

# Lecture 13: Pretraining + Transformers

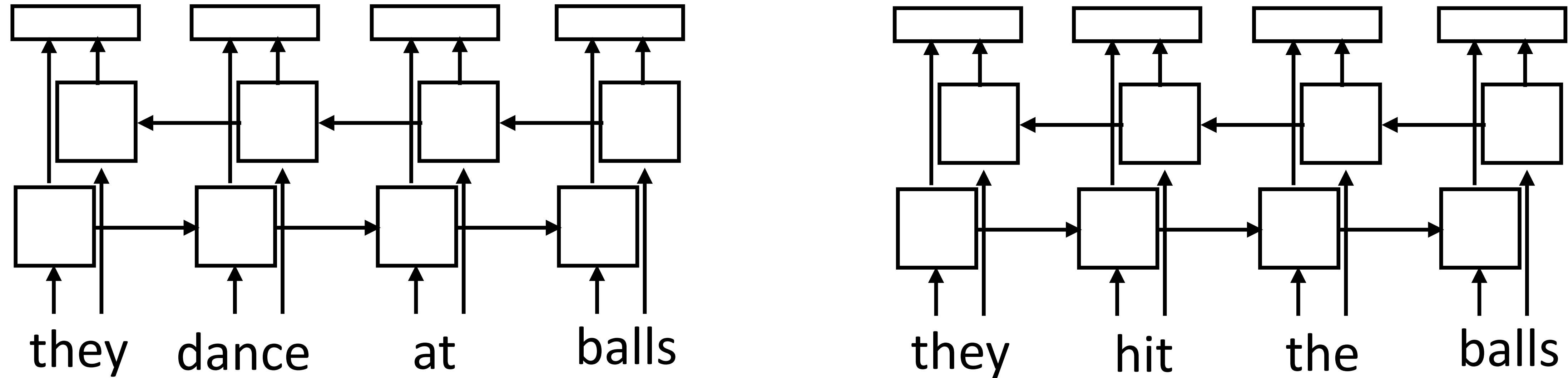
Alan Ritter

(many slides from Greg Durrett)

Pretraining / ELMo

# Recall: 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

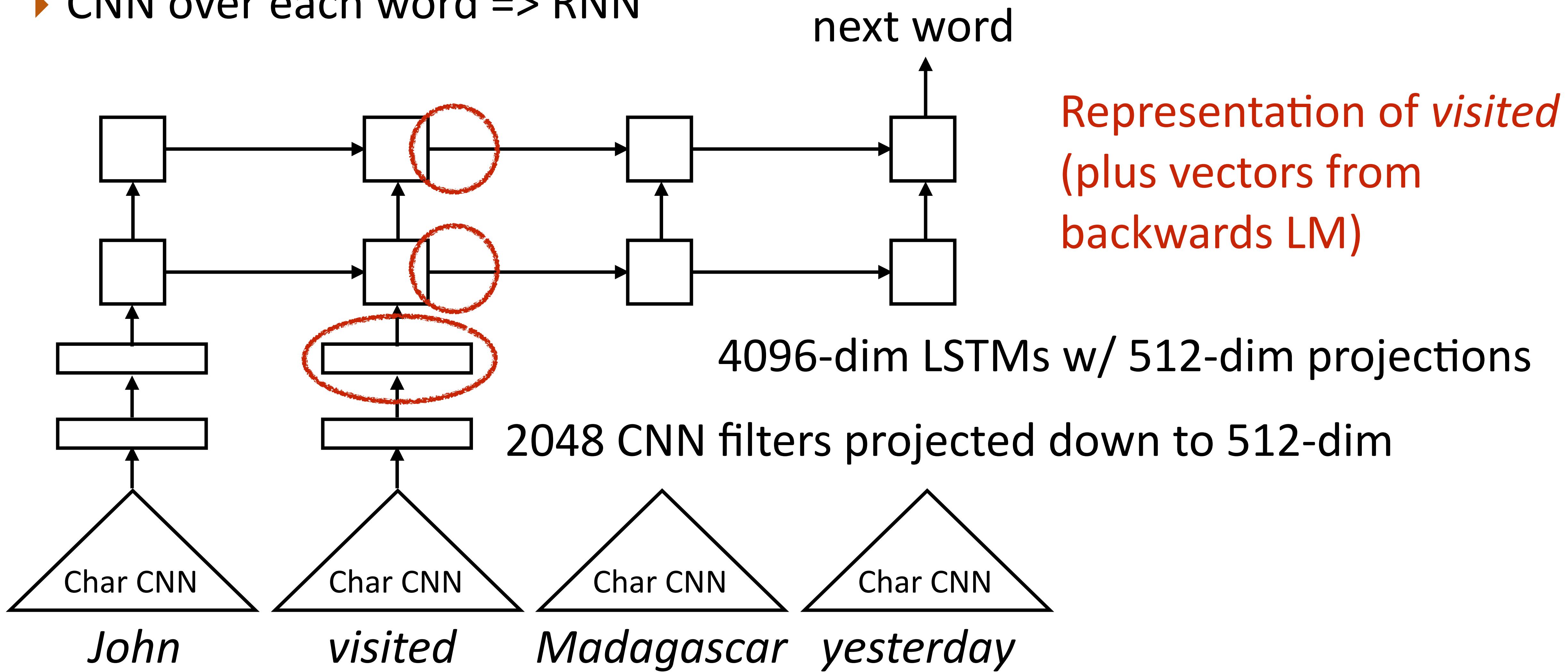
# ELMo

---

- ▶ 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
- ▶ What do we want our LM to look like?

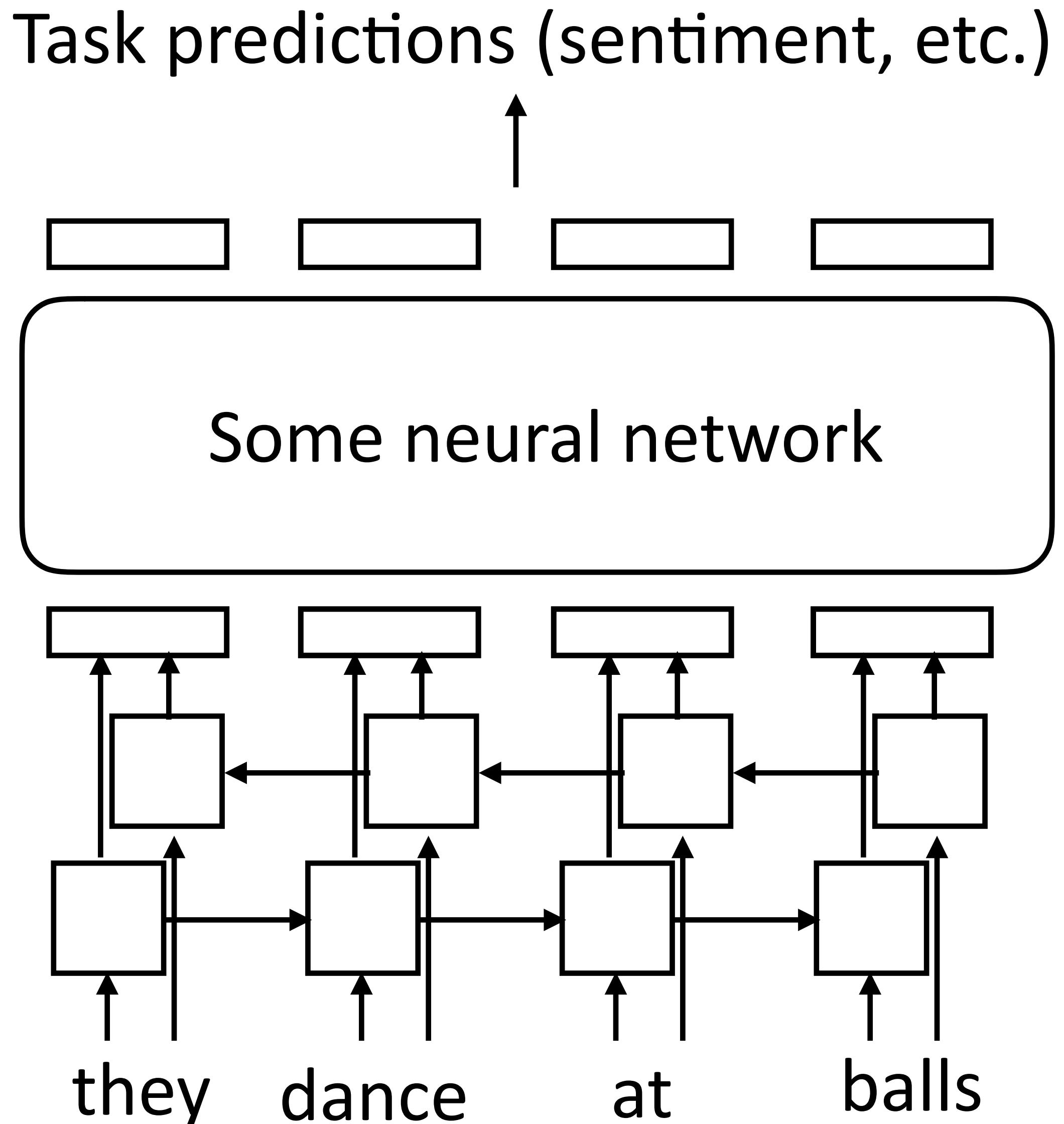
# ELMo

- ▶ 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

# 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	<b>91.8</b>	<b>79.6</b>	<b>86.3</b>	<b>86.1</b>	<b>76.0</b>	<b>75.9</b>
	🔥	<b>91.9</b>	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

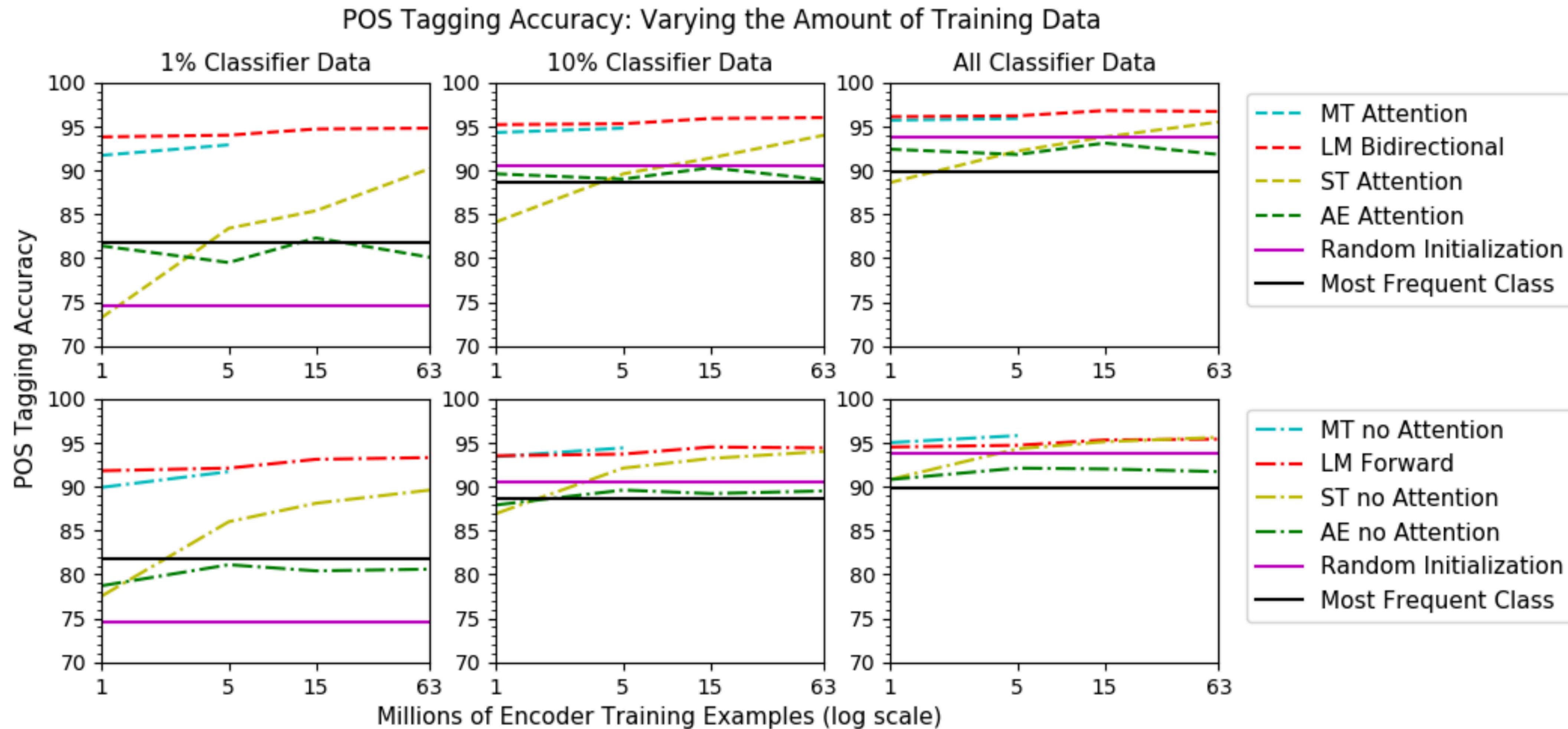
*Context:* My wife refused to allow me to come to Hong Kong when the plague was at its height and –” “Your wife, Johanne? You are married at last ?” Johanne grinned. “Well, when a man gets to my age, he starts to need a few home comforts.

*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, Winograd schema, and much more

# Why is language modeling a good objective?



Zhang and Bowman (2018)

# 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

# Probing ELMo

- ▶ From each layer of the ELMo model, attempt to predict something: POS tags, word senses, etc.
- ▶ Higher accuracy => ELMo is capturing that thing more nicely

Model	F <sub>1</sub>
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	<b>70.1</b>
CoVe, First Layer	59.4
CoVe, Second Layer	64.7
biLM, First layer	67.4
biLM, Second layer	69.0

Table 5: All-words fine grained WSD F<sub>1</sub>. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Model	Acc.
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	<b>97.8</b>
CoVe, First Layer	93.3
CoVe, Second Layer	92.8
biLM, First Layer	97.3
biLM, Second Layer	96.8

Table 6: Test set POS tagging accuracies for PTB. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

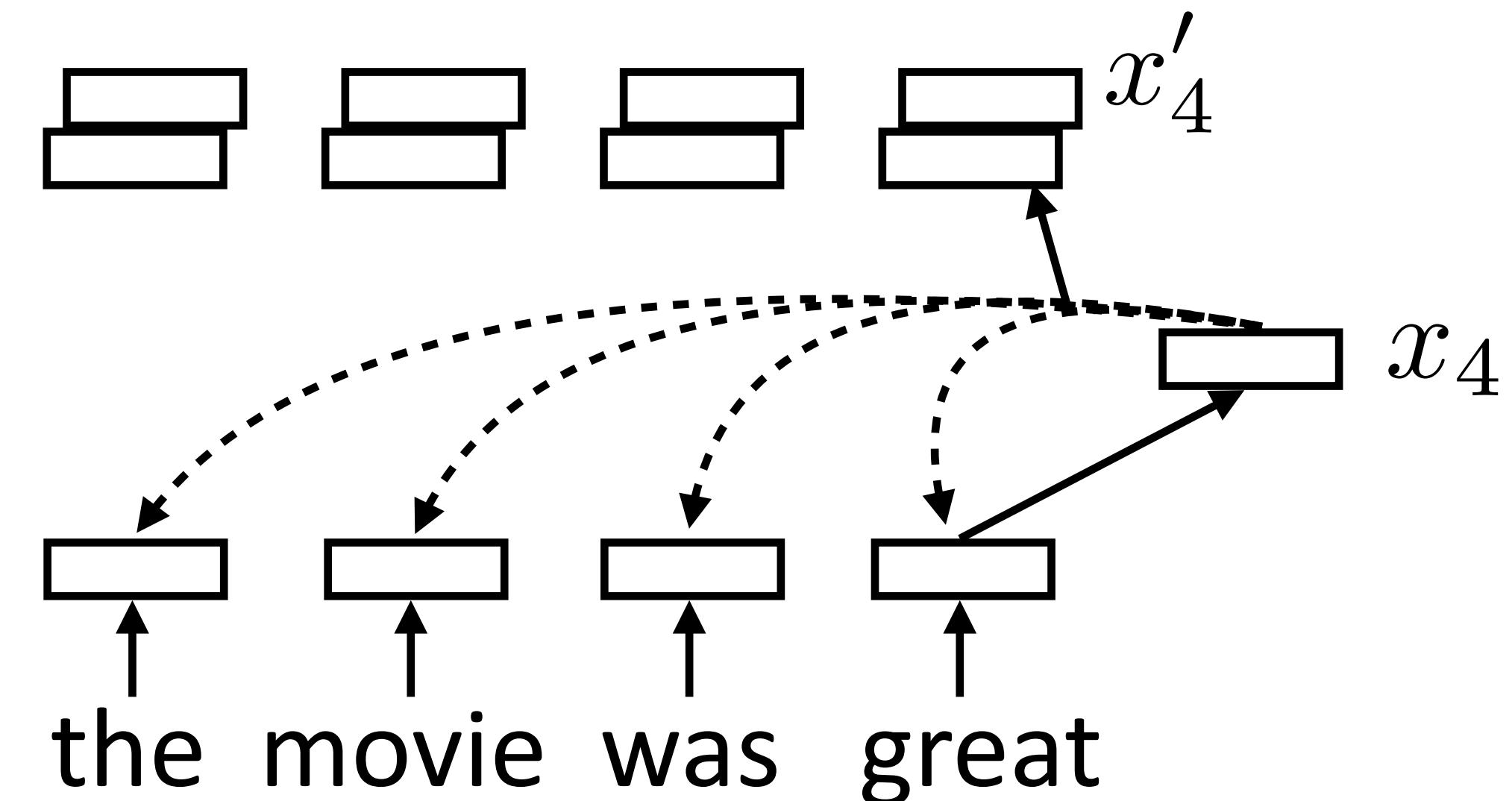
**BERT**

# Recall: Self-Attention

- Each word forms a “query” which then computes attention over each word

$$\alpha_{i,j} = \text{softmax}(x_i^\top x_j) \quad \text{scalar}$$

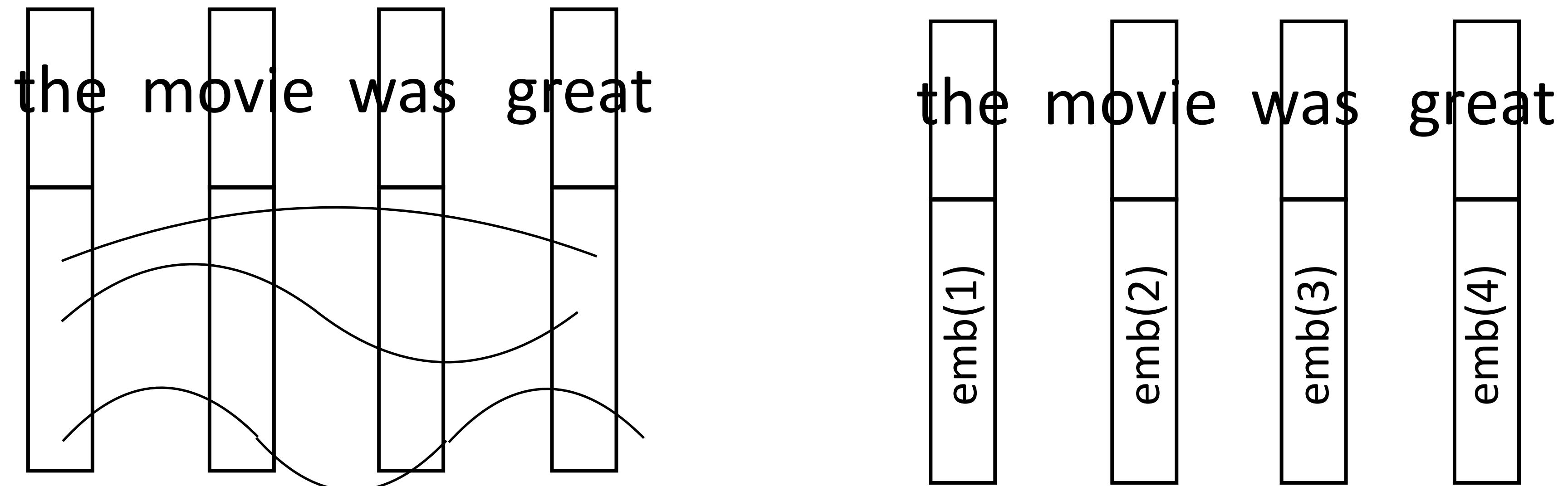
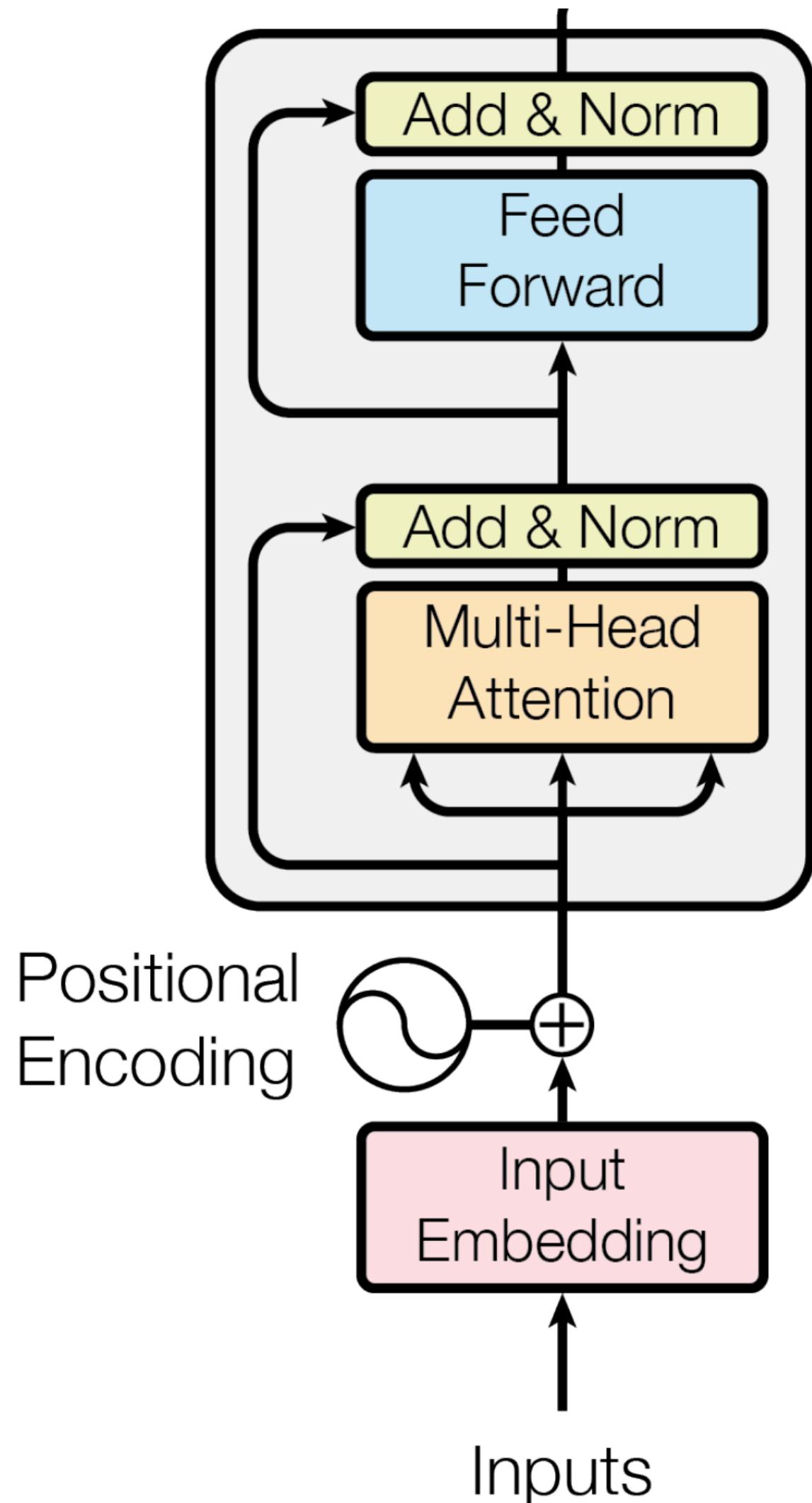
$$x'_i = \sum_{j=1}^n \alpha_{i,j} x_j \quad \text{vector} = \text{sum of scalar} * \text{vector}$$



- Multiple “heads” analogous to different convolutional filters. Use parameters  $W_k$  and  $V_k$  to get different attention values + transform vectors

$$\alpha_{k,i,j} = \text{softmax}(x_i^\top W_k x_j) \quad x'_{k,i} = \sum_{j=1}^n \alpha_{k,i,j} V_k x_j$$

# Recall: Transformers



- ▶ Augment word embedding with position embeddings, each dim is a sine/cosine wave of a different frequency. Closer points = higher dot products
- ▶ Works essentially as well as just encoding position as a one-hot vector

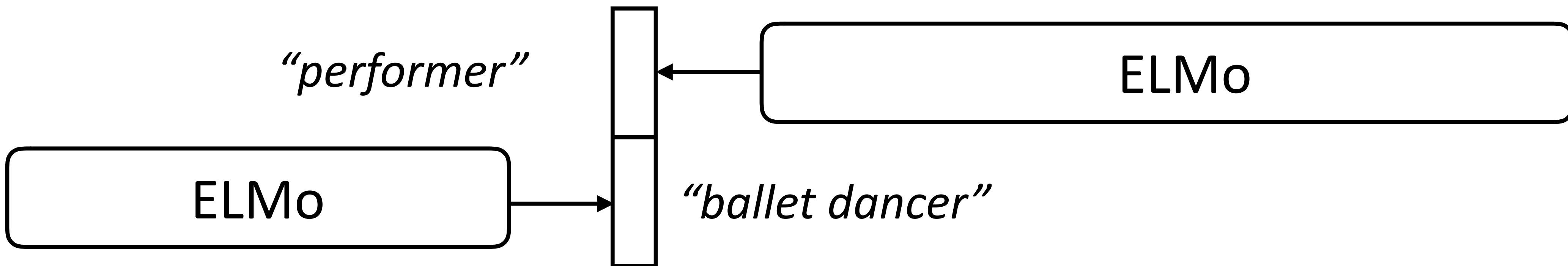
# BERT

---

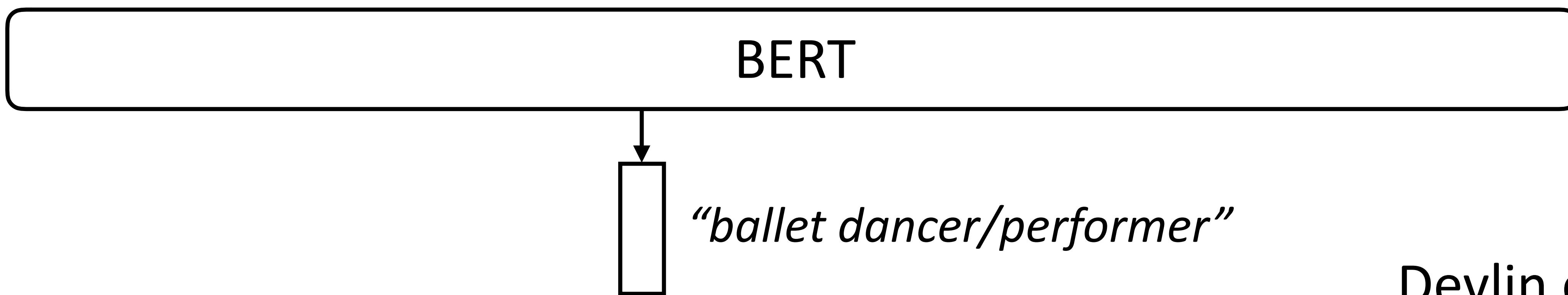
- ▶ AI2 made ELMo in spring 2018, GPT was released in summer 2018, BERT came out October 2018
- ▶ Three major changes compared to ELMo:
  - ▶ Transformers instead of LSTMs (transformers in GPT as well)
  - ▶ Bidirectional <=> Masked LM objective instead of standard LM
  - ▶ Fine-tune instead of freeze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

# BERT

- ▶ ELMo is a unidirectional model (as is GPT): we can concatenate two unidirectional models, but is this the right thing to do?
- ▶ ELMo reprs look at each direction in isolation; BERT looks at them jointly

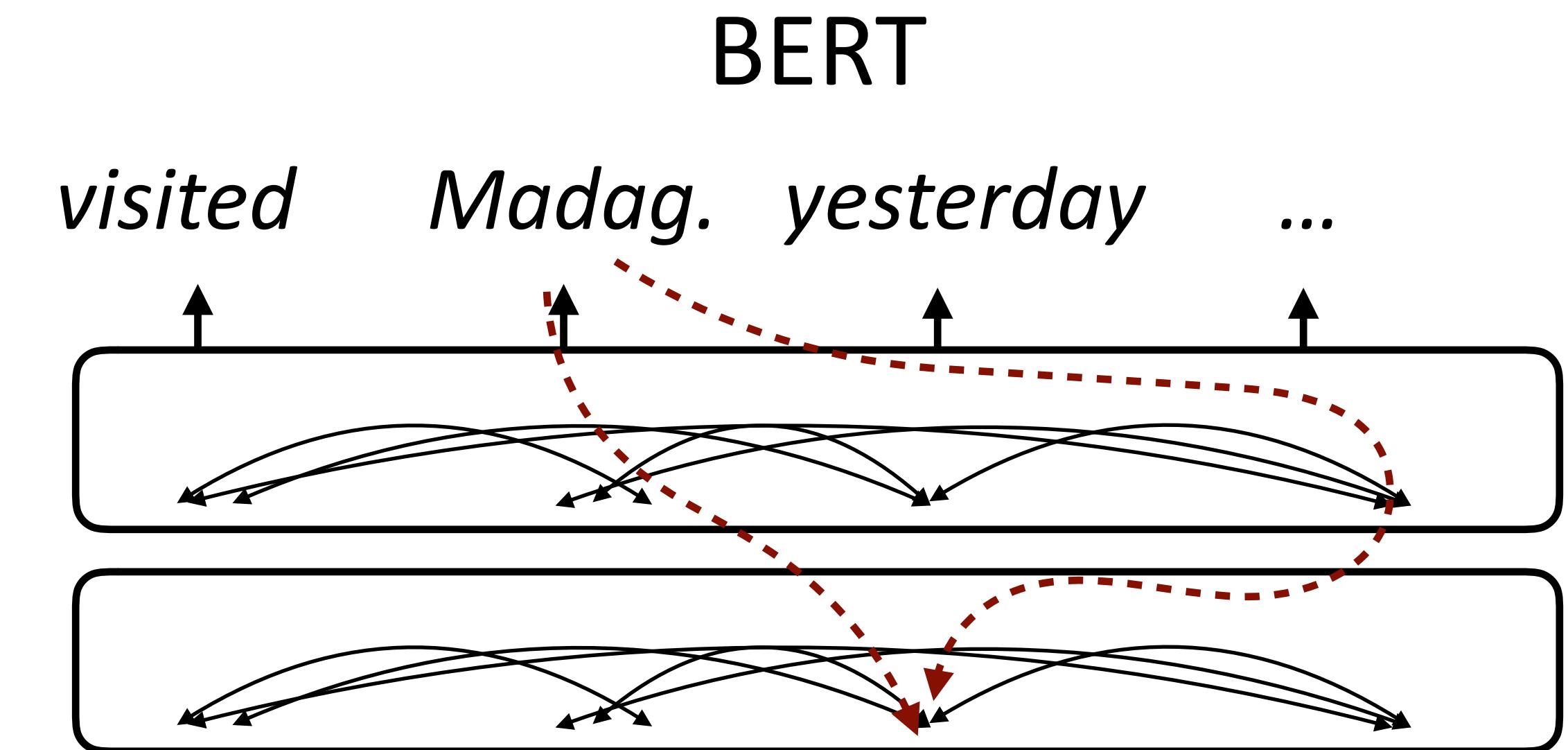
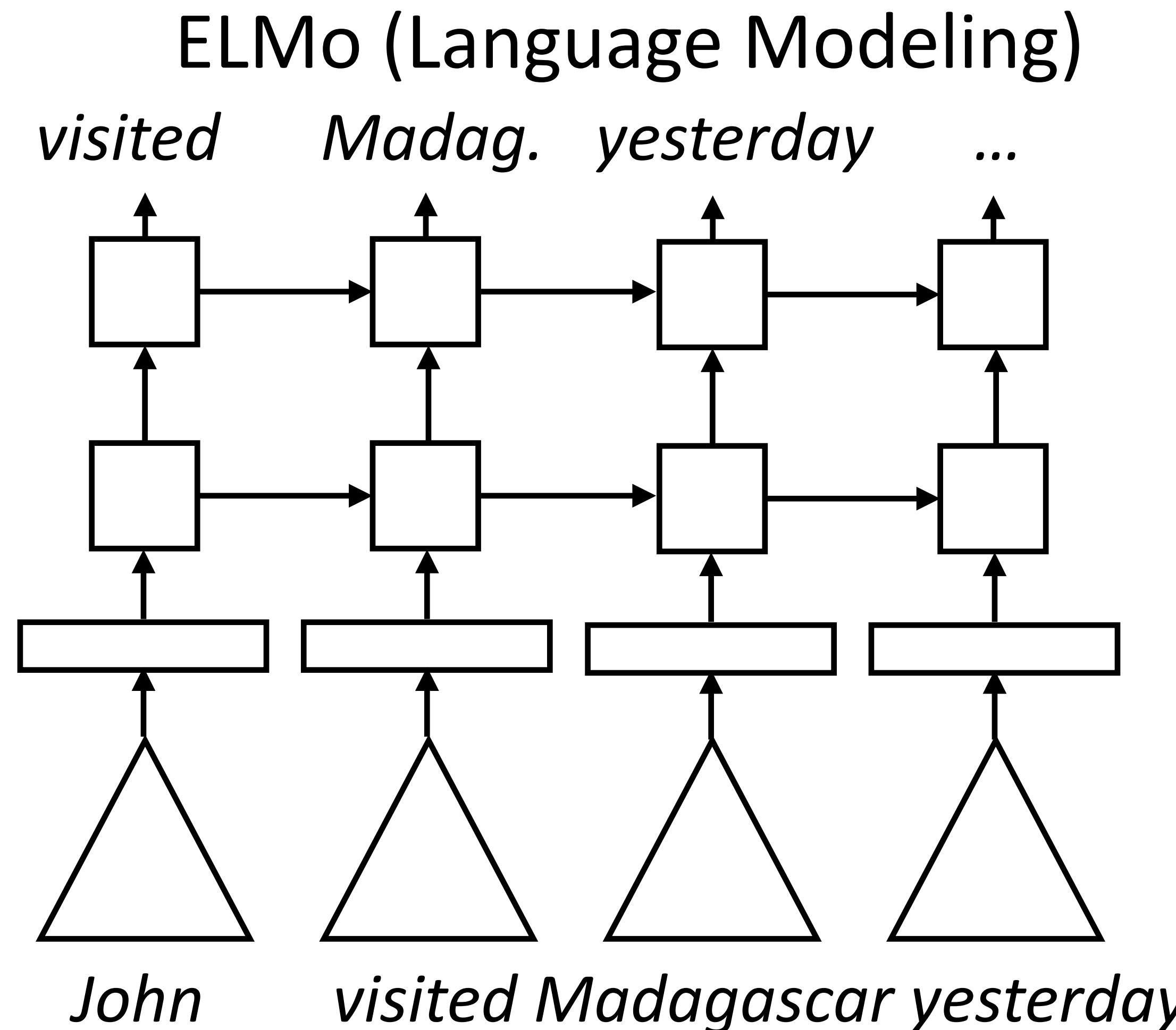


*A stunning ballet dancer, Copeland is one of the best performers to see live.*



# BERT

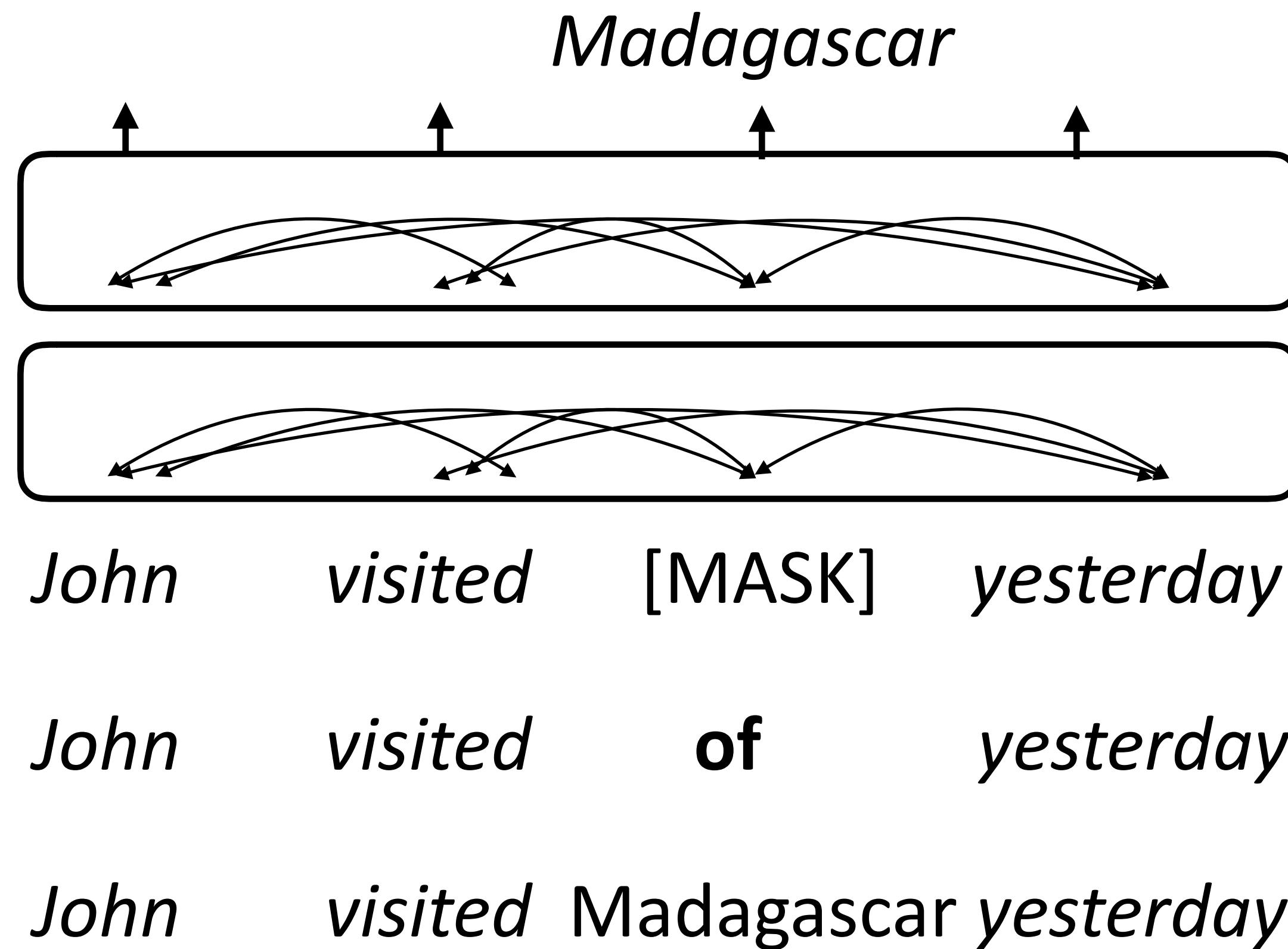
- ▶ How to learn a “deeply bidirectional” model? What happens if we just replace an LSTM with a transformer?



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

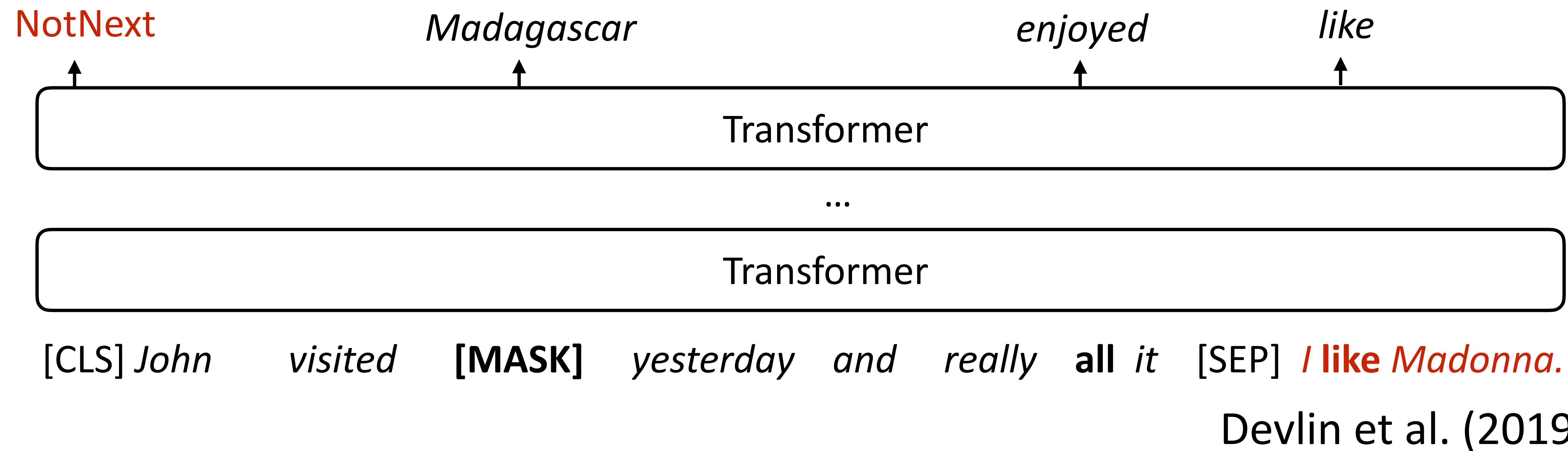
# Masked Language Modeling

- ▶ 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



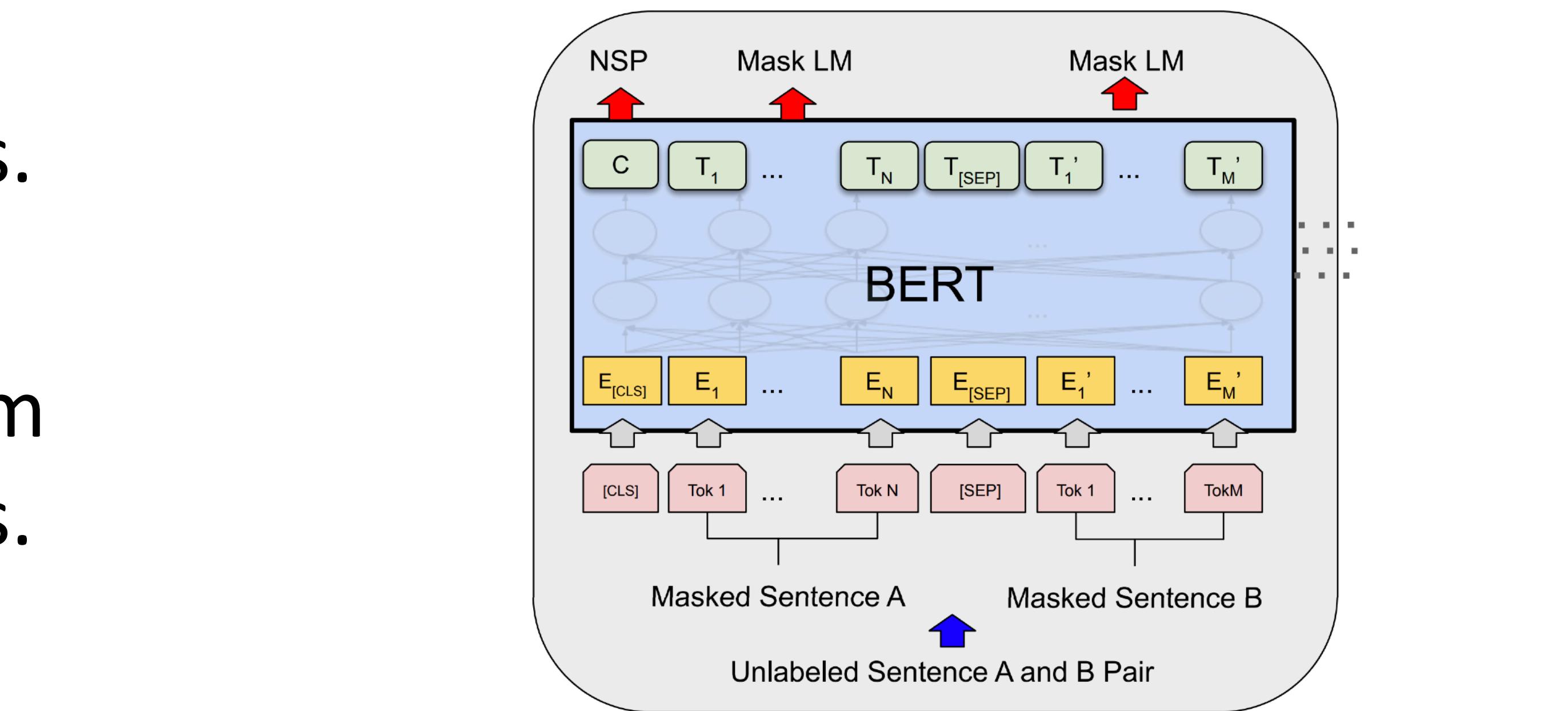
# Next “Sentence” Prediction

- ▶ 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

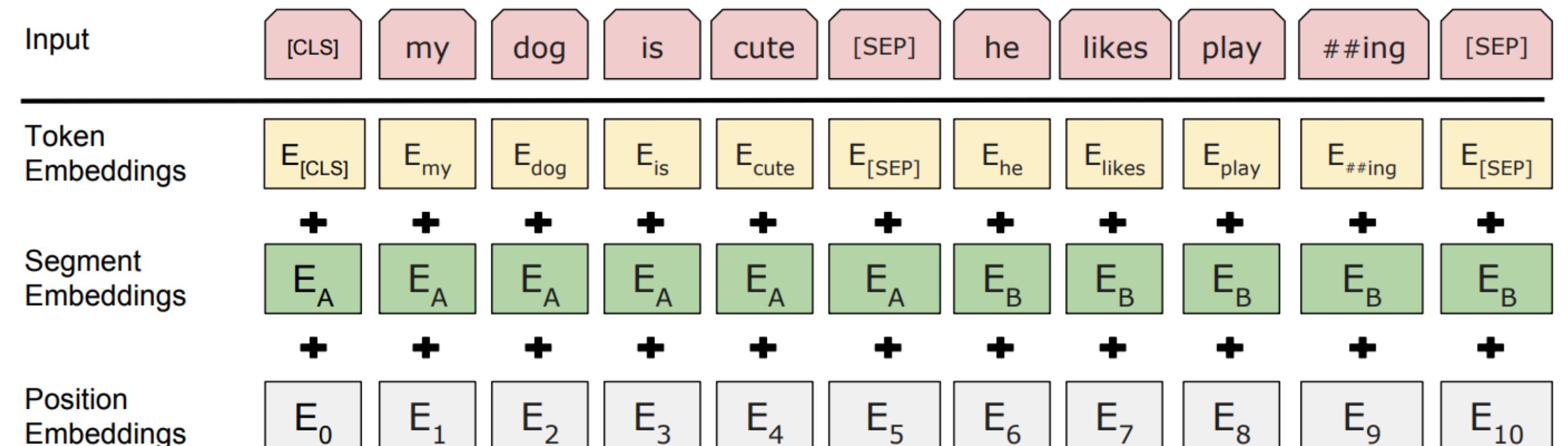


# BERT Architecture

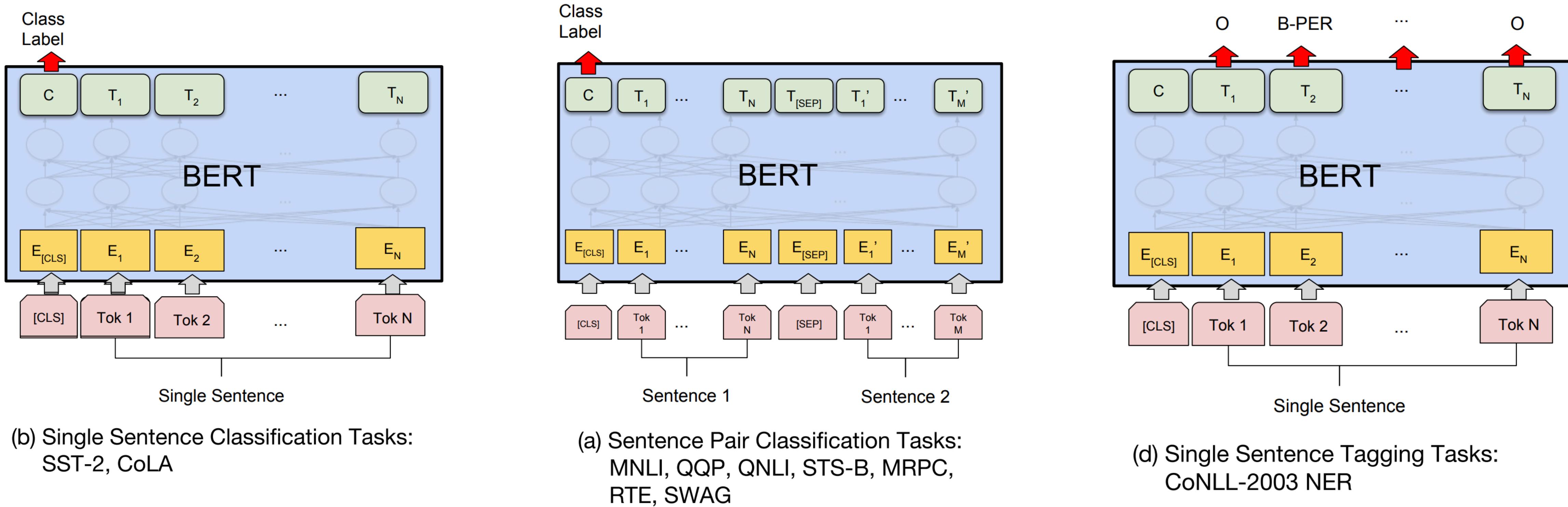
- ▶ BERT Base: 12 layers, 768-dim per wordpiece token, 12 heads.  
Total params = 110M
- ▶ BERT Large: 24 layers, 1024-dim per wordpiece token, 16 heads.  
Total params = 340M



- ▶ Positional embeddings and segment embeddings, 30k word pieces
- ▶ This is the model that gets pre-trained on a large corpus



# What can BERT do?

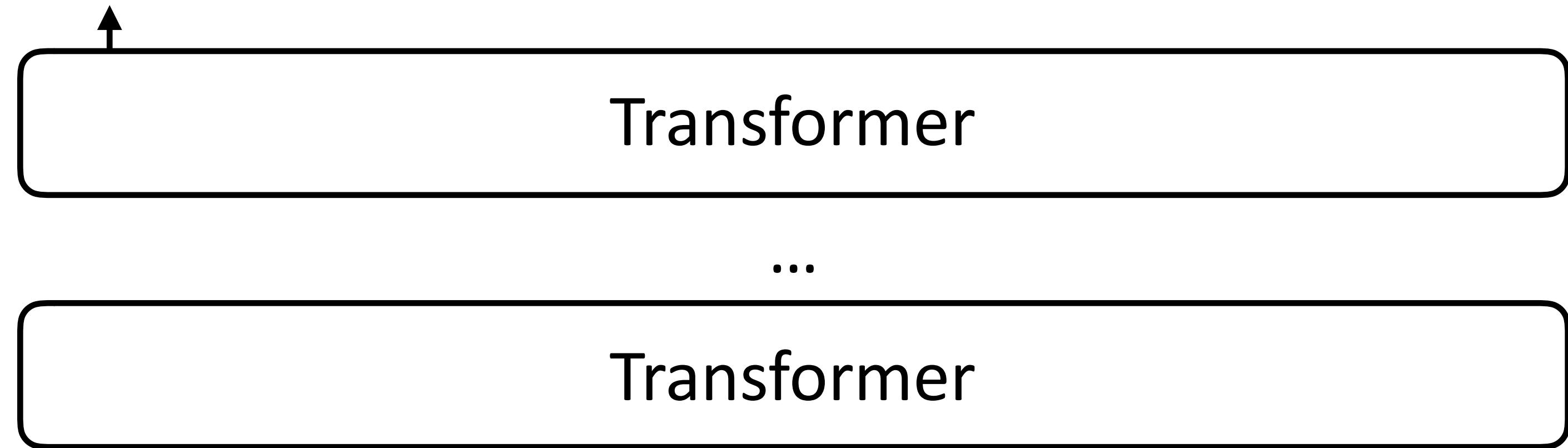


- ▶ CLS token is used to provide classification decisions
- ▶ Sentence pair tasks (entailment): feed both sentences into BERT
- ▶ BERT can also do tagging by predicting tags at each word piece

Devlin et al. (2019)

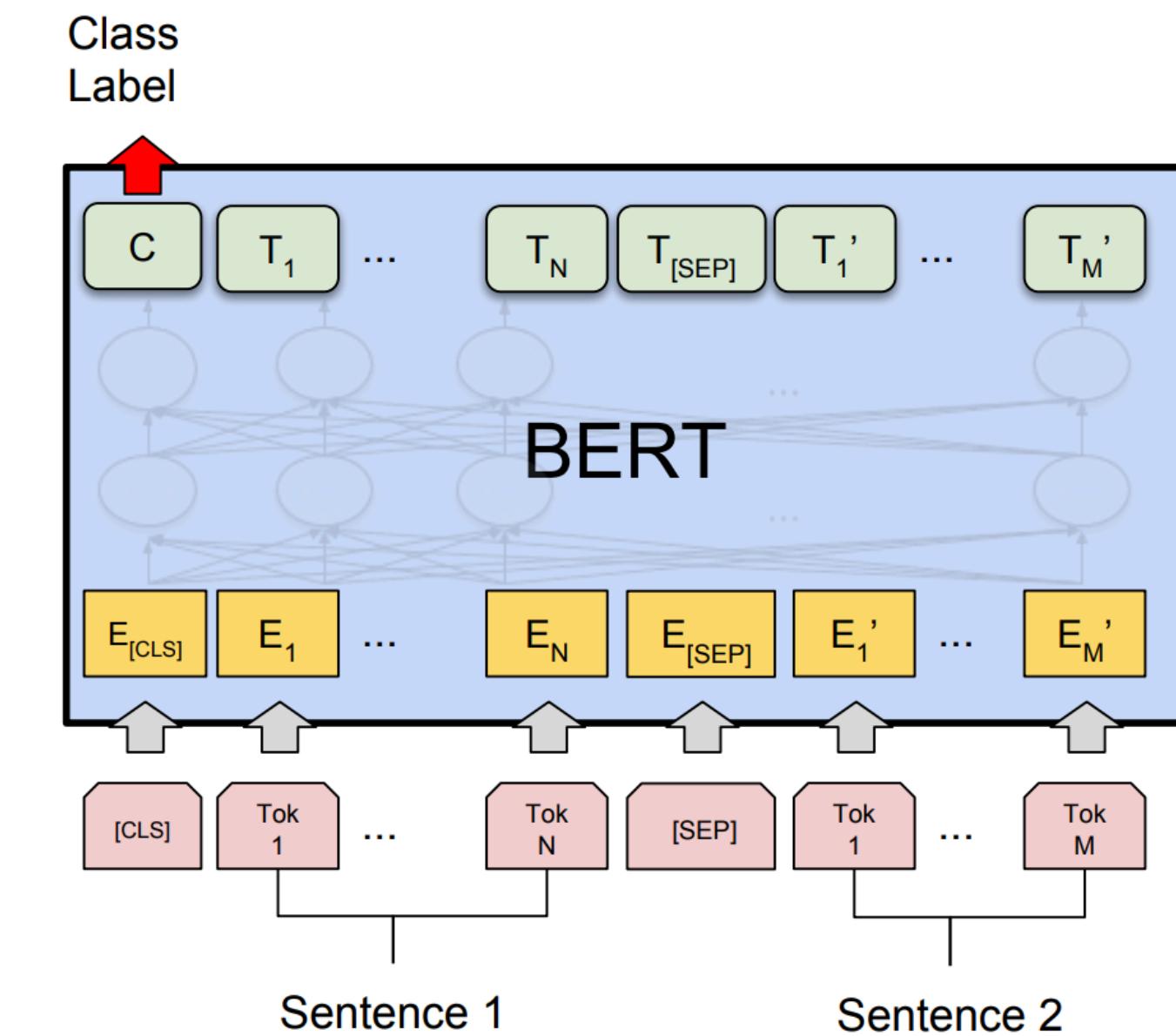
# What can BERT do?

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

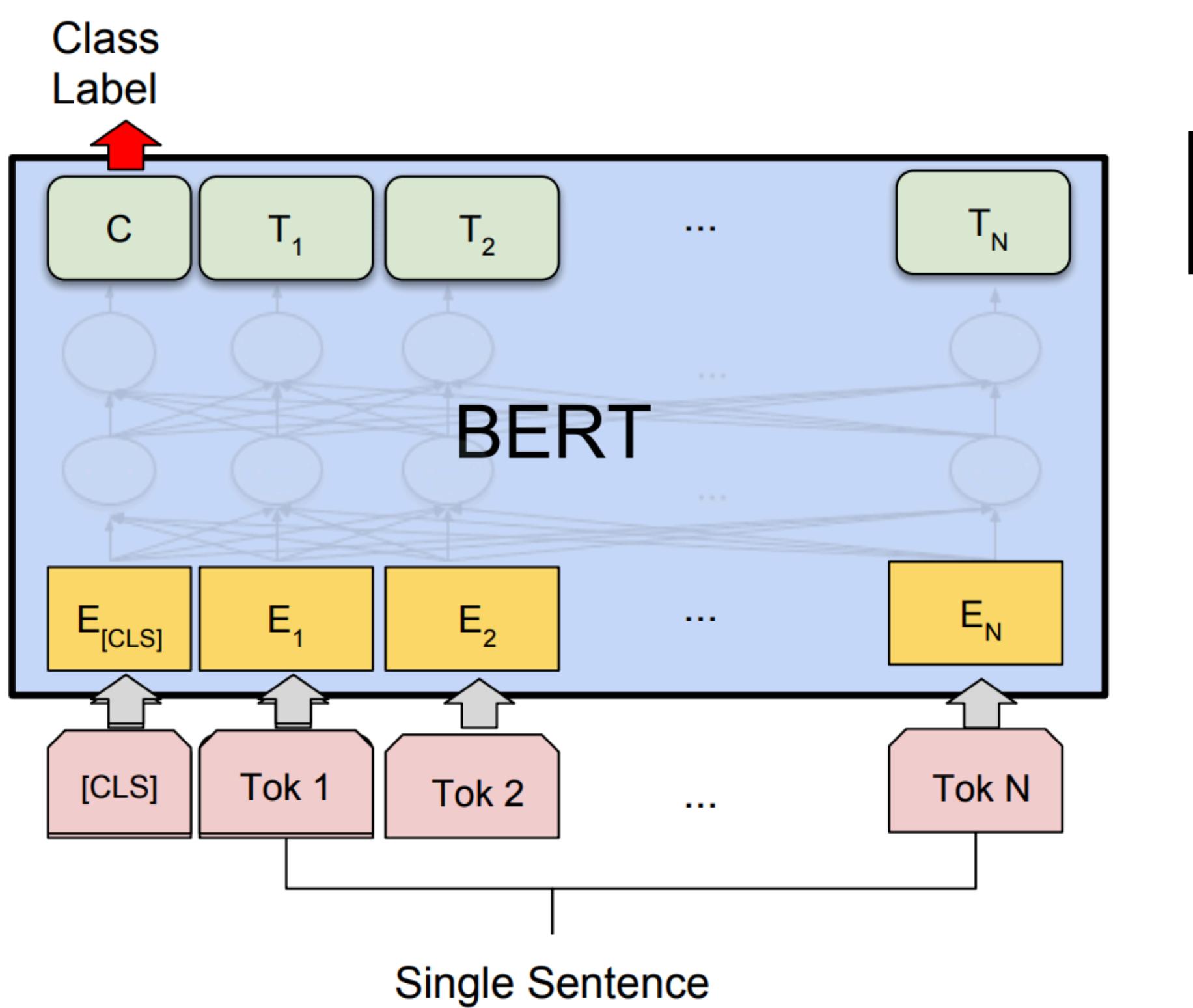
# What can BERT NOT do?

---

- ▶ 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

# 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

Pretraining	Adaptation	NER CoNLL 2003	SA SST-2	Nat. lang. inference MNLI	SICK-E	Semantic textual similarity SICK-R	MRPC	STS-B
Skip-thoughts	❄️	-	81.8	62.9	-	86.6	75.8	71.8
ELMo	❄️	91.7	<b>91.8</b>	<b>79.6</b>	<b>86.3</b>	<b>86.1</b>	<b>76.0</b>	<b>75.9</b>
	🔥	<b>91.9</b>	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	<b>84.6</b>	84.8	86.4	78.1	82.9
	🔥	<b>92.4</b>	<b>93.5</b>	<b>84.6</b>	<b>85.8</b>	<b>88.7</b>	<b>84.8</b>	<b>87.1</b>
	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	<b>1k</b>	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	<b>391k</b>	paraphrase	acc./F1	social QA questions
Inference Tasks					
MNLI	393k	<b>20k</b>	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	<b>146</b>	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 <sub>BASE</sub>	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>91.1</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>81.9</b>

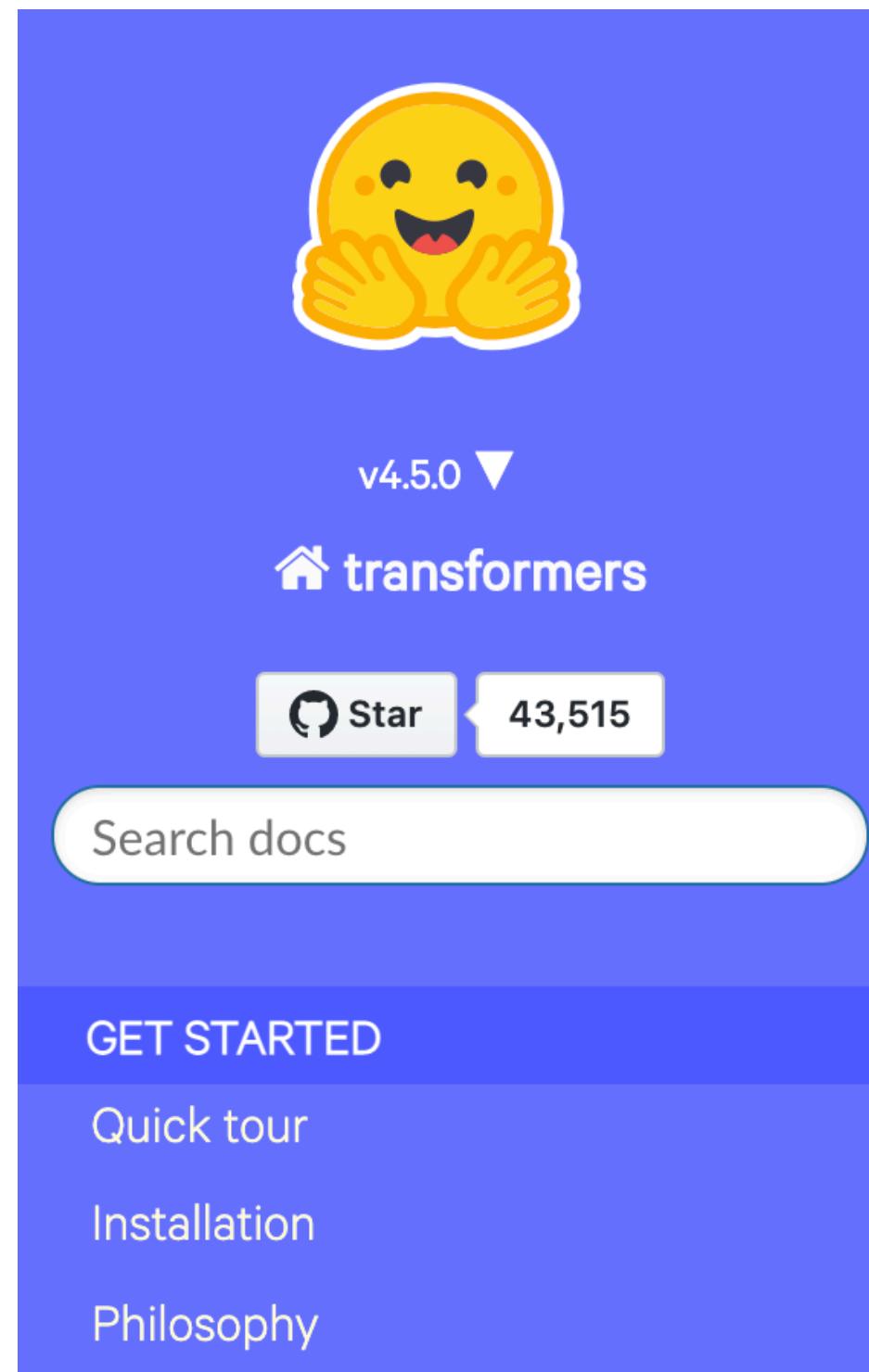
- ▶ Huge improvements over prior work (even compared to ELMo)
- ▶ Effective at “sentence pair” tasks: textual entailment (does sentence A imply sentence B), paraphrase detection

# RoBERTa

- ▶ “Robustly optimized BERT”
- ▶ 160GB of data instead of 16 GB
- ▶ Dynamic masking: standard BERT uses the same MASK scheme for every epoch, RoBERTa recomputes them
- ▶ 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	<b>94.6/89.4</b>	<b>90.2</b>	<b>96.4</b>

BERT <sub>LARGE</sub>	with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7



Docs » Transformers

SIGN IN

MODELS

FORUM

[View page source](#)

# Transformers

State-of-the-art Natural Language Processing for Pytorch and TensorFlow 2.0.

Transformers (formerly known as *pytorch-transformers* and *pytorch-pretrained-bert*) provides general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch.

This is the documentation of our repository [transformers](#).

GPT/GPT2

# OpenAI GPT/GPT2

---

- ▶ “ELMo with transformers” (works better than ELMo)
- ▶ Train a single unidirectional transformer LM on long contexts
- ▶ GPT2: trained on 40GB of text collected from upvoted links from reddit
- ▶ 1.5B parameters — by far the largest of these models trained as of March 2019
- ▶ Because it's a language model, we can **generate** from it

Parameters	Layers	$d_{model}$
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

# OpenAI GPT2

---

**SYSTEM PROMPT  
(HUMAN-WRITTEN)**

*Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today.*

**MODEL COMPLETION  
(MACHINE-WRITTEN,  
SECOND TRY)**

The 19-year-old singer was caught on camera being escorted out of the store by security guards.

The singer was wearing a black hoodie with the label 'Blurred Lines' on the front and 'Fashion Police' on the back.

Scroll down for video

Shoplifting: Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today (pictured)

The singer was wearing a black hoodie with the label 'Blurred Lines' on the front and 'Fashion Police' on the back

The singer was also wearing a pair of black-rimmed glasses, a black jacket, black jeans and black sandals.

She was carrying a pair of black and white striped gloves and a small black bag.

slide credit:  
OpenAI

# Open Questions

---

- 1) How novel is the stuff being generated? (Is it just doing nearest neighbors on a large corpus?)
- 2) How do we understand and distill what is learned in this model?
- 3) How do we harness these priors for conditional generation tasks (summarization, generate a report of a basketball game, etc.)
- 4) Is this technology dangerous? (OpenAI has only released 774M param model, not 1.5B yet)

# Grover

- ▶ Sample from a large language model conditioned on a domain, date, authors, and headline
- ▶ Humans rank Grover-generated propaganda as more realistic than real “fake news”
- ▶ Fine-tuned Grover can detect Grover propaganda easily — authors argue for releasing it for this reason
- ▶ NOTE: Not a GAN, discriminator trained separately from the generator

Discriminator size	Unpaired Accuracy			Paired Accuracy		
	Generator size	Generator size	Generator size	Generator size	Generator size	Generator size
Chance	50.0	50.0	50.0	50.0	50.0	50.0
1.5B	GROVER-Mega	<b>92.0</b>	<b>98.5</b>	<b>99.8</b>	<b>97.4</b>	<b>100.0</b>
	GROVER-Large	<b>80.8</b>	<b>91.2</b>	<b>98.4</b>	<b>89.0</b>	<b>96.9</b>
	BERT-Large	73.1	75.9	97.5	84.1	91.5
355M	GPT2	70.1	78.0	90.3	78.8	87.0
	GROVER-Base	<b>70.1</b>	<b>80.0</b>	<b>89.2</b>	77.5	88.2
	BERT-Base	67.2	76.6	84.1	<b>80.0</b>	<b>89.5</b>
124M	GPT2	66.2	71.9	83.5	72.5	79.6
	FastText	63.8	65.6	69.7	65.9	69.0
11M						74.4

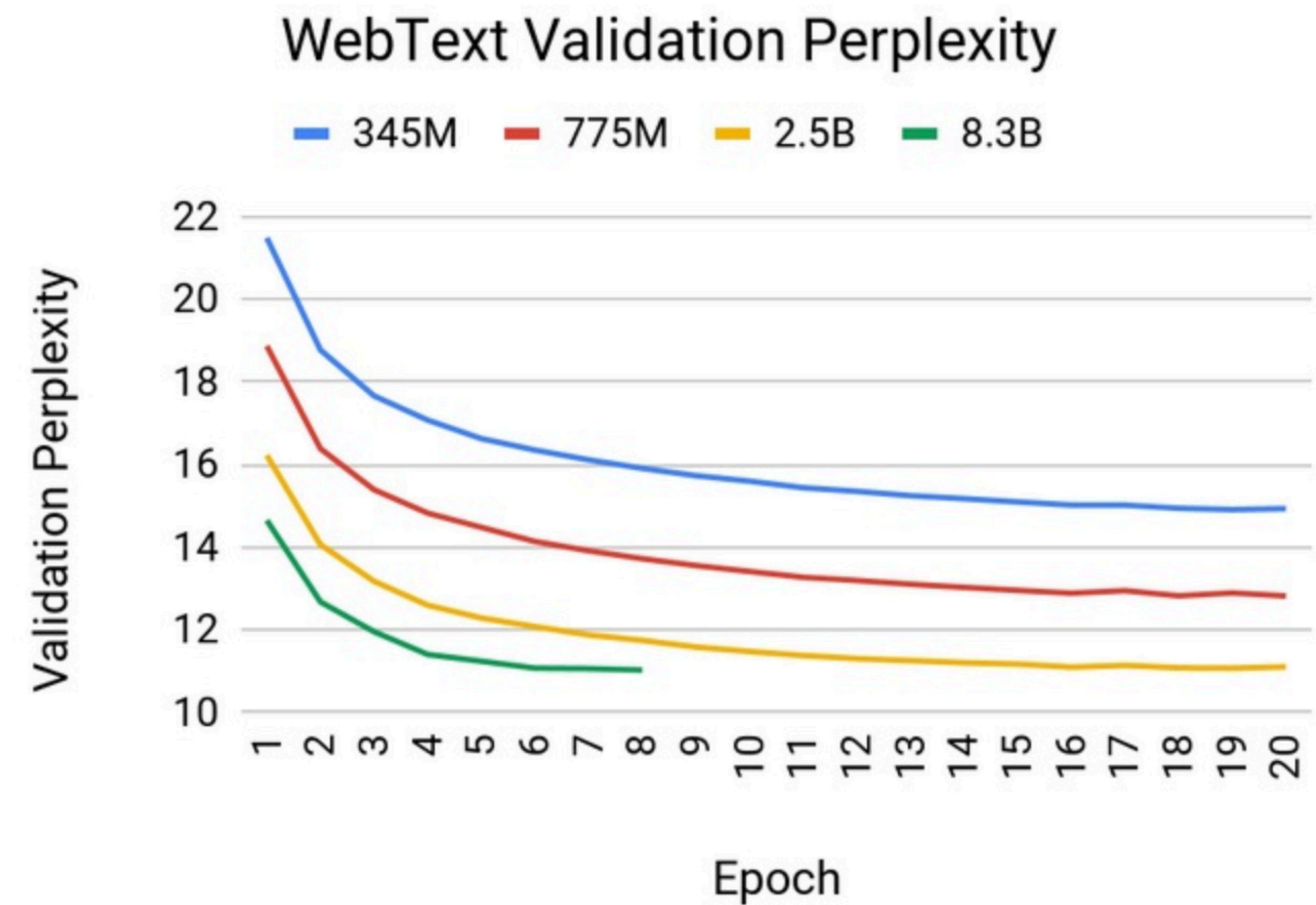
# Pre-Training Cost (with Google/AWS)

---

- ▶ BERT: Base \$500, Large \$7000
- ▶ Grover-MEGA: \$25,000
- ▶ XLNet (BERT variant): \$30,000 – \$60,000 (unclear)
- ▶ This is for a single pre-training run...developing new pre-training techniques may require many runs
- ▶ *Fine-tuning* these models can typically be done with a single GPU (but may take 1-3 days for medium-sized datasets)

# Pushing the Limits

- ▶ NVIDIA: trained 8.3B parameter GPT model (5.6x the size of GPT-2)
- ▶ Arguably these models are still underfit: larger models still get better held-out perplexities



NVIDIA blog (Narasimhan, August 2019)

# Google T5

---

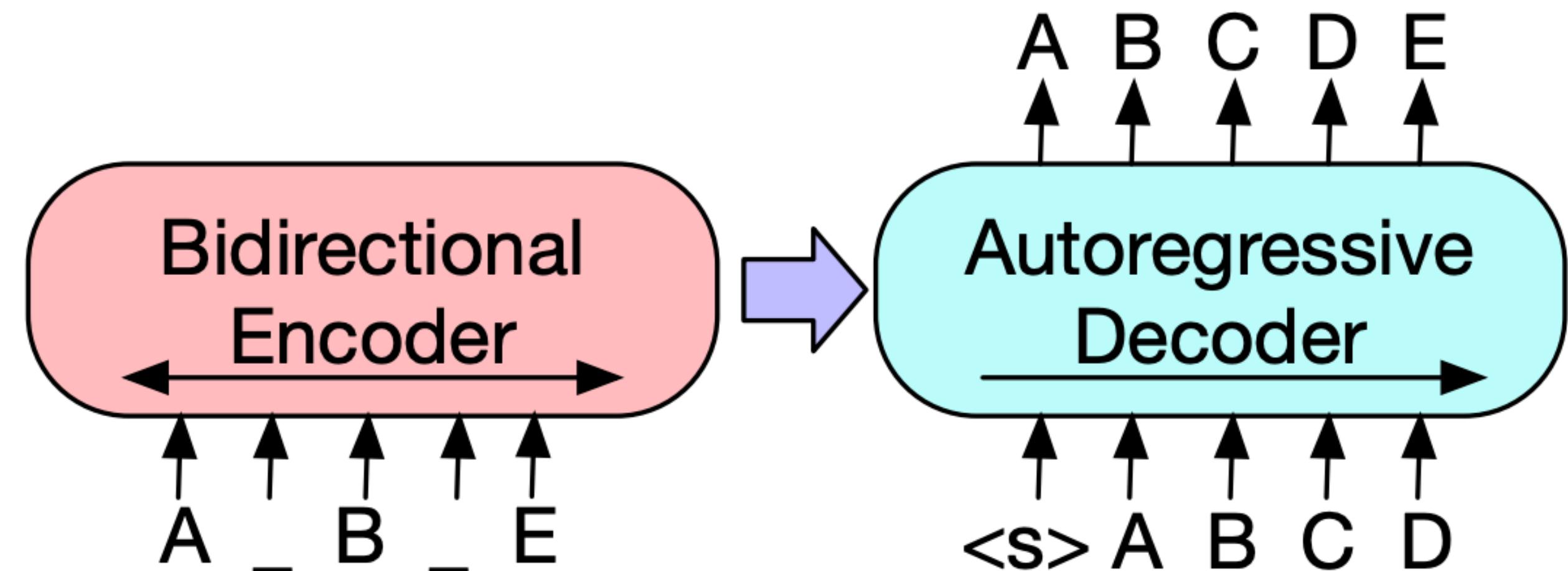
Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full dataset	0	<b>83.28</b>	<b>19.24</b>	<b>80.88</b>	<b>71.36</b>	<b>26.98</b>	<b>39.82</b>	<b>27.65</b>
$2^{29}$	64	<b>82.87</b>	<b>19.19</b>	<b>80.97</b>	<b>72.03</b>	<b>26.83</b>	<b>39.74</b>	<b>27.63</b>
$2^{27}$	256	82.62	<b>19.20</b>	79.78	69.97	<b>27.02</b>	<b>39.71</b>	27.33
$2^{25}$	1,024	79.55	18.57	76.27	64.76	26.38	39.56	26.80
$2^{23}$	4,096	76.34	18.33	70.92	59.29	26.37	38.84	25.81

- ▶ Colossal Cleaned Common Crawl: 750 GB of text
- ▶ We still haven't hit the limit of bigger data being useful

# BART

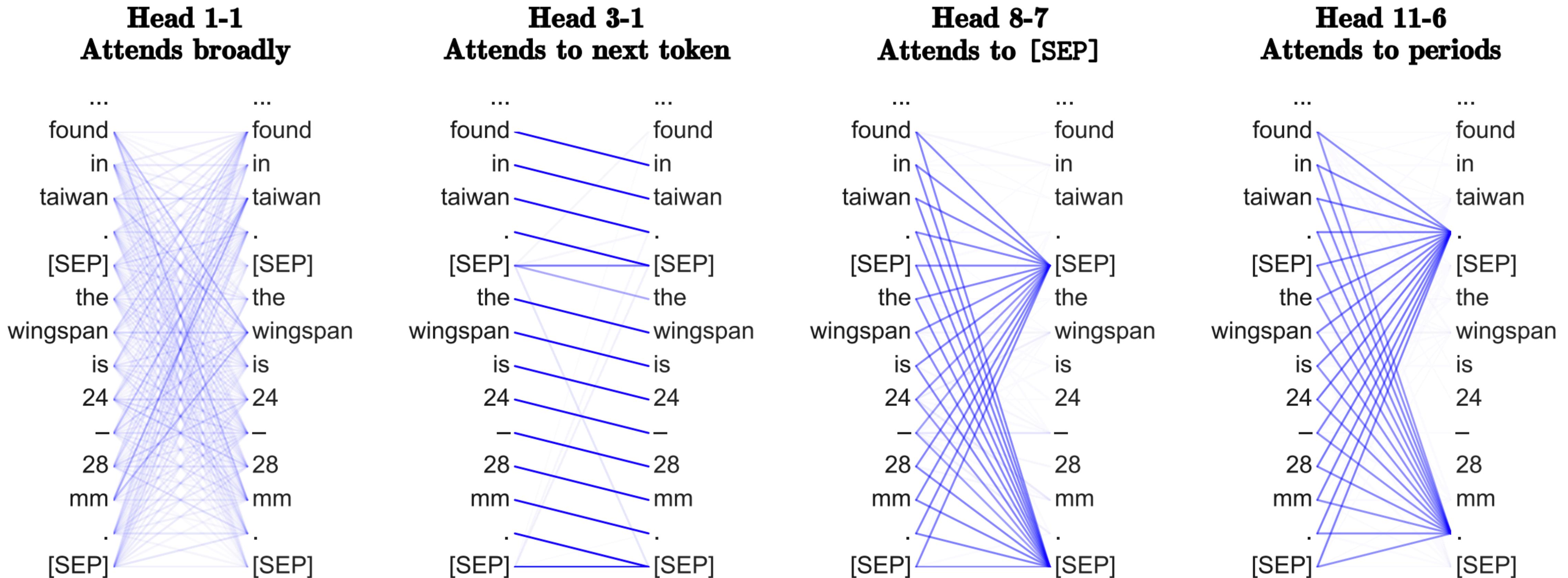
---

- ▶ Sequence-to-sequence BERT variant: permute/make/delete tokens, then predict full sequence autoregressively
- ▶ For downstream tasks: feed document into both encoder + decoder, use decoder hidden state as output
- ▶ Good results on dialogue, summarization tasks



# Analysis

# What does BERT learn?



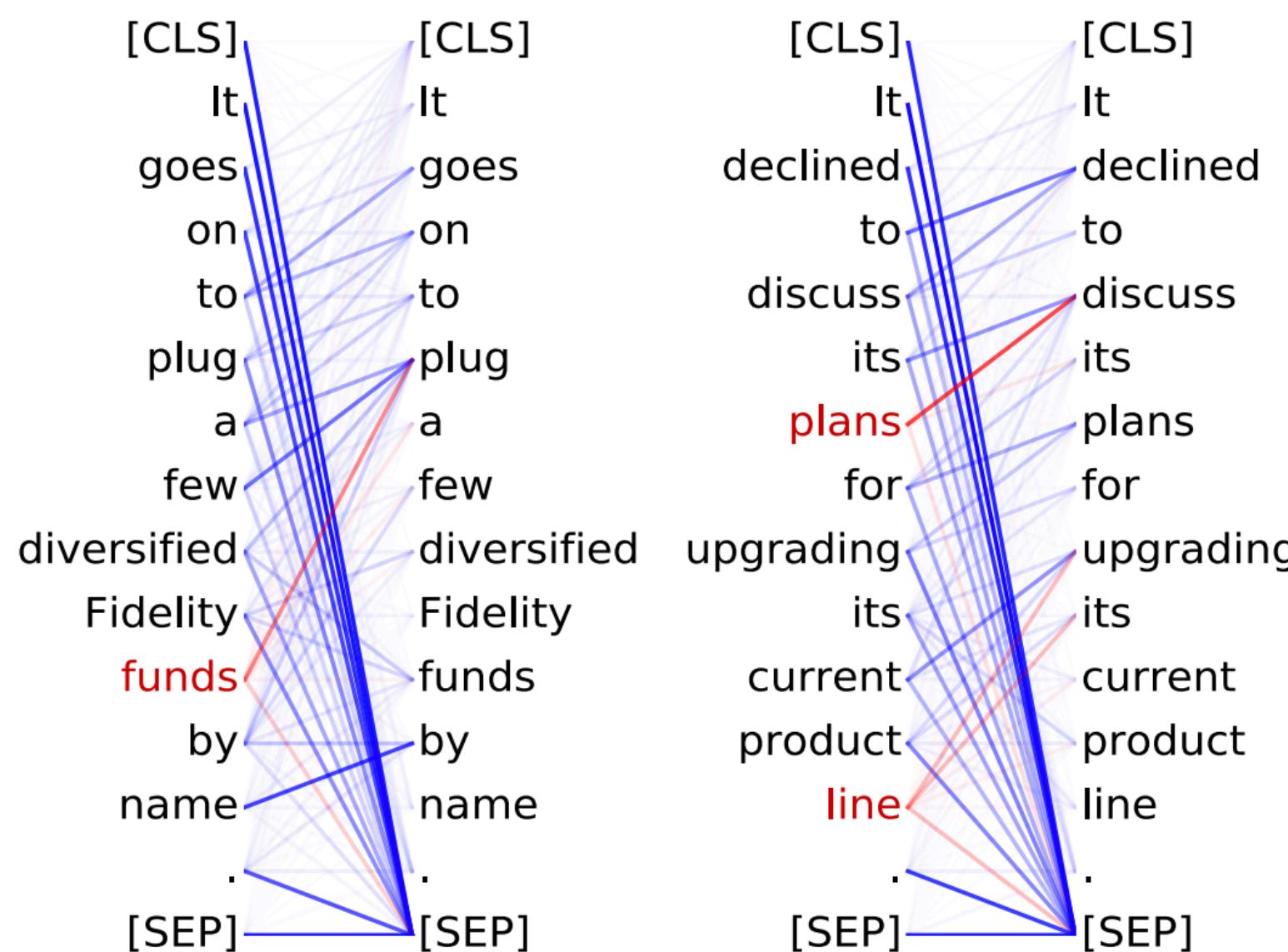
- ▶ Heads on transformers learn interesting and diverse things: content heads (attend based on content), positional heads (based on position), etc.

Clark et al. (2019)

# What does BERT learn?

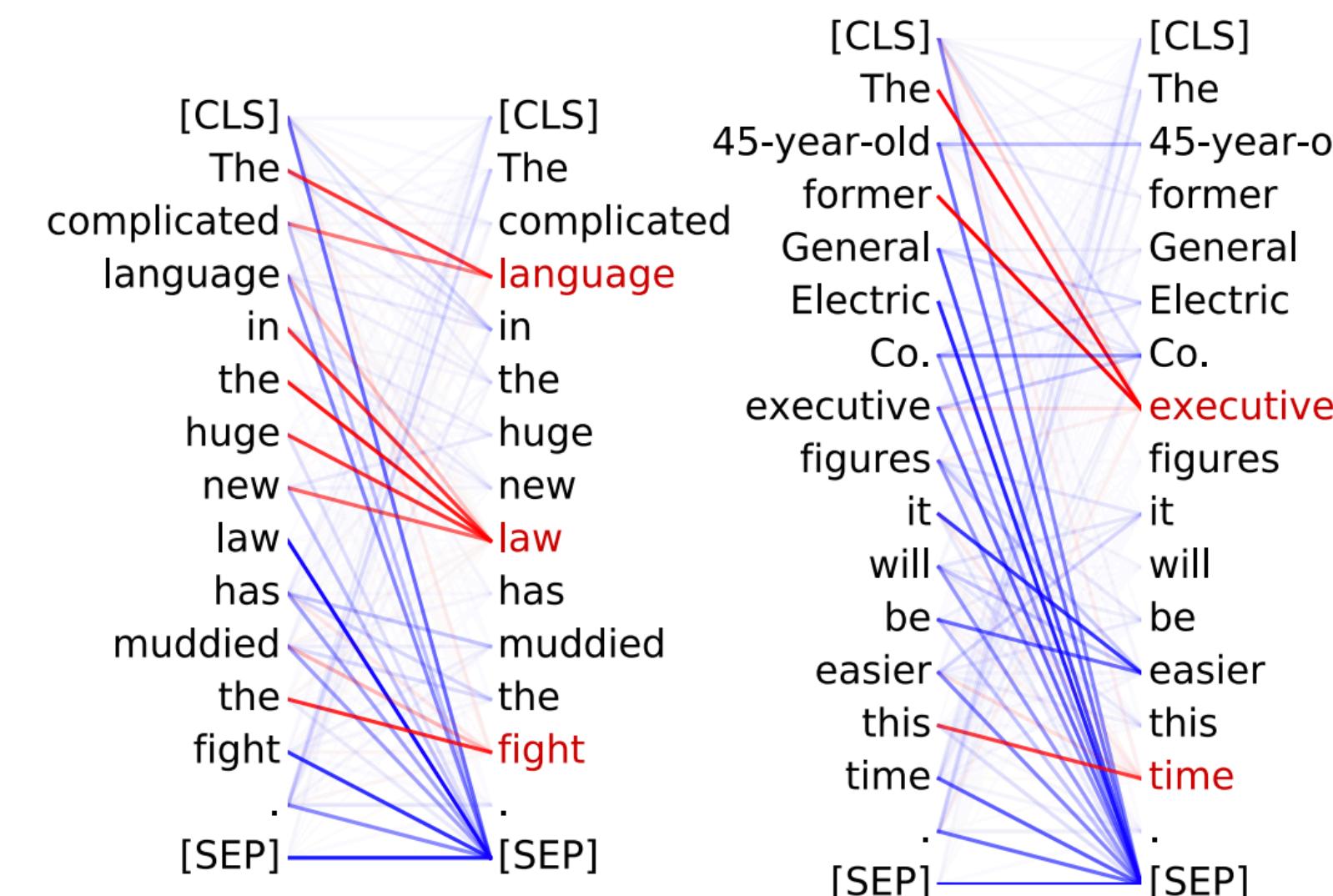
## Head 8-10

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



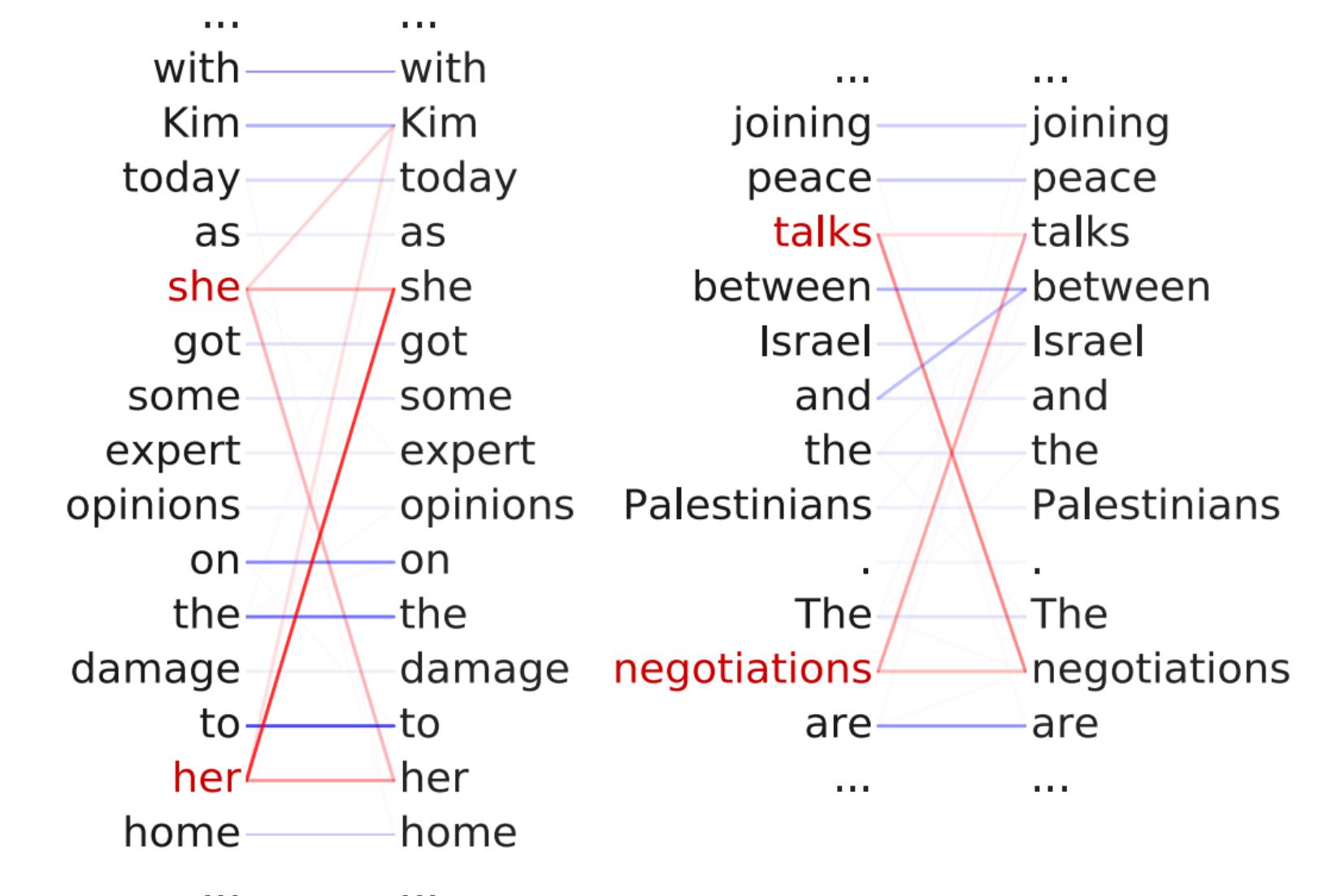
## Head 8-11

- 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

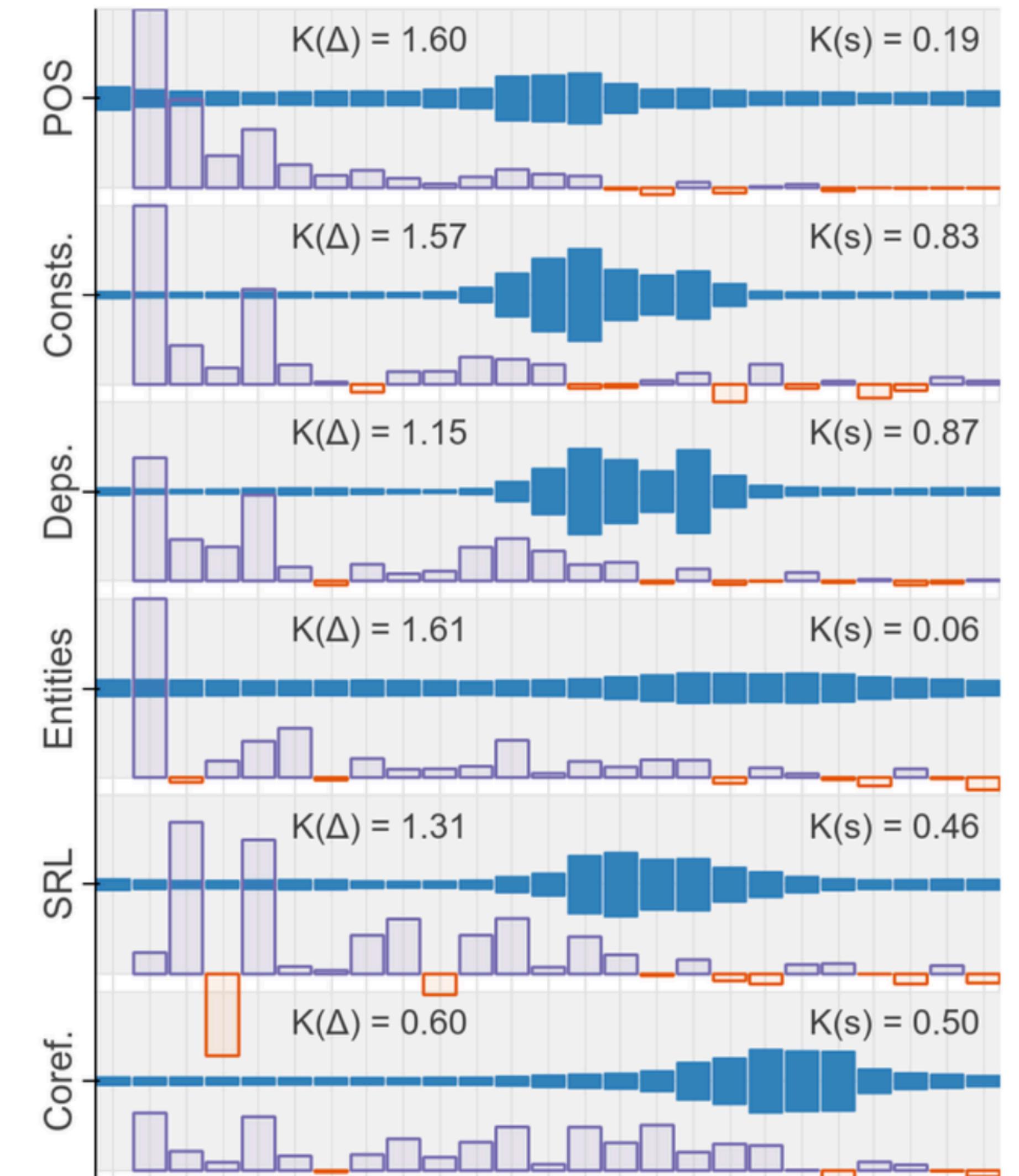
# Probing BERT

- ▶ 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  $\tau$

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

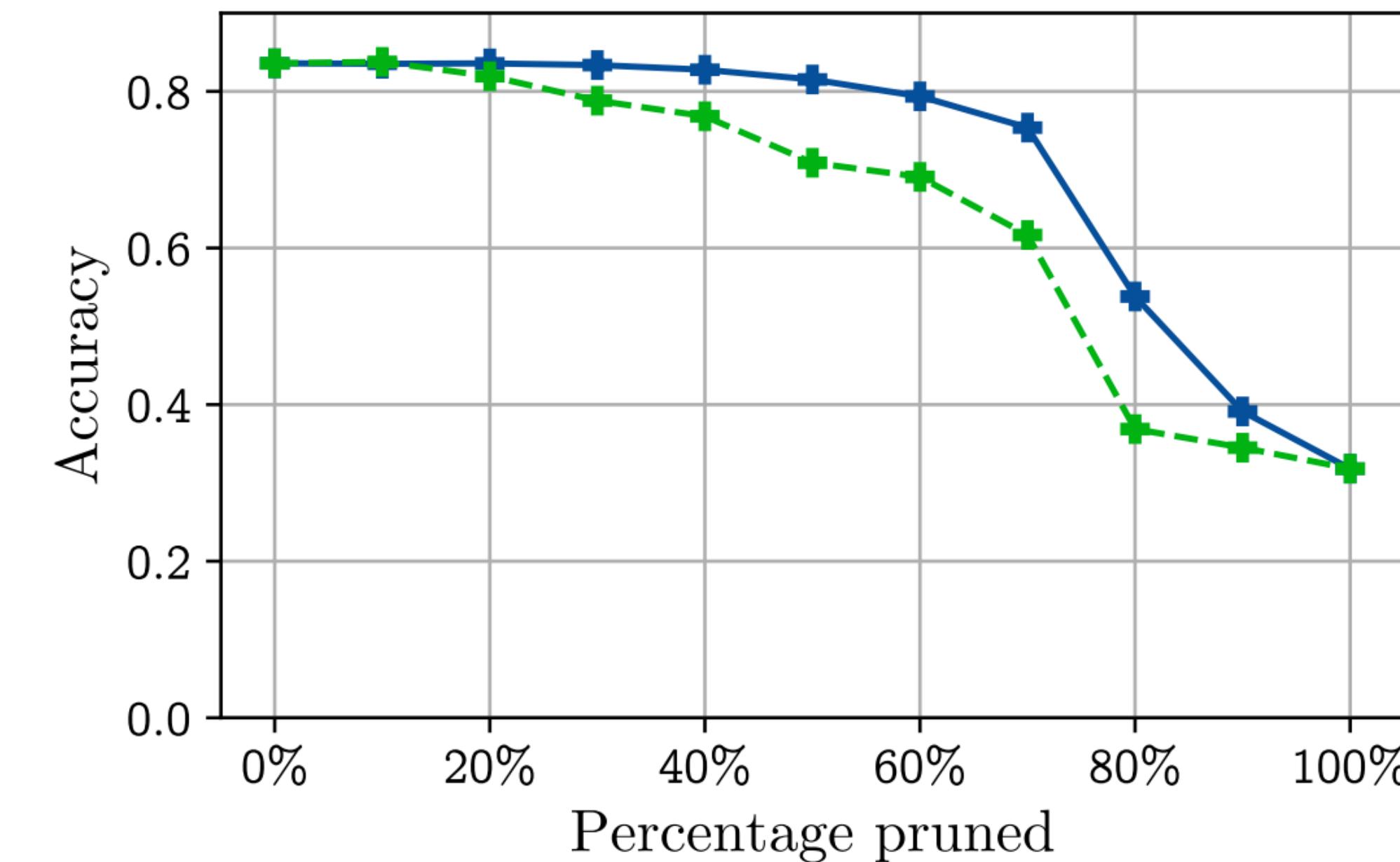


Tenney et al. (2019)

# Compressing BERT

---

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



(b) Evolution of accuracy on the MultiNLI-matched validation set when heads are pruned from BERT according to  $I_h$  (solid blue) and accuracy difference (dashed green).

# Open Questions

---

- ▶ BERT-based systems are state-of-the-art for nearly every major text analysis task
- ▶ These techniques are here to stay, unclear what form will win out
- ▶ Role of academia vs. industry: no major pretrained model has come purely from academia
- ▶ Cost/carbon footprint: a single model costs \$10,000+ to train (though this cost should come down)