrain	Adapt.	Task	
Any	❄️	Any	Add many task parameters
Any	🔥	Any	Add minimal task parameters ⚠ Hyper-parameters
Any	Any	Seq. / clas.	❄️ and 🔥 have similar performance
ELMo	Any	Sent. pair	use ❄️
BERT	Any	Sent. pair	use 🔥

Why is language modeling a good objective?

- ▶ “Impossible” problem but bigger models seem to do better and better at distributional modeling (no upper limit yet)
- ▶ Successfully predicting next words requires modeling lots of different effects in text

Context: My wife refused to allow me to come to Hong Kong when the plague was at its height and –” “Your wife, Johanne? You are married at last ?” Johanne grinned. “Well, when a man gets to my age, he starts to need a few home comforts.

Target sentence: After my dear mother passed away ten years ago now, I became _____.

Target word: lonely

- ▶ LAMBADA dataset (Papernot et al., 2016): explicitly targets world knowledge and very challenging LM examples
- ▶ Coreference,

Recall: Winograd Schema

- ▶ Hector Levesque (2011): “Winograd schema challenge” (named after Terry Winograd, the creator of SHRDLU 1968-1972)

The city council refused the demonstrators a permit because they _____ violence

they advocated
they feared

- ▶ This is so complicated that it's an AI challenge problem! (AI-complete)
- ▶ Referential/semantic ambiguity

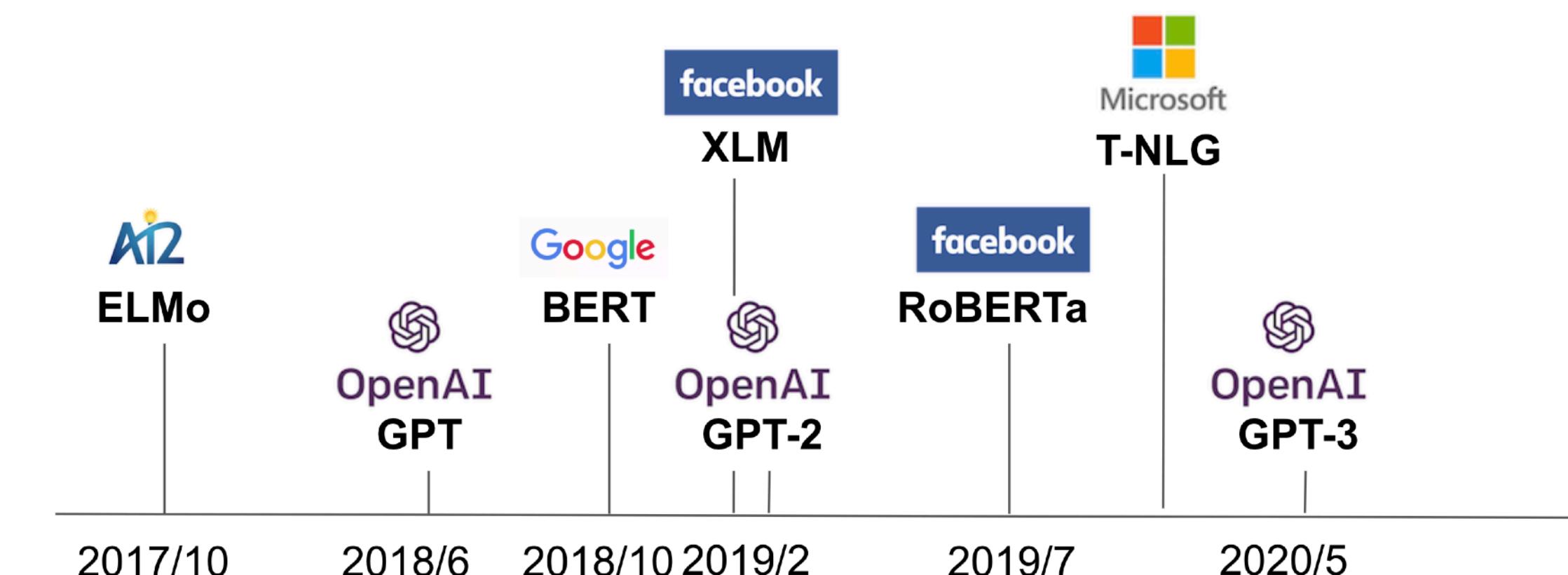
Why did this take time to catch on?

- ▶ Earlier version of ELMo by the same authors in 2017, but it was only evaluated on tagging tasks, gains were 1% or less
- ▶ Required: training on lots of data, having the right architecture, significant hyperparameter tuning

BERT

BERT

- ▶ AI2 released ELMo in 2017-2018, GPT was released in summer 2018, BERT came out October 2018
- ▶ Three major changes compared to ELMo:
 - ▶ Transformers instead of LSTMs (transformers in GPT as well)
 - ▶ Bidirectional <=> Masked LM objective instead of standard LM
 - ▶ Fine-tune instead of freeze at test time



BERT

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

Google AI Language

{jacobdevlin, mingweichang, kentonl, kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

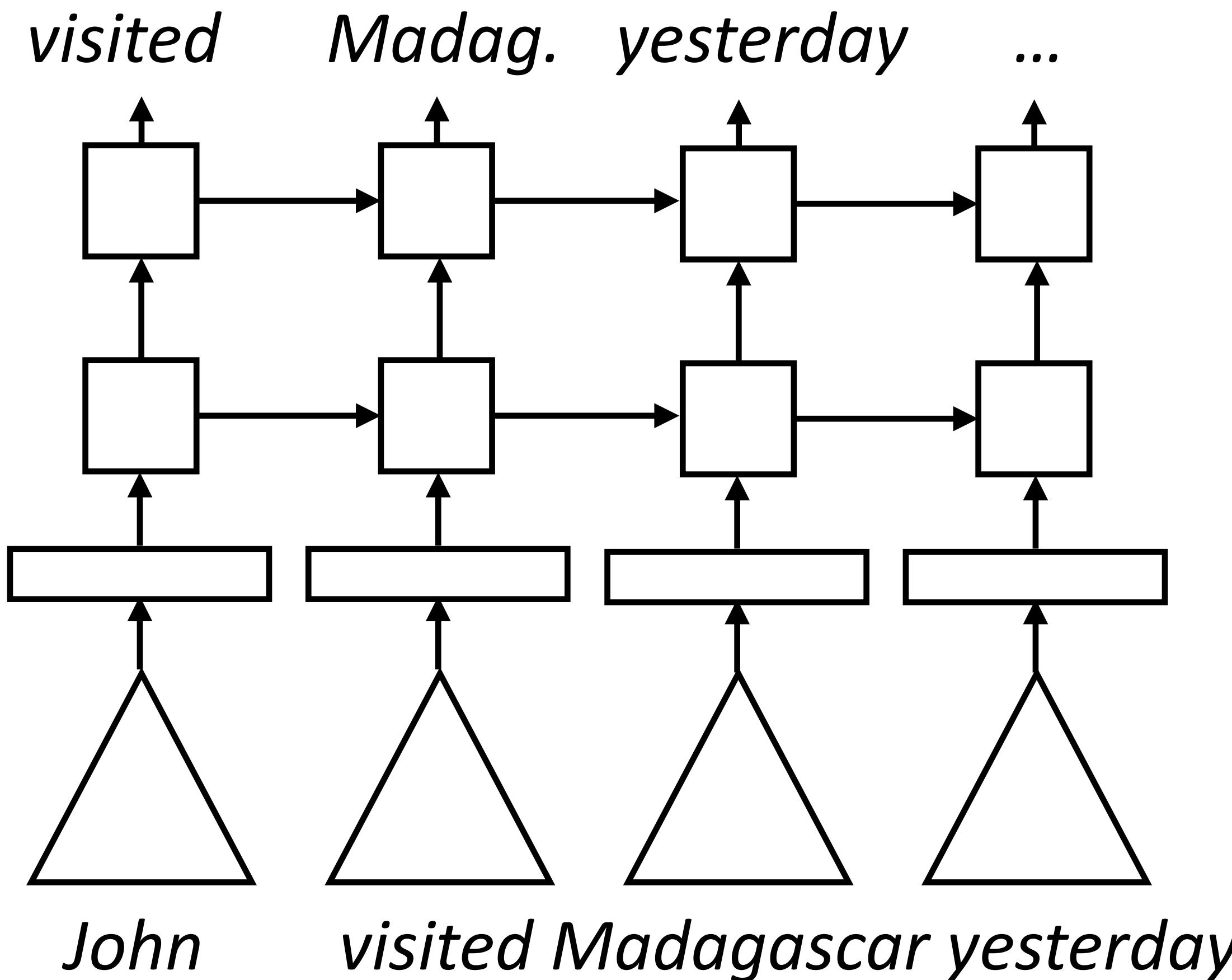
There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers

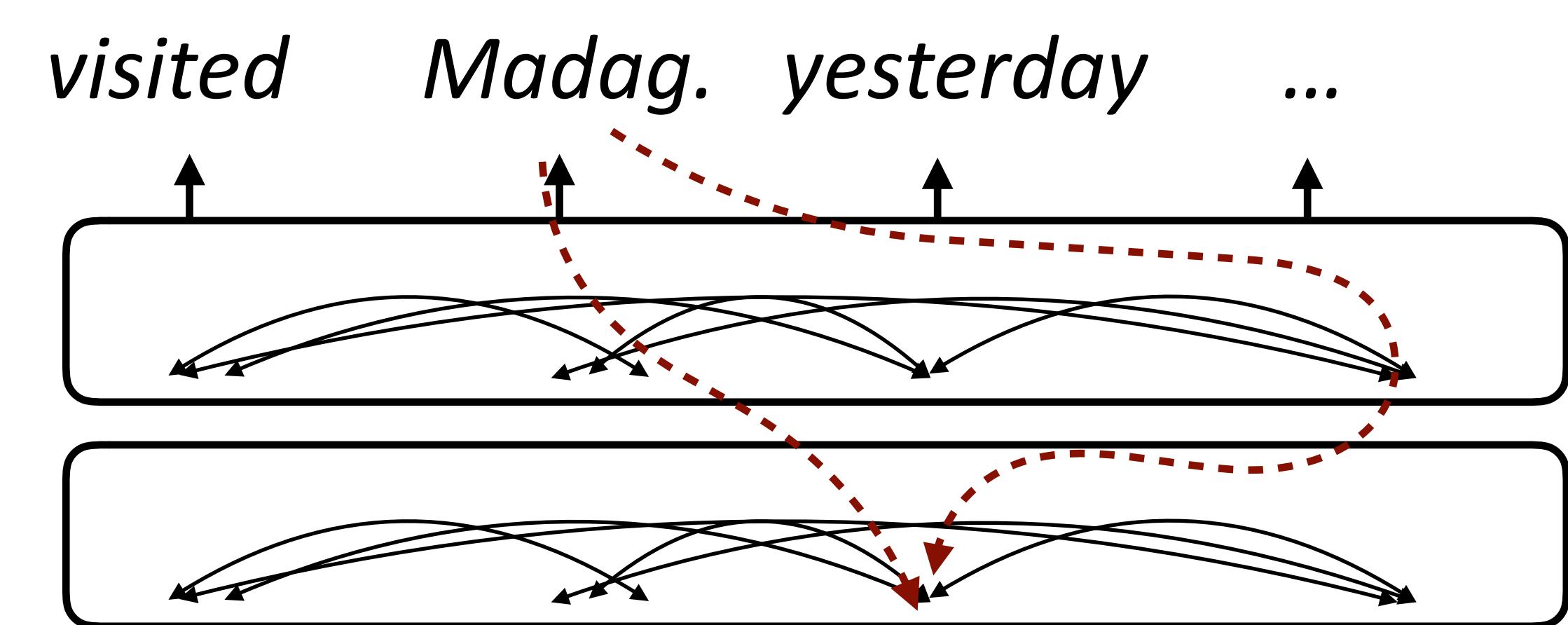
BERT

- ▶ How to learn a “deeply bidirectional” model? What happens if we just replace an LSTM with a transformer?

ELMo (Language Modeling)



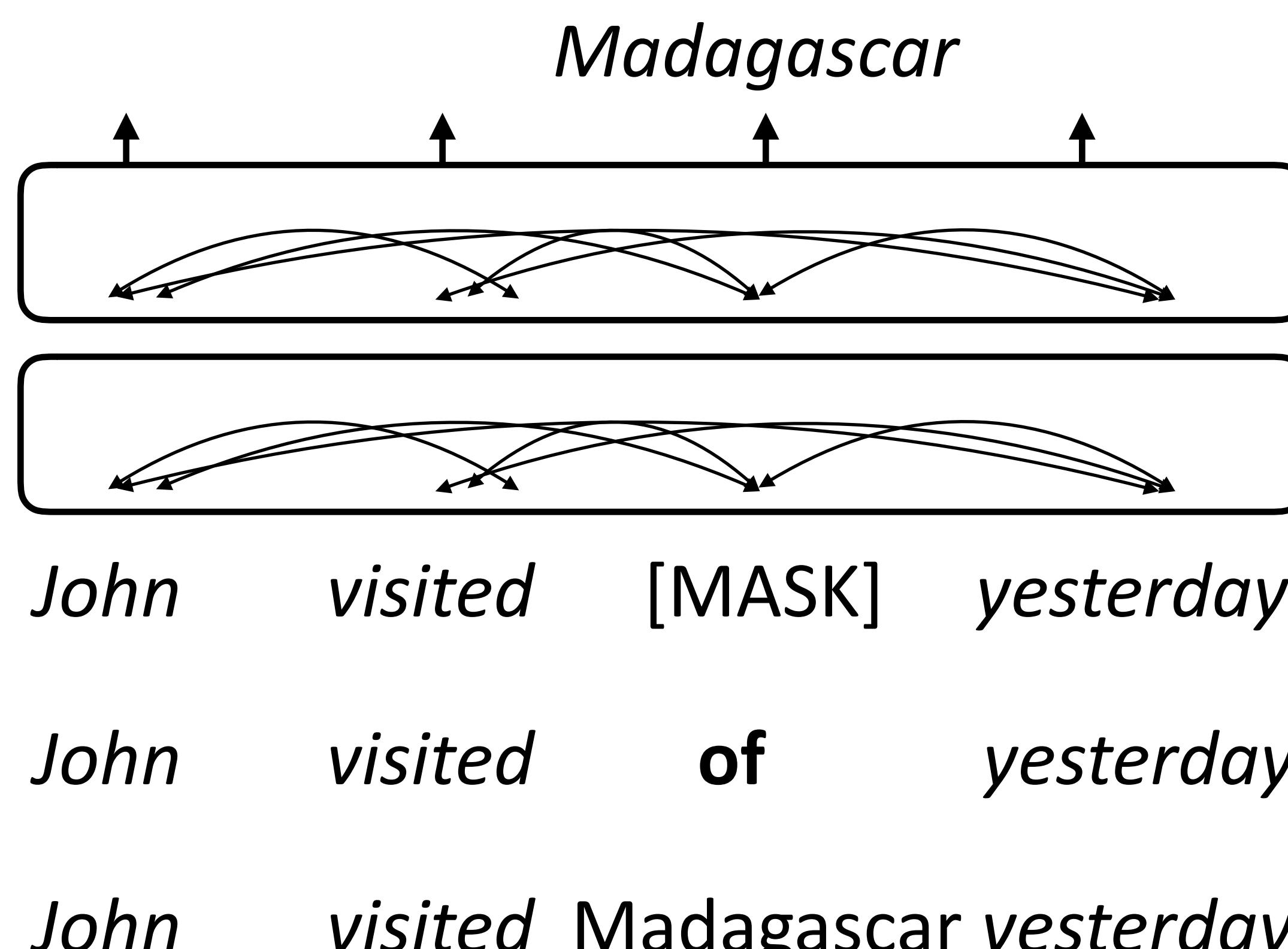
BERT



John visited Madagascar yesterday

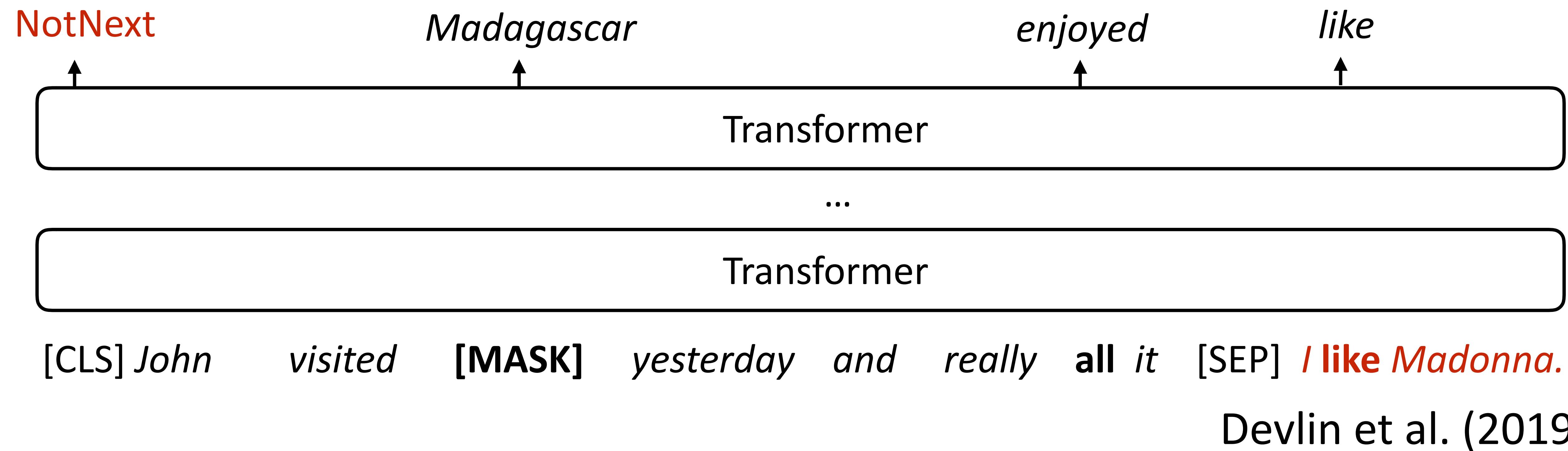
- ▶ Transformer LMs have to be “one-sided” (only attend to previous tokens), not what we want

Masked Language Modeling

- ▶ How to prevent cheating? Next word prediction fundamentally doesn't work for bidirectional models, instead do *masked language modeling*
 - ▶ BERT formula: take a chunk of text, predict 15% of the tokens
 - ▶ For 80% (of the 15%), replace the input token with [MASK]
 - ▶ For 10%, replace w/random
 - ▶ For 10%, keep same (why?)
- 
- John visited [MASK] yesterday
- John visited of yesterday
- John visited Madagascar yesterday

Next “Sentence” Prediction

- ▶ Input: [CLS] Text chunk 1 [SEP] Text chunk 2
- ▶ 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk. Predict whether the next chunk is the “true” next
- ▶ BERT objective: masked LM + next sentence prediction



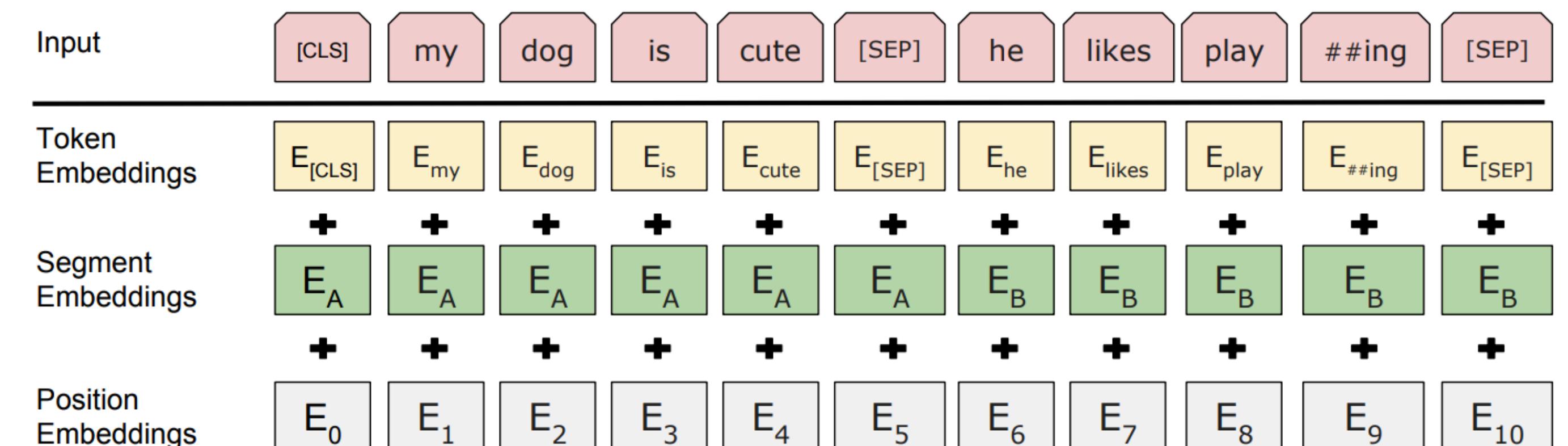
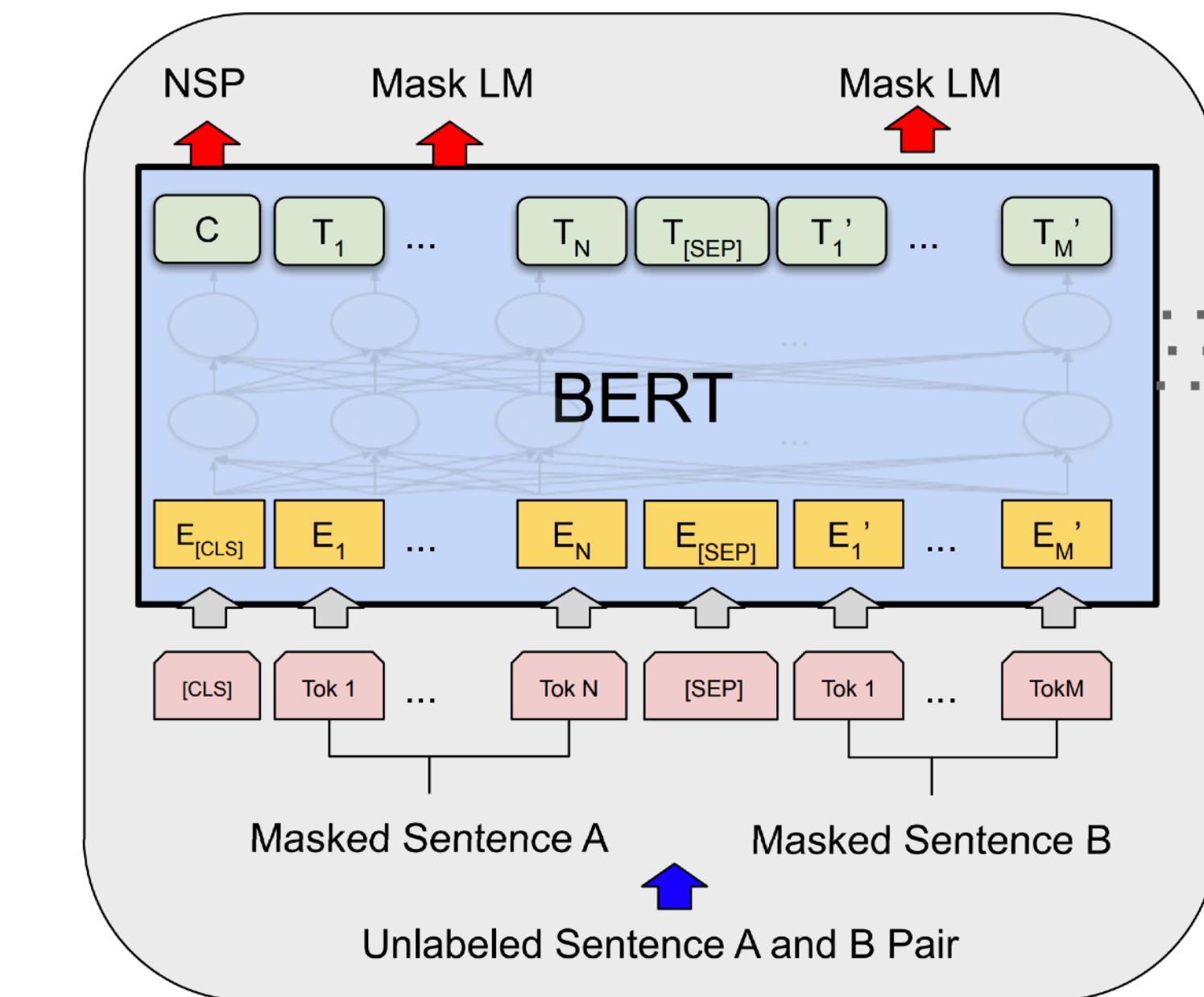
BERT Architecture

- ▶ BERT Base: 12 layers, 768-dim, 12 heads. Total params = 110M

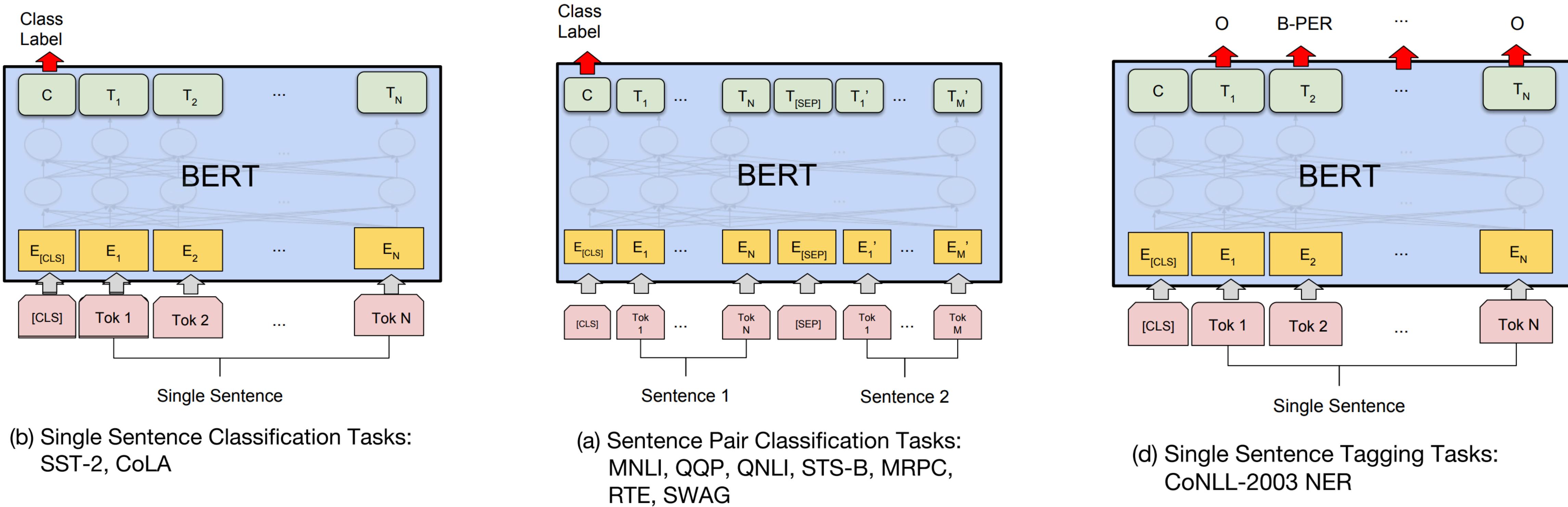
- ▶ BERT Large: 24 layers, 1024-dim, 16 heads. Total params = 340M

- ▶ Positional embeddings and segment embeddings, 30k word pieces

- ▶ This is the model that gets pre-trained on a large corpus



What can BERT do?

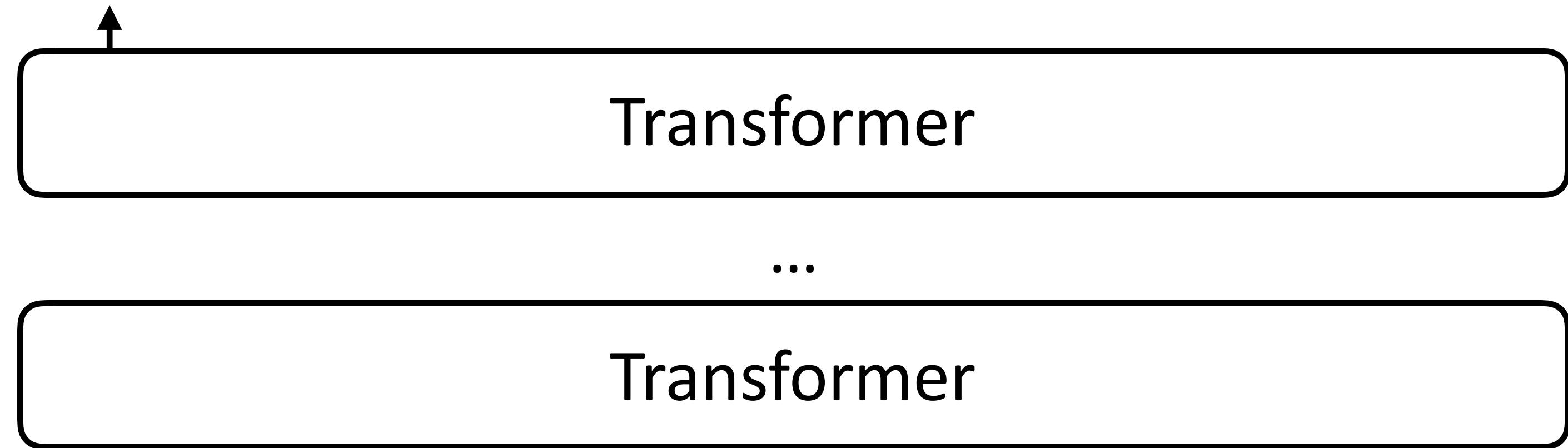


- ▶ CLS token is used to provide classification decisions
- ▶ Sentence pair tasks (entailment): feed both sentences into BERT
- ▶ BERT can also do tagging by predicting tags at each word piece

Devlin et al. (2019)

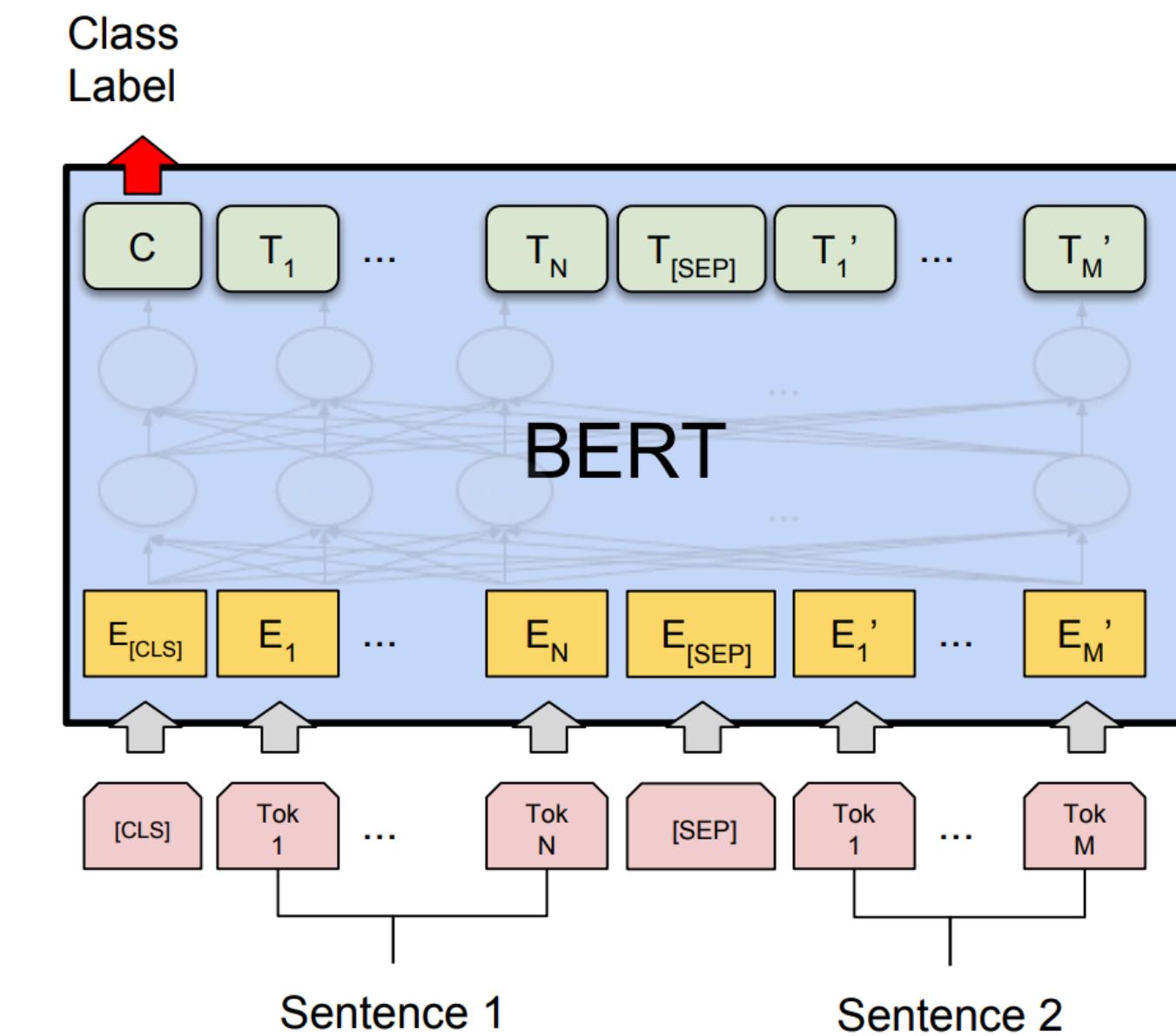
What can BERT do?

Entails



[CLS] A boy plays in the snow [SEP] A boy is outside

- ▶ How does BERT model this sentence pair stuff?
- ▶ Transformers can capture interactions between the two sentences, even though the NSP objective doesn't really cause this to happen



(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

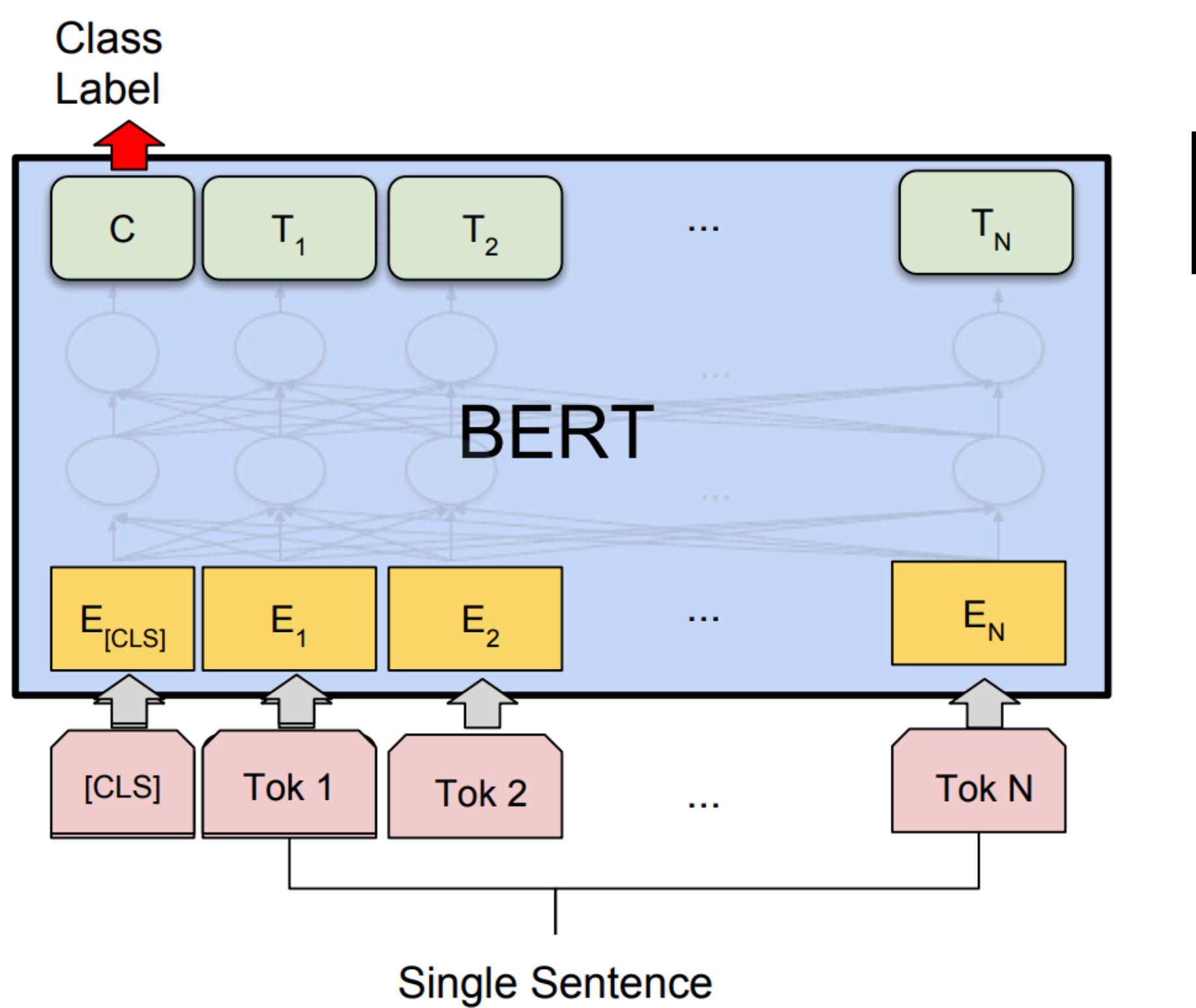
What can BERT NOT do?

- ▶ BERT **cannot** generate text (at least not in an obvious way)
- ▶ Not an autoregressive model, can do weird things like stick a [MASK] at the end of a string, fill in the mask, and repeat
- ▶ Masked language models are intended to be used primarily for “analysis” tasks

BERT Results, Extensions

Fine-tuning BERT

- ▶ Fine-tune for 1-3 epochs, batch size 2-32, learning rate 2e-5 - 5e-5



(b) Single Sentence Classification Tasks:
SST-2, CoLA

- ▶ Large changes to weights up here (particularly in last layer to route the right information to [CLS])
- ▶ Smaller changes to weights lower down in the transformer
- ▶ Small LR and short fine-tuning schedule mean weights don't change much
- ▶ More complex “triangular learning rate” schemes exist

Fine-tuning BERT

- ▶ How does frozen (❄️) vs. fine-tuned (🔥) compare?

Pretraining	Adaptation	NER		SA	Nat. lang. inference		Semantic textual similarity		
		CoNLL 2003	SST-2	MNLI	SICK-E	SICK-R	MRPC	STS-B	
Skip-thoughts	❄️	-	81.8	62.9	-	86.6	75.8	71.8	
ELMo	❄️	91.7	91.8	79.6	86.3	86.1	76.0	75.9	
	🔥	91.9	91.2	76.4	83.3	83.3	74.7	75.5	
	Δ=🔥-❄️	0.2	-0.6	-3.2	-3.3	-2.8	-1.3	-0.4	
BERT-base	❄️	92.2	93.0	84.6	84.8	86.4	78.1	82.9	
	🔥	92.4	93.5	84.6	85.8	88.7	84.8	87.1	
	Δ=🔥-❄️	0.2	0.5	0.0	1.0	2.3	6.7	4.2	

- ▶ BERT is typically better if the whole network is fine-tuned, unlike ELMo

Evaluation: GLUE

Corpus	Train	Test	Task	Metrics	Domain
Single-Sentence Tasks					
CoLA	8.5k	1k	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
Similarity and Paraphrase Tasks					
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	391k	paraphrase	acc./F1	social QA questions
Inference Tasks					
MNLI	393k	20k	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	146	coreference/NLI	acc.	fiction books

Results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT _{BASE}	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	91.1	94.9	60.5	86.5	89.3	70.1	81.9

- ▶ Huge improvements over prior work (even compared to ELMo)
- ▶ Effective at “sentence pair” tasks: textual entailment (does sentence A imply sentence B), paraphrase detection

Subsequent Improvements to BERT

- ▶ Dynamic masking: standard BERT uses the same MASK scheme for every epoch, RoBERTa recomputes them

epoch 2

epoch 1

... John visited Madagascar yesterday ...

- ▶ Whole word masking: don't mask out parts of words

... _John _visited _Mada gas car yesterday ...

RoBERTa

- ▶ “Robustly optimized BERT” incorporating some of these tricks
- ▶ 160GB of data instead of 16 GB
- ▶ New training + more data = better performance

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	94.6/89.4	90.2	96.4
BERT _{LARGE}						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

ALBERT

- ▶ A Lite BERT (18x fewer parameters, 1.7x faster training than BERT)
- ▶ Factorized embedding matrix to save parameters, model context-independent words with fewer parameters

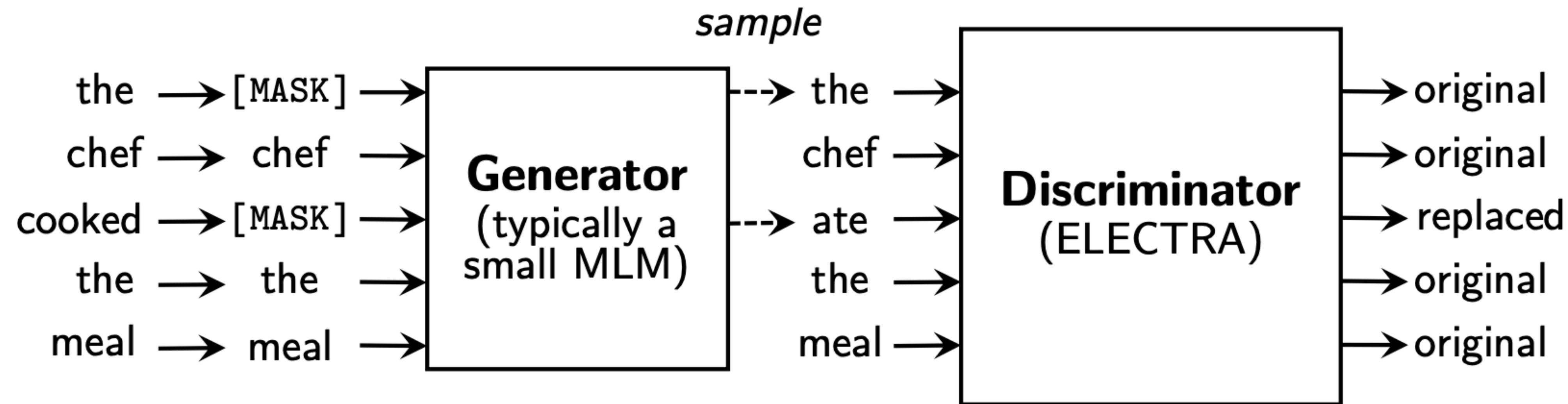
Ordinarily $|V| \times H - |V|$ is 30k-90k, H is >1000

Factor into two matrices with a low-rank approximation

Now: $|V| \times E$ and $E \times H - E$ is 128 in their implementation

- ▶ Additional cross-layer parameter sharing

ELECTRA



- ▶ No need to necessarily have a generative model (predicting words)
- ▶ This objective is more computationally efficient (trains faster) than the standard BERT objective

BERT/MLMs

- ▶ There are lots of ways to train these models!
- ▶ Key factors:
 - ▶ Big enough model
 - ▶ Big enough data
 - ▶ Well-designed “self-supervised” objective (something like language modeling). Needs to be a hard enough problem!

Analysis/Visualization of BERT

Probing BERT

- ▶ A probe is a simple classifier that uses the feature presentations (e.g. hidden states in different layers, attention weights, etc.) from a pre-trained LM to perform a supervised task (e.g., SRL)

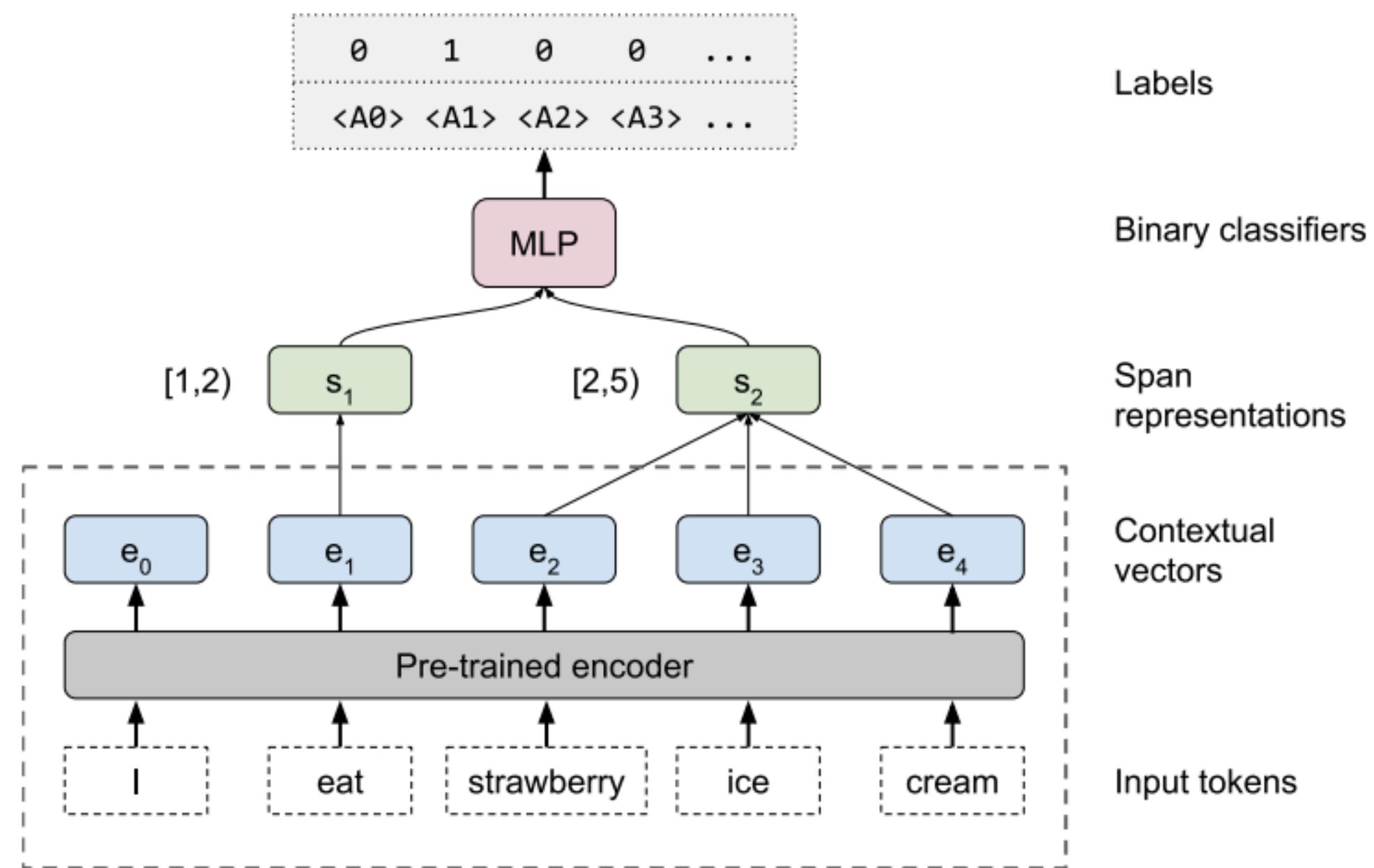


Figure 1: Probing model architecture (§ 3.1). All parameters inside the dashed line are fixed, while we train the span pooling and MLP classifiers to extract information from the contextual vectors. The example shown is for semantic role labeling, where $s^{(1)} = [1, 2]$ corresponds to the predicate (“eat”), while $s^{(2)} = [2, 5]$ is the argument (“strawberry ice cream”), and we predict label A1 as positive and others as negative. For entity and constituent labeling, only a single span is used.

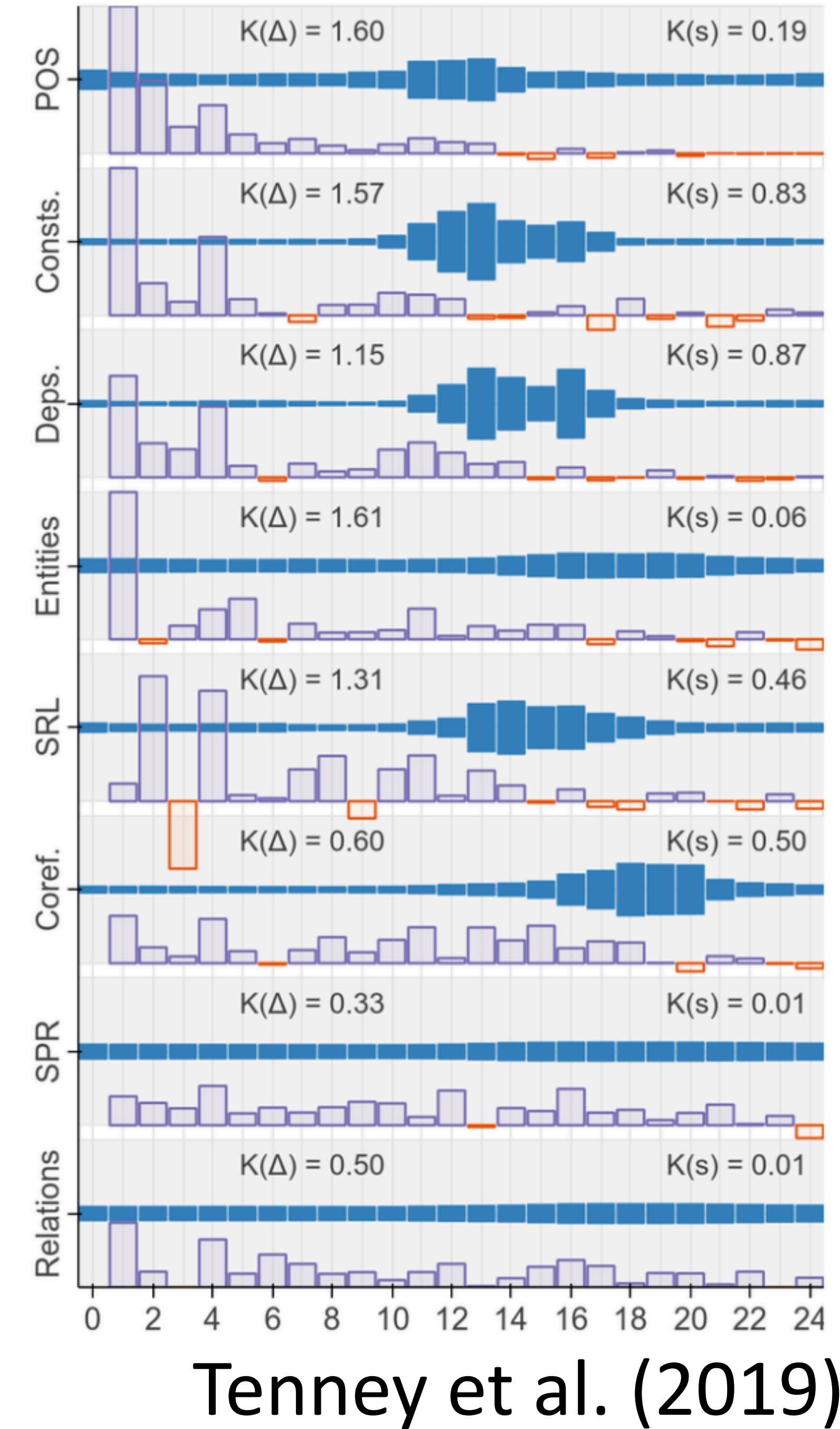
Probing BERT

- ▶ Try to predict POS, etc. from each layer.
Learn mixing weights

$$\mathbf{h}_{i,\tau} = \gamma_\tau \sum_{\ell=0}^L s_\tau^{(\ell)} \mathbf{h}_i^{(\ell)}$$

↑
representation of wordpiece i for task τ

- ▶ Plot shows s weights (blue) and performance deltas when an additional layer is incorporated (purple/orange)
- ▶ BERT “redisCOVERS the classical NLP pipeline”: first syntactic tasks then semantic ones



Tenney et al. (2019)

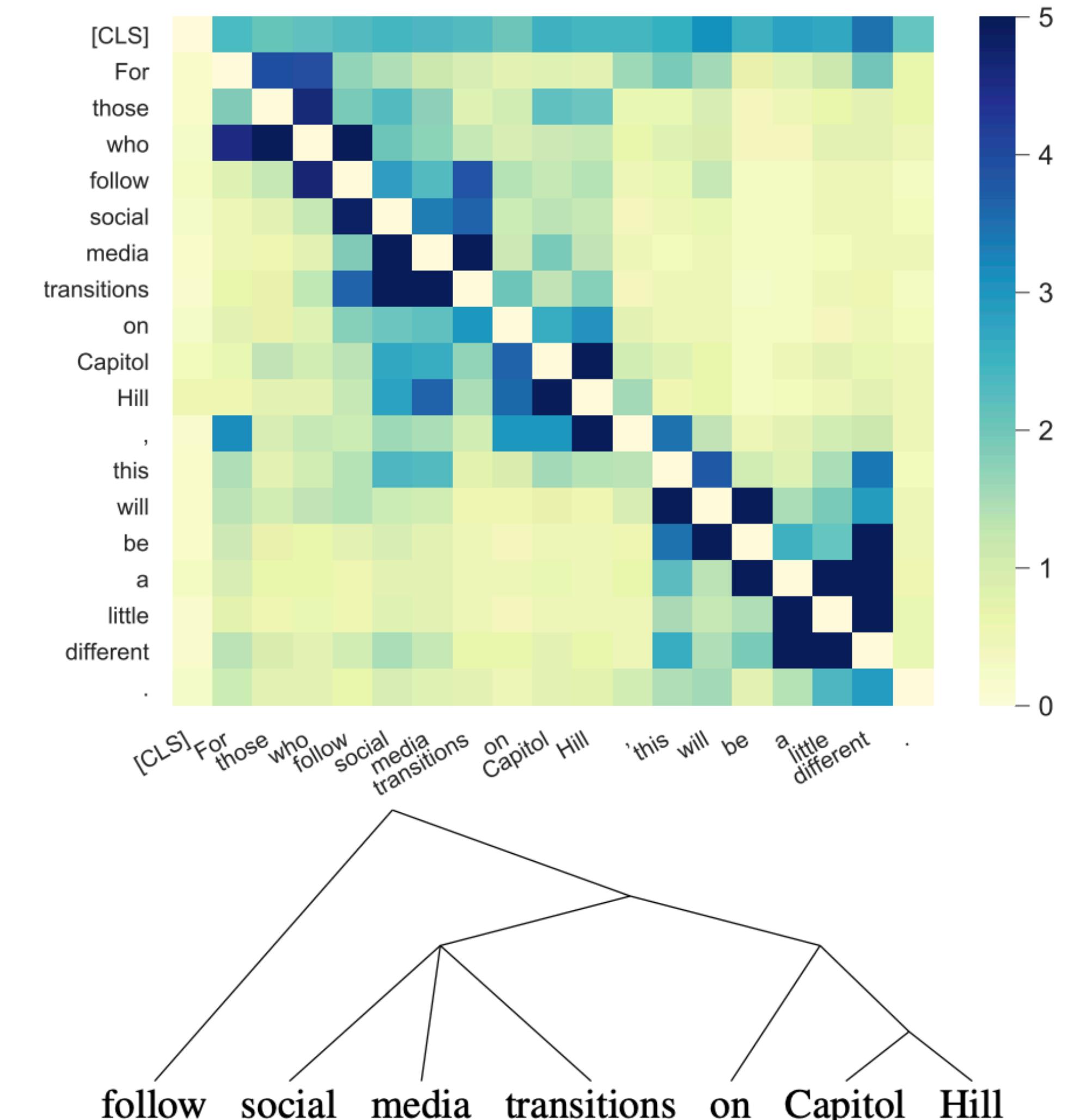
BERTology

- (1) How can we probe syntactic + semantic knowledge of BERT? What does BERT “know” in its representations?
- (2) What can we learn from looking at attention heads?
- (3) What can we learn about training BERT (more efficiently, etc.)?

BERTology: Probing

(1) In general: set up some “probing” task to try to determine syntactic features from BERT’s hidden states

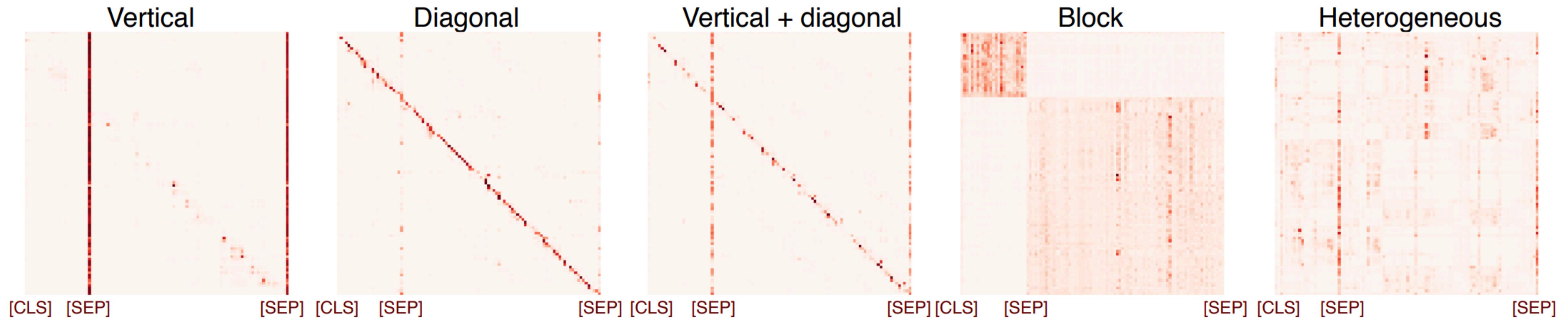
E.g.: Words with syntactic relations have a higher impact on one another during MLM prediction



Rogers et al. (2020)

BERTology

(2) What's going inside attention heads?



BERTology

(2) What's going inside attention heads?

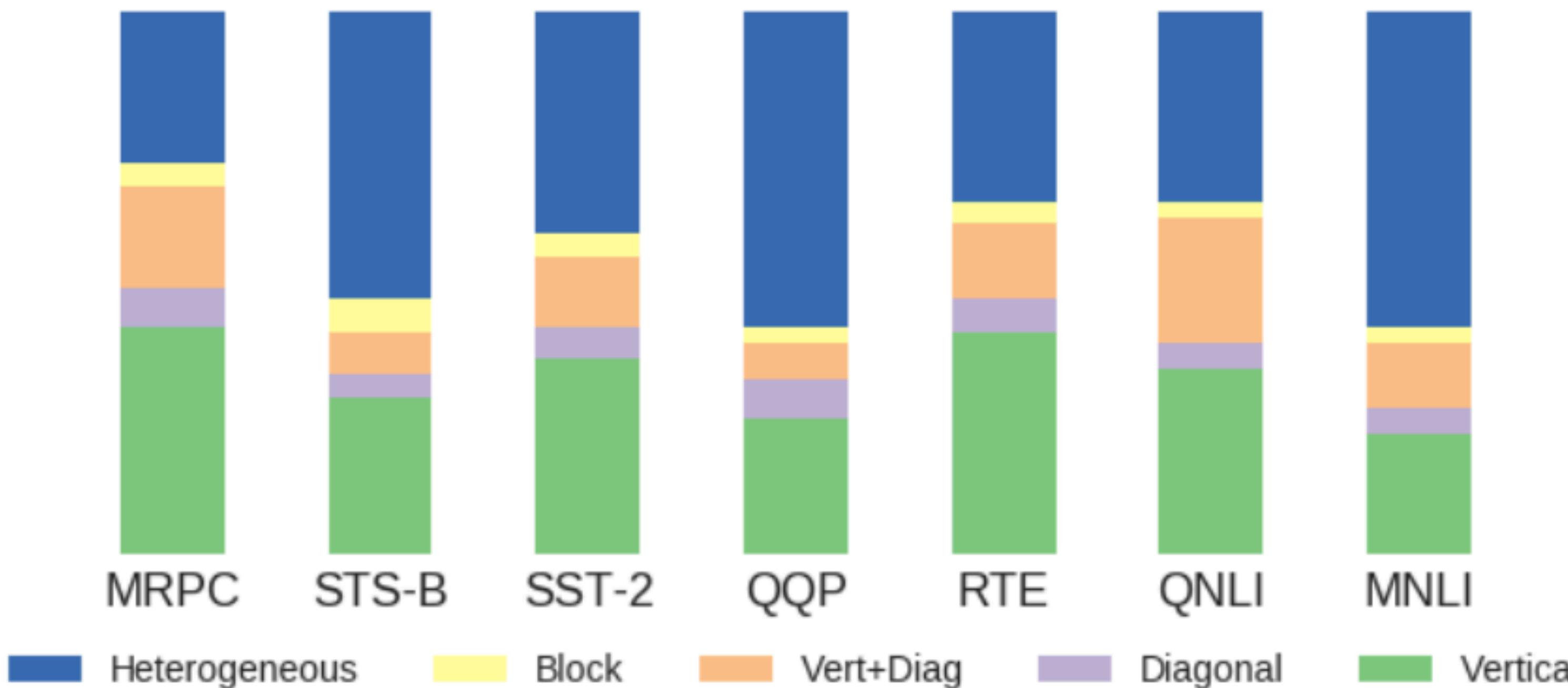
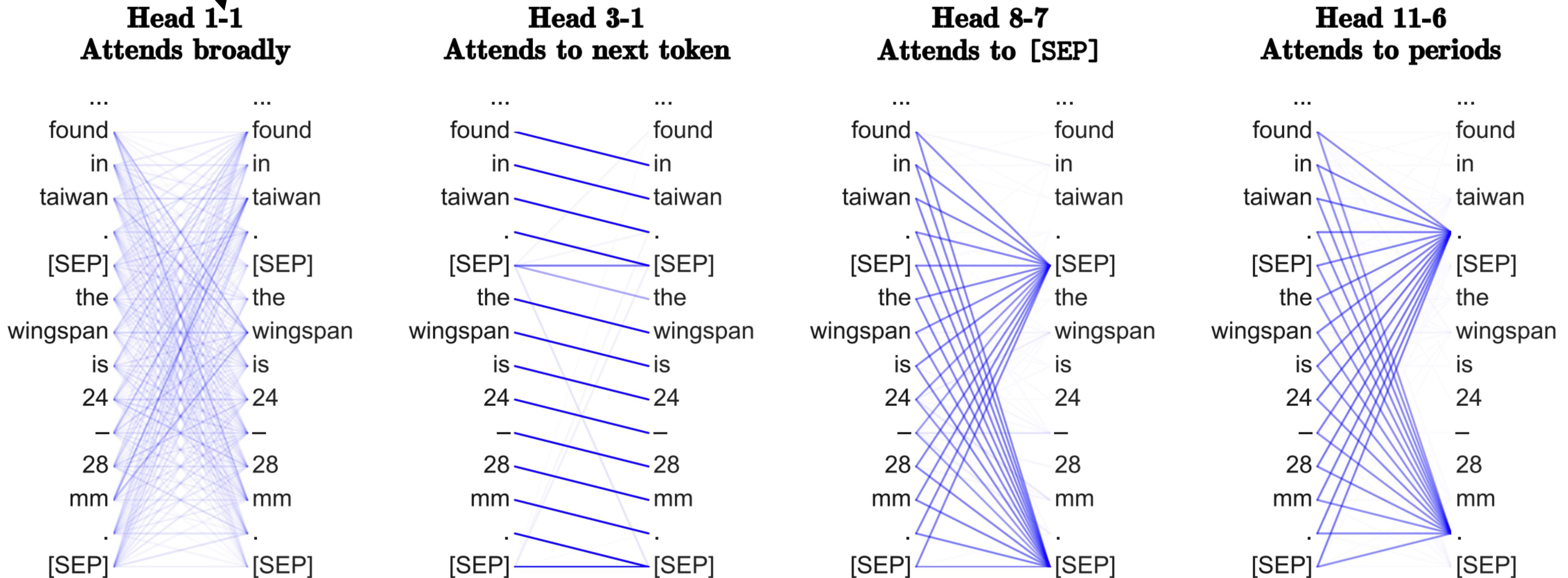


Figure 2: Estimated percentages of the identified self-attention classes for each of the selected GLUE tasks.

Kovaleva et al. (2019)

layer-head

What does BERT learn?

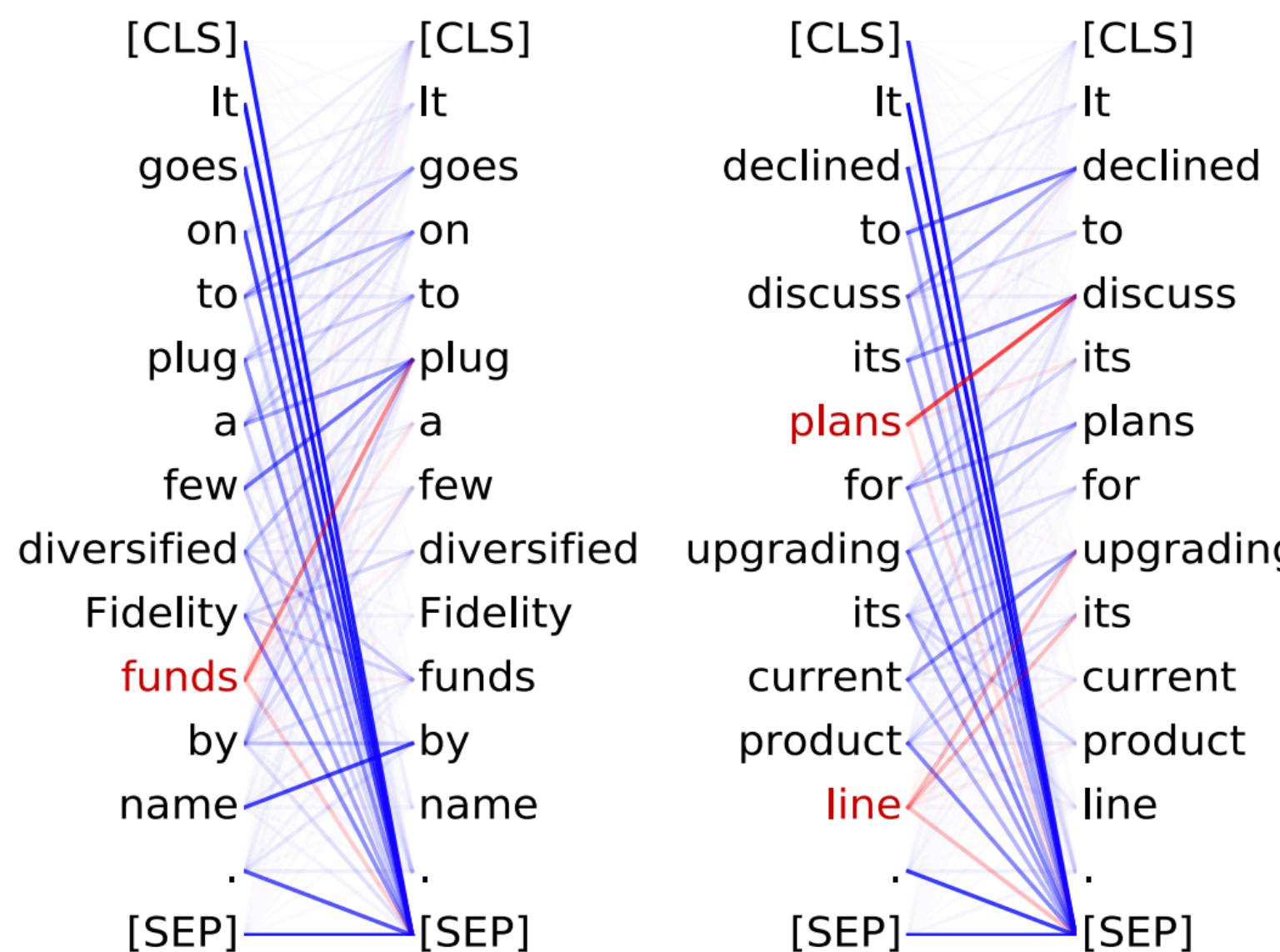


- ▶ Heads on transformers learn interesting and diverse things: content heads (attend based on content), positional heads (based on position), etc.

What does BERT learn?

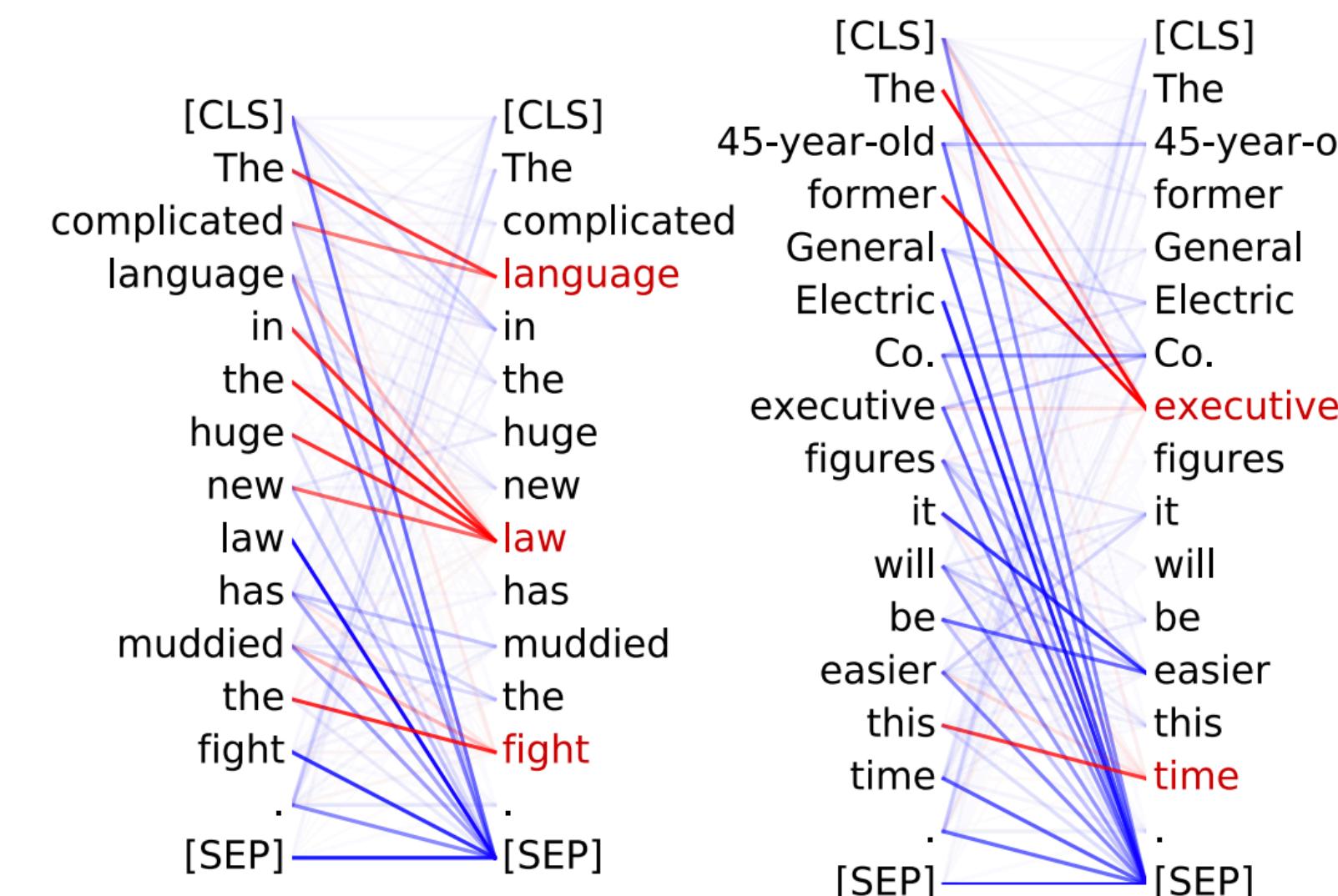
Head 8-10

- **Direct objects** attend to their verbs
- 86.8% accuracy at the `dobj` relation



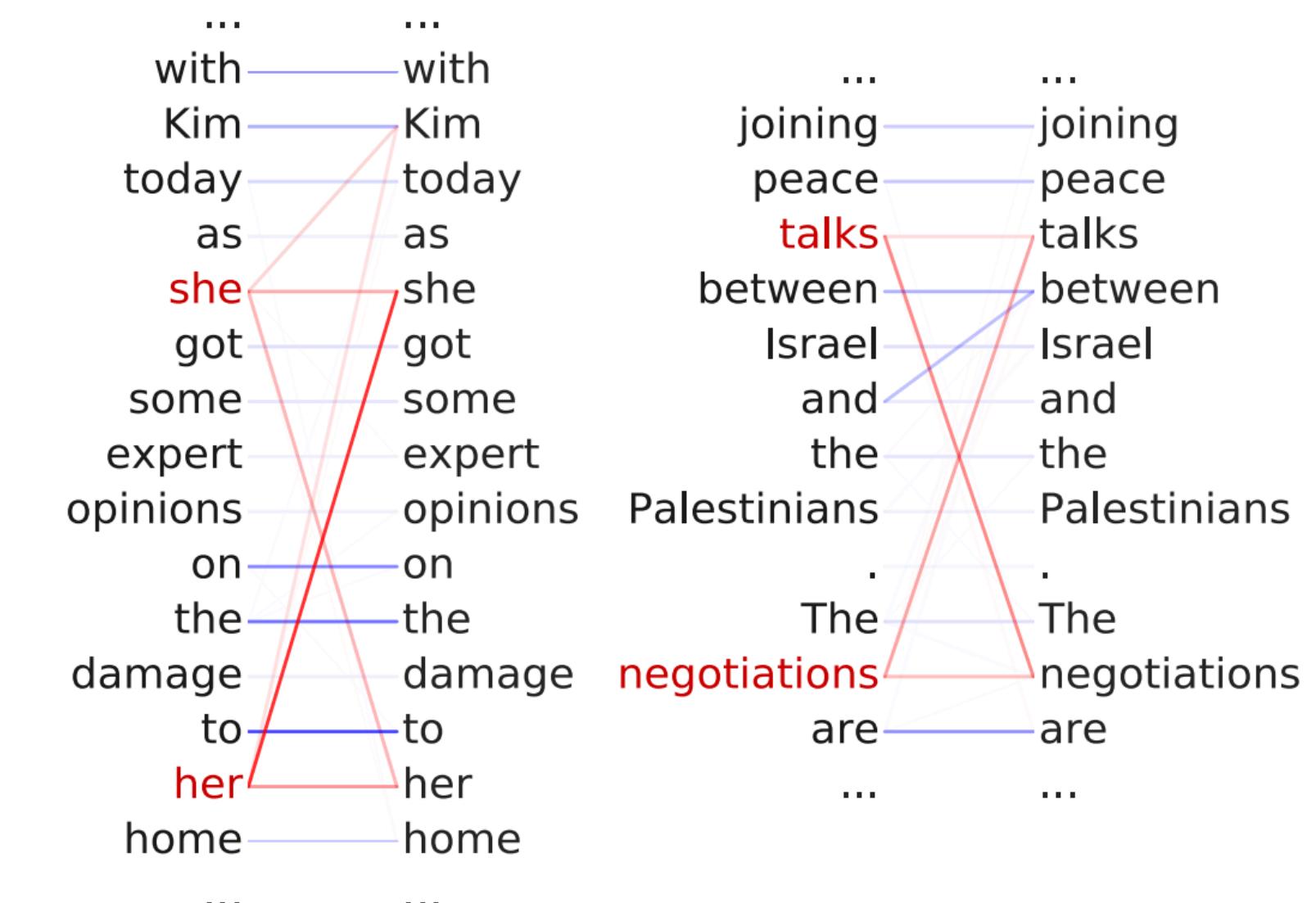
Head 8-11

- **Noun modifiers** (e.g., determiners) attend to their noun
- 94.3% accuracy at the `det` relation



Head 5-4

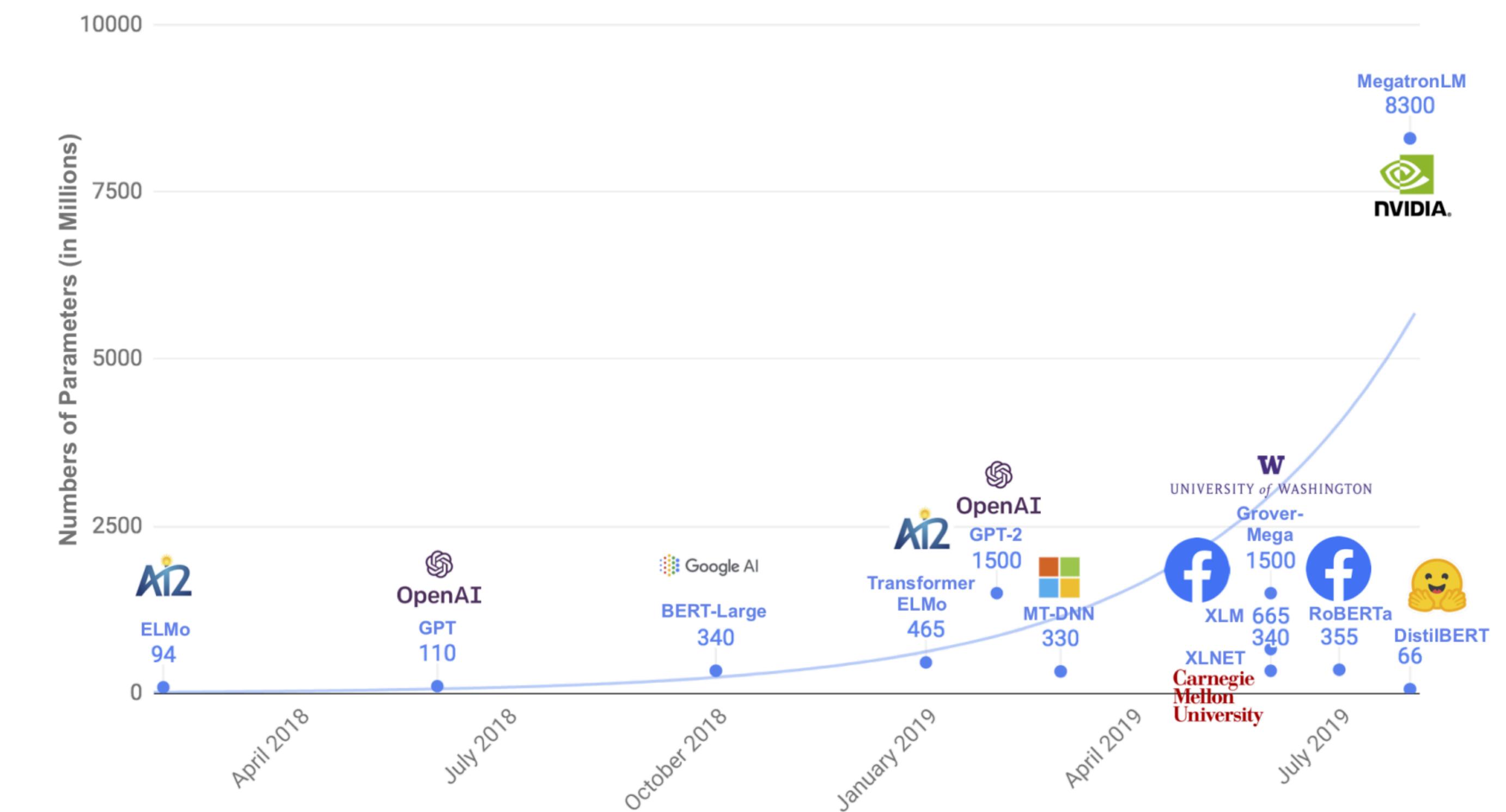
- **Coreferent** mentions attend to their antecedents
- 65.1% accuracy at linking the head of a coreferent mention to the head of an antecedent



► Still way worse than what supervised systems can do, but interesting that this is learned organically

Compressing BERT

- ▶ Remove 60+% of BERT's heads post-training with minimal drop in performance
- ▶ DistilBERT (Sanh et al., 2019): nearly as good with half the parameters of BERT (via knowledge distillation)



Michel et al. (2019)

Takeaways

- ▶ BERT-based systems are state-of-the-art for nearly every major text analysis task
- ▶ Transformers + lots of data + self-supervision seems to do very well
- ▶ Lots of work studying and analyzing these, but few “deep” conclusions have emerged
- ▶ Next time: GPT/GPT-2, BART/T5, GPT-3, etc.