

CS371N: Natural Language Processing

Lecture 12: Pre-training, BERT

Greg Durrett



TEXAS

The University of Texas at Austin





Announcements

- ▶ A3 due in one week
- ▶ Tuesday's lecture **on Zoom** (Greg at COLM)
- ▶ Midterm in 3 weeks



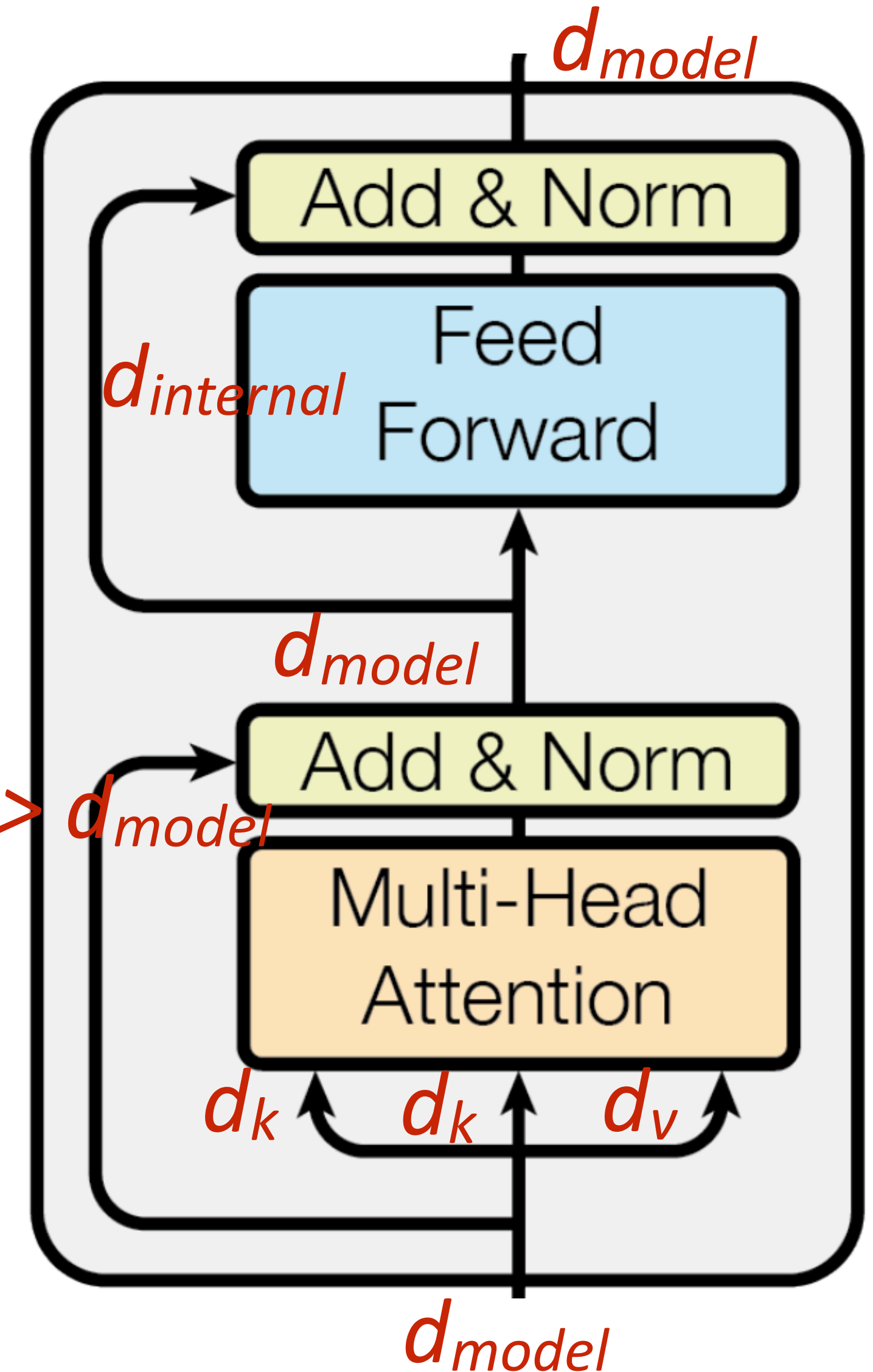
Recap: Transformers

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad d_v \rightarrow d_{model}$$

Query $Q = EW^Q$ Values $V = EW^V$ Keys $K = EW^K$

Position encoding: $E = E + \text{enc}(\text{index})$





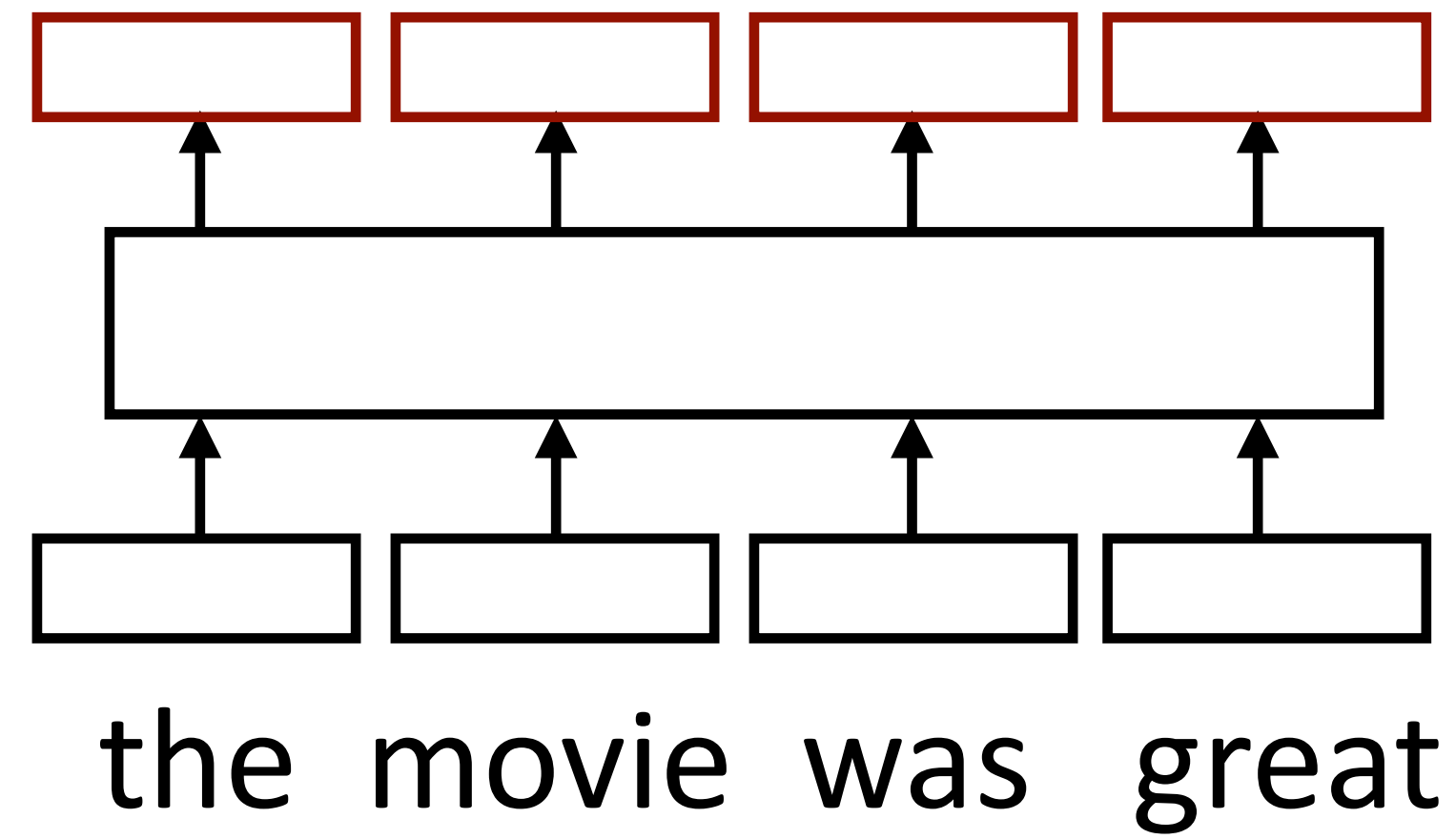
Today

- ▶ Transformer Language Modeling
- ▶ ELMo
- ▶ BERT
- ▶ BERT results
- ▶ Subword tokenization (if time)

Transformer Language Modeling



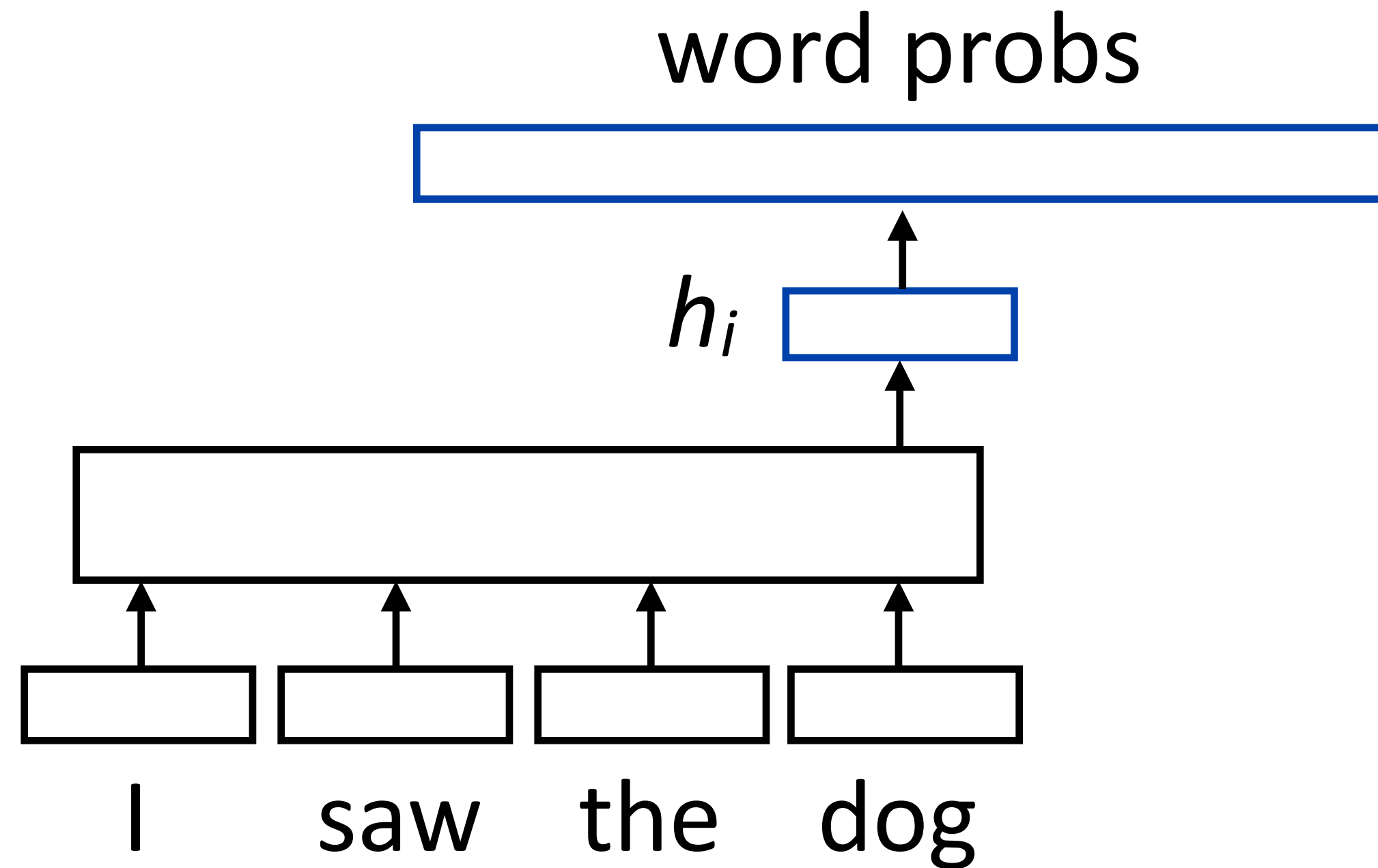
What do Transformers produce?



- ▶ **Encoding of each word** — can pass this to another layer to make a prediction (like predicting the next word for language modeling)
- ▶ Like RNNs, Transformers can be viewed as a transformation of a sequence of vectors into a sequence of context-dependent vectors



Transformer Language Modeling

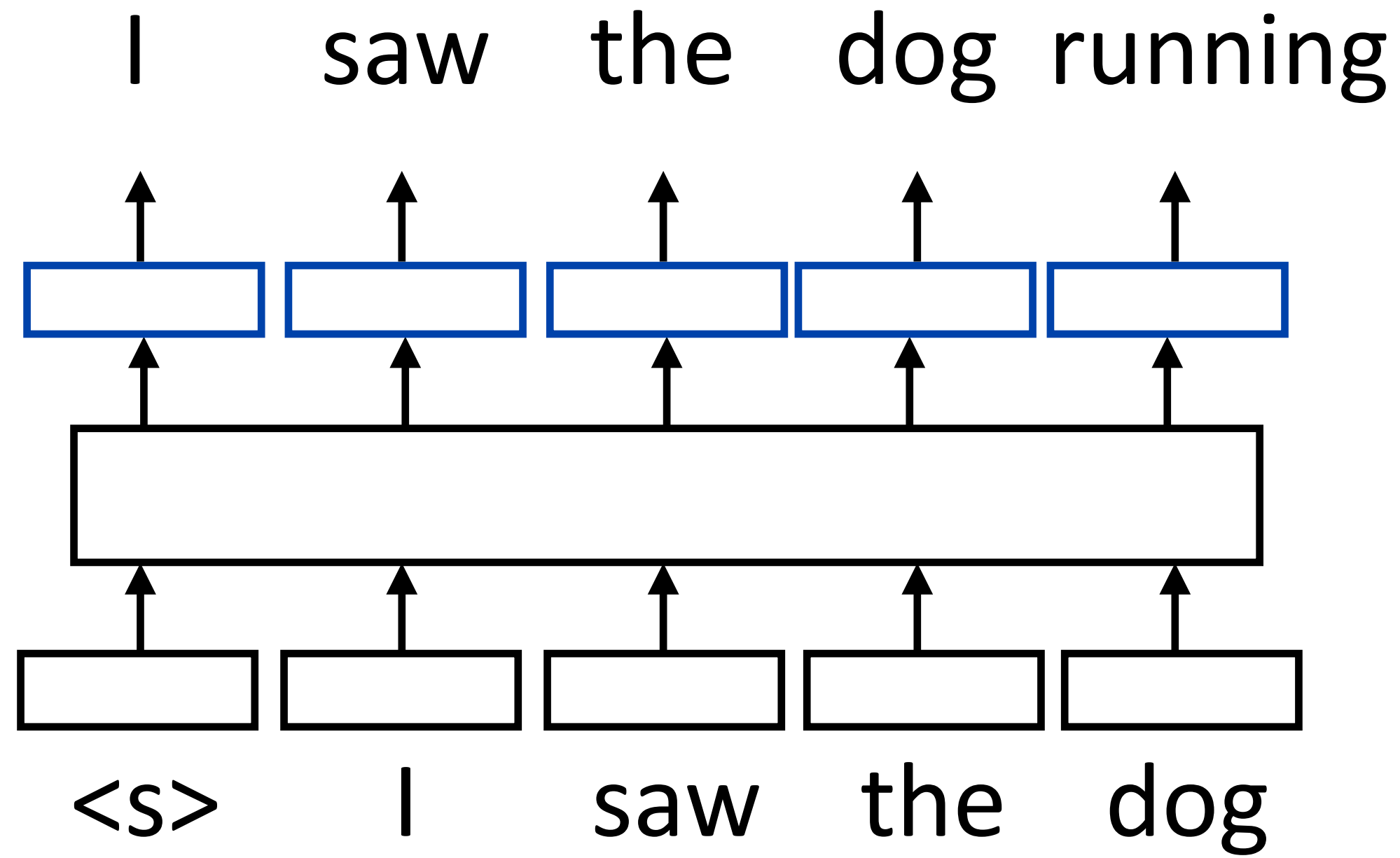


$$P(w|\text{context}) = \text{softmax}(W\mathbf{h}_i)$$

- ▶ W is a (vocab size) x (hidden size) matrix; linear layer in PyTorch (rows are word embeddings)



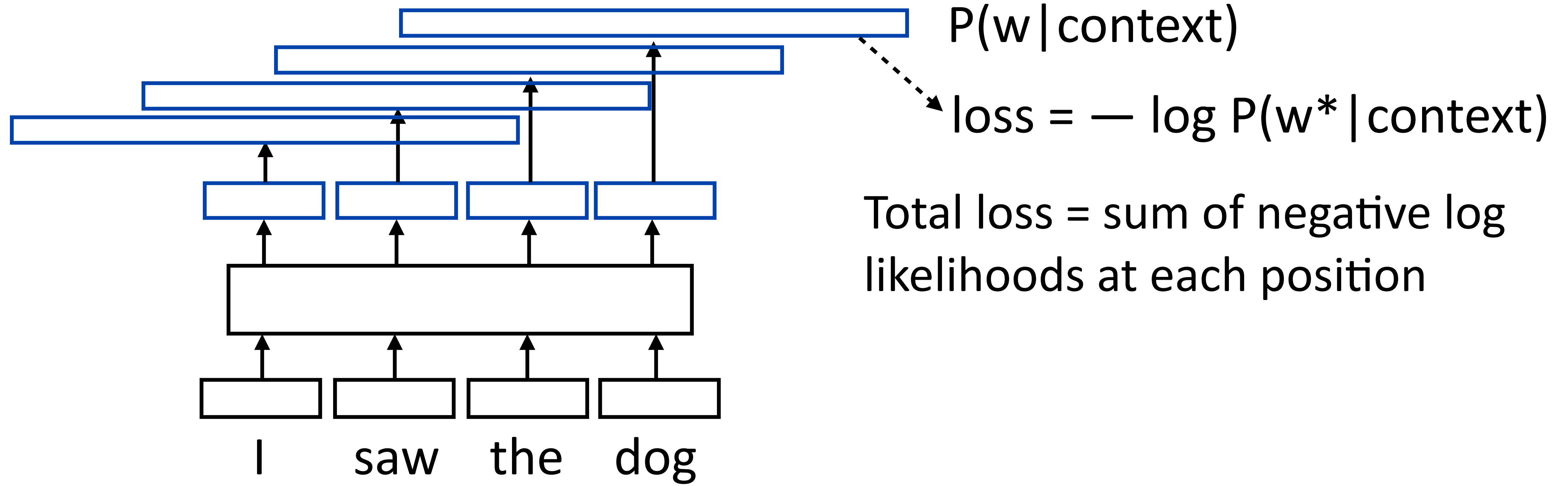
Training Transformer LMs



- ▶ Input is a sequence of words, output is those words shifted by one,
- ▶ Allows us to train on predictions across several timesteps simultaneously (similar to batching but this is NOT what we refer to as batching)



Training Transformer LMs



```
loss_fcn = nn.NLLLoss()
```

```
loss += loss_fcn(log_probs, ex.output_tensor)
```

[seq len, num output classes] [seq len]

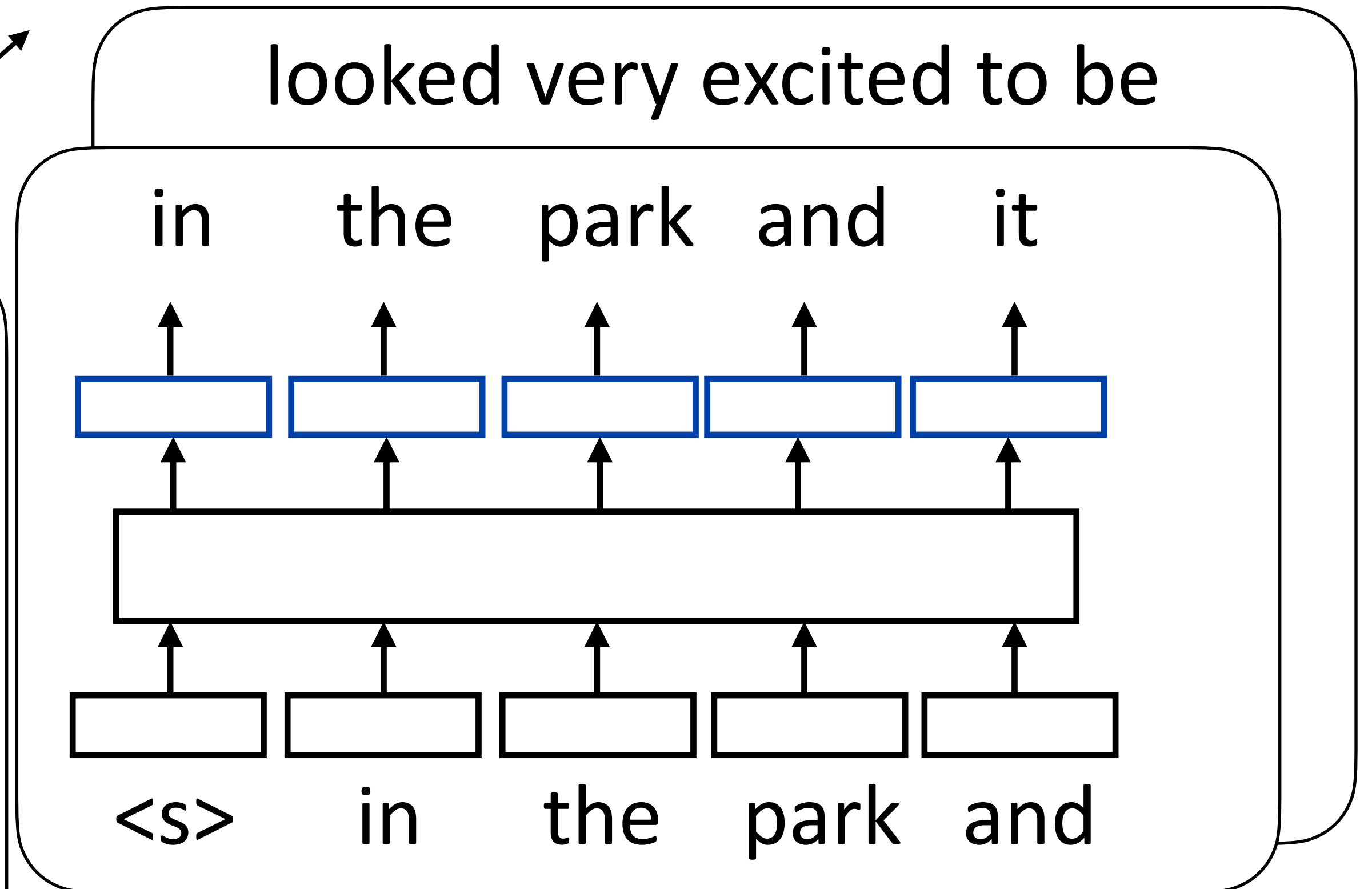
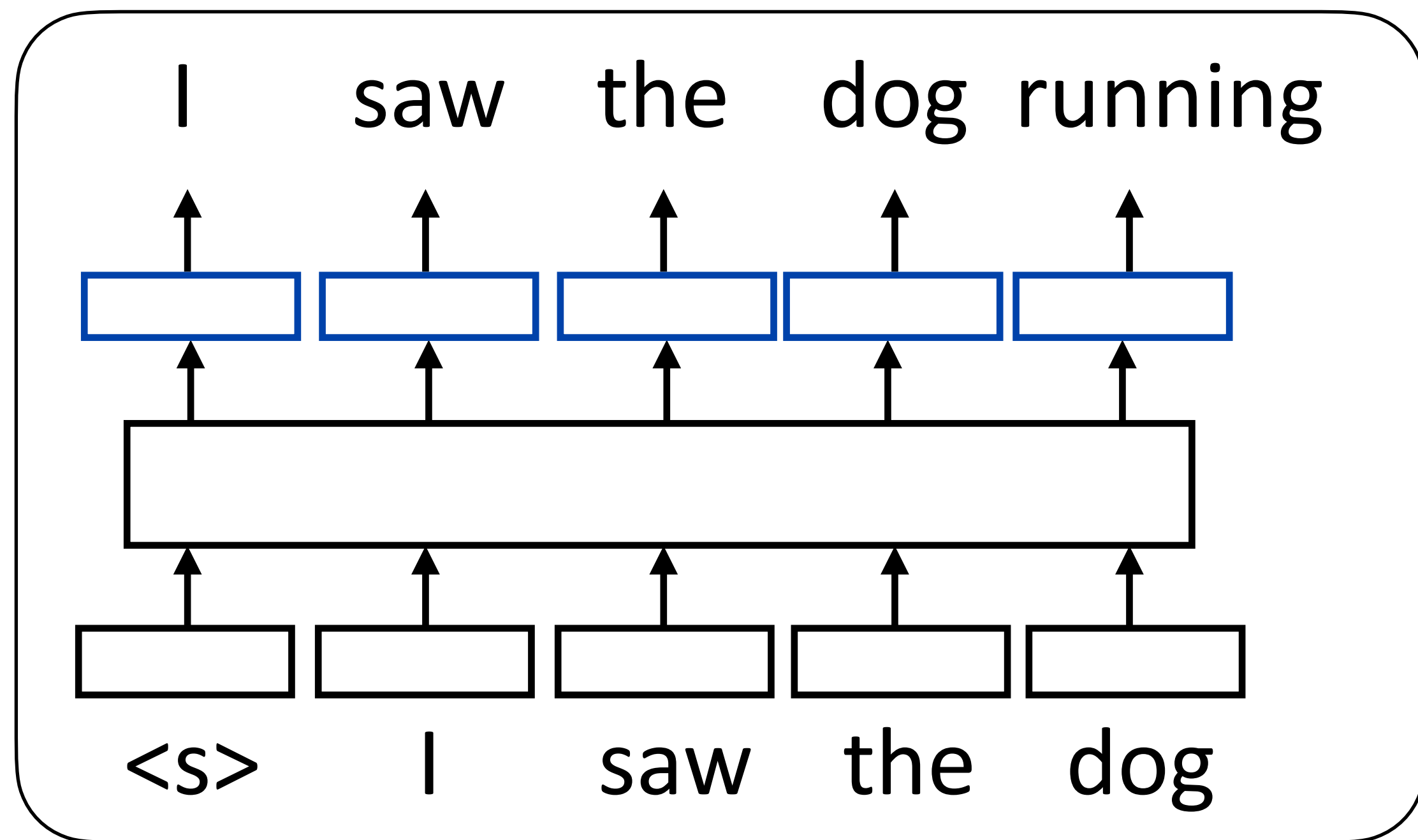
- ▶ Batching is a little tricky with NLLLoss: need to collapse [batch, seq len, num classes] to [batch * seq len, num classes]. You do not need to batch



Batched LM Training

I saw the dog running in the park and it looked very excited to be there

batch dim

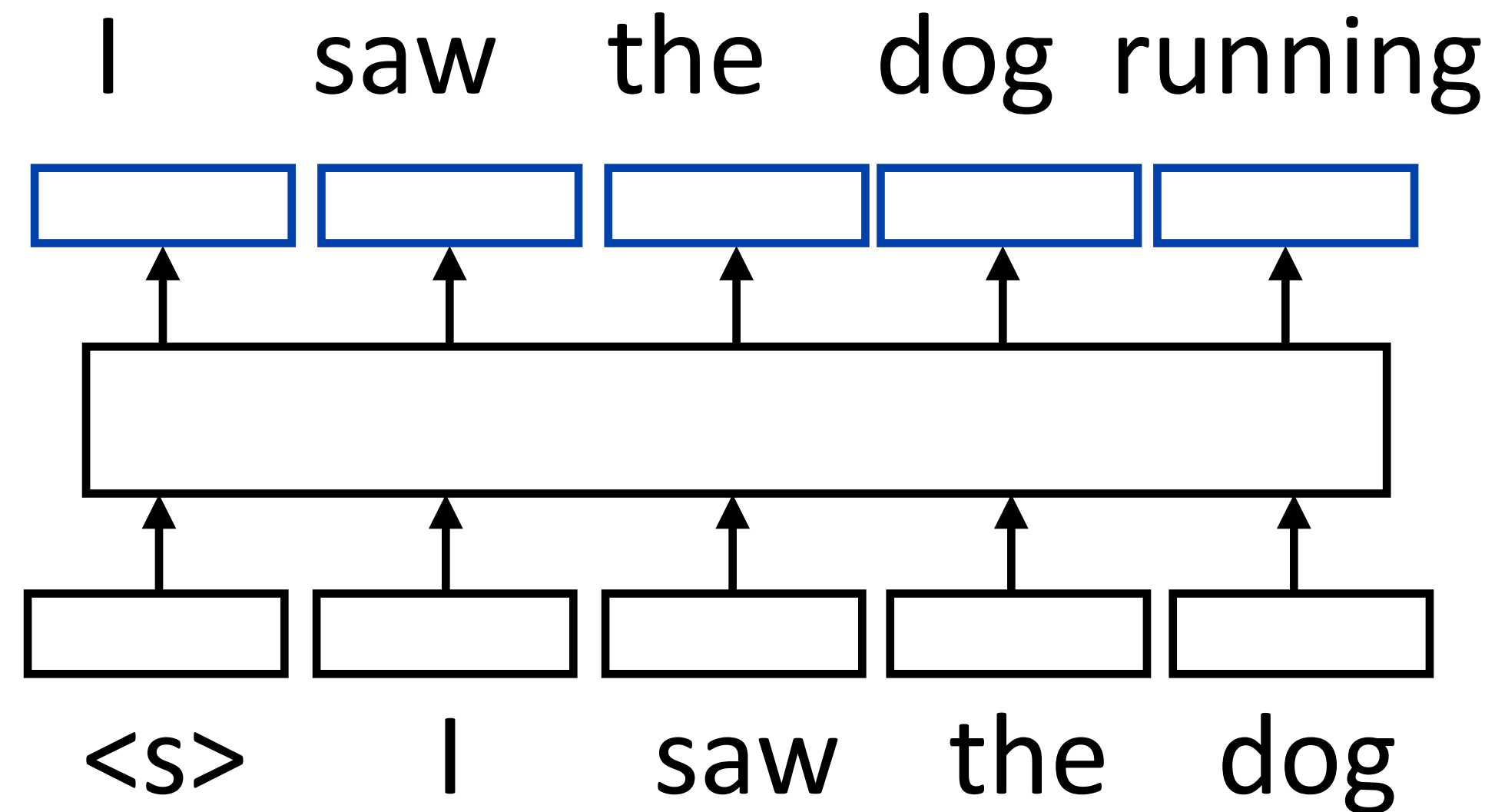


- ▶ Multiple sequences **and** multiple timesteps per sequence



A Small Problem with Transformer LMs

- ▶ This Transformer LM as we've described it will *easily* achieve perfect accuracy. Why?

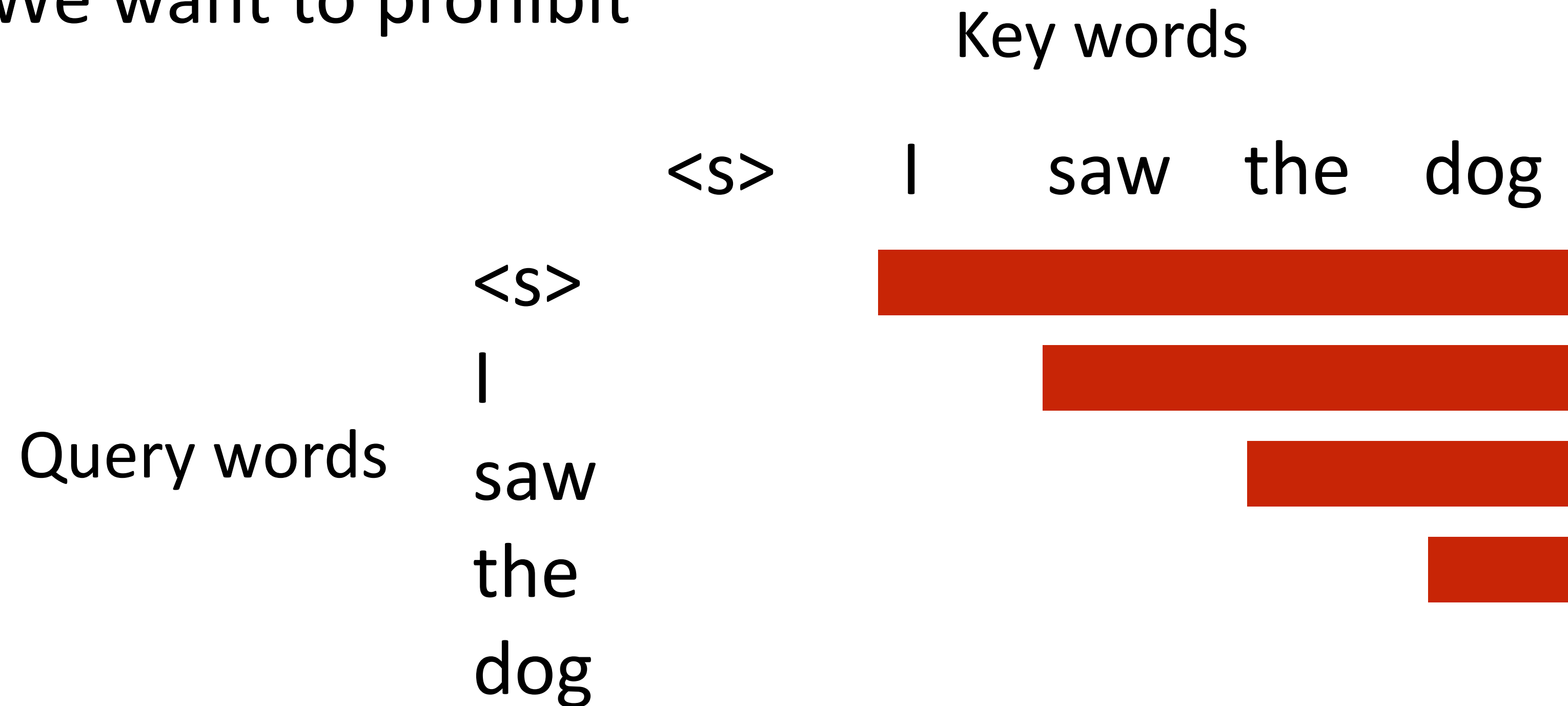


- ▶ With standard self-attention: “I” attends to “saw” and the model is “cheating”. How do we ensure that this doesn’t happen?



Attention Masking

- ▶ We want to prohibit



- ▶ We want to mask out everything in red (an upper triangular matrix)



Implementing in PyTorch

- ▶ `nn.TransformerEncoder` can be built out of `nn.TransformerEncoderLayers`, can accept an input and a mask for language modeling:

```
# Inside the module; need to fill in size parameters
layers = nn.TransformerEncoderLayer([...])
transformer_encoder = nn.TransformerEncoder(encoder_layers, num_layers=[...])
[. . .]
# Inside forward(): puts negative infinities in the red part
mask = torch.triu(torch.ones(len, len) * float('-inf'), diagonal=1)
output = transformer_encoder(input, mask=mask)
```

- ▶ **You cannot use these for Part 1, only for Part 2**



LM Evaluation

- ▶ Accuracy doesn't make sense — predicting the next word is generally impossible so accuracy values would be very low
- ▶ Evaluate LMs on the likelihood of held-out data (averaged to normalize for length)

$$\frac{1}{n} \sum_{i=1}^n \log P(w_i | w_1, \dots, w_{i-1})$$

- ▶ Perplexity: $\exp(\text{average negative log likelihood})$. Lower is better
 - ▶ Suppose we have probs $1/4, 1/3, 1/4, 1/3$ for 4 predictions
 - ▶ Avg NLL (base e) = 1.242 Perplexity = 3.464 \leq geometric mean of denominators



Scaling Laws

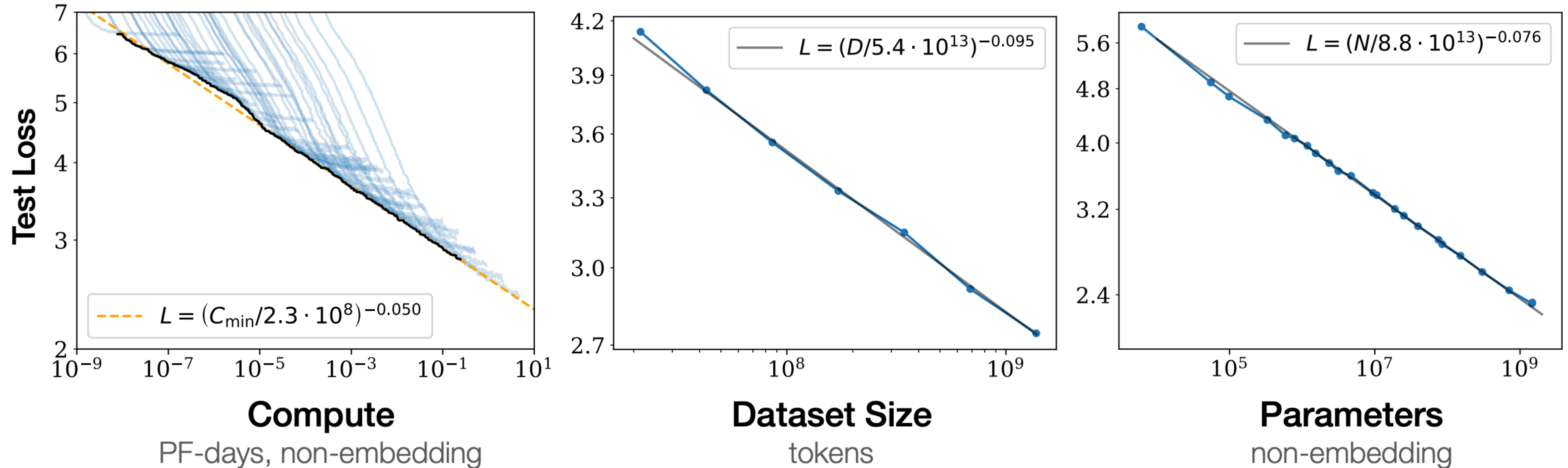


Figure 1 Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute² used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

- Transformers scale really well!

Kaplan et al. (2020)



Takeaways

- ▶ Transformers are going to be the foundation for the much of the rest of this class and are a ubiquitous architecture nowadays
- ▶ Many details to get right, many ways to tweak and extend them, but core idea is the multi-head self attention and their ability to contextualize items in sequences

Pretraining Intro, ELMo



What is pre-training?

- ▶ “Pre-train” a model on a large dataset for task X, then “fine-tune” it on a dataset for task Y
- ▶ Key idea: X is somewhat related to Y, so a model that can do X will have some good neural representations for Y as well
- ▶ ImageNet pre-training is huge in computer vision: learn generic visual features for recognizing objects
- ▶ GloVe can be seen as pre-training: learn vectors with the skip-gram objective on large data (task X), then fine-tune them as part of a neural network for sentiment/any other task (task Y)

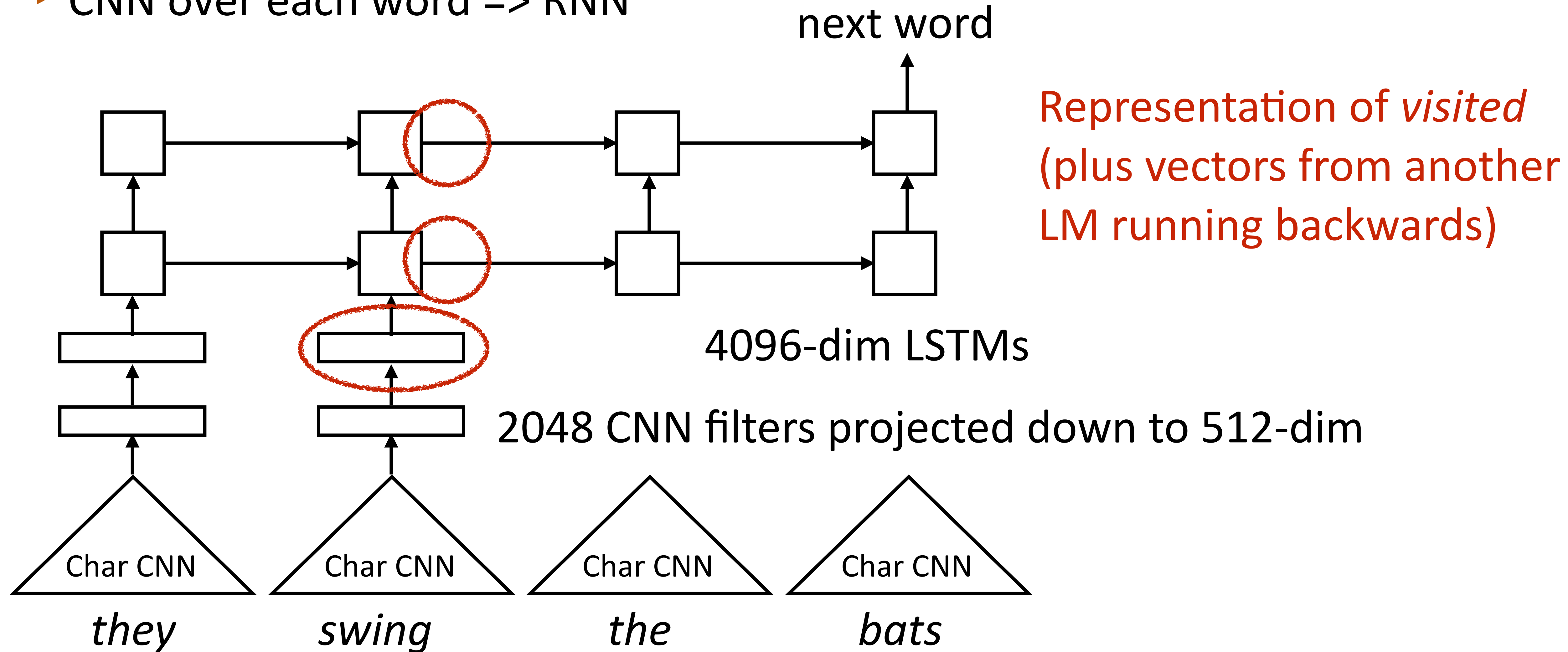


GloVe is insufficient

- ▶ GloVe uses a lot of data but in a weak way
- ▶ Having a single embedding for each word is wrong

they swing the bats *they see the bats*
- ▶ Identifying discrete word senses is hard, doesn't scale. Hard to identify how many senses each word has
- ▶ How can we make our word embeddings more *context-dependent*?

- ▶ CNN over each word => RNN





ELMo

- ▶ Use the embeddings as a drop-in replacement for GloVe
- ▶ Huge gains across many high-profile tasks: NER, question answering, semantic role labeling (similar to parsing), etc.
- ▶ But what if the pre-training **isn't only the embeddings?**

BERT



BERT



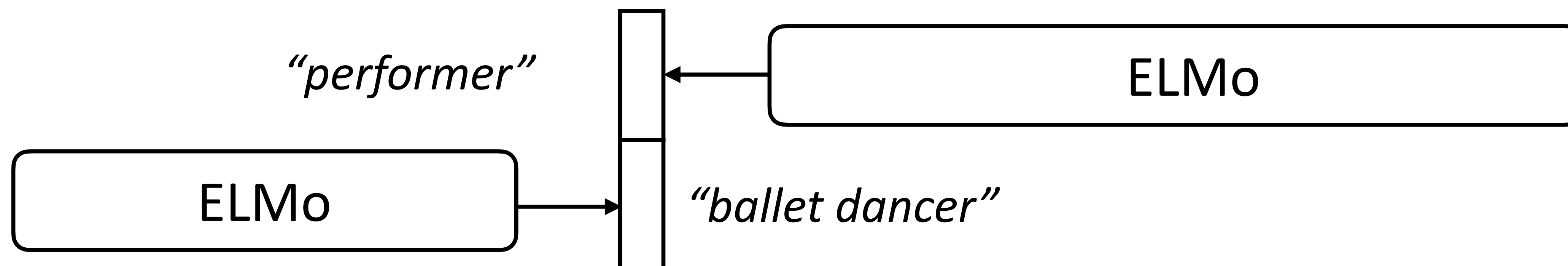
- ▶ AI2 made ELMo in spring 2018, GPT (transformer-based ELMo) was released in summer 2018, BERT came out October 2018
- ▶ Four major changes compared to ELMo:
 - ▶ Transformers instead of LSTMs
 - ▶ Bidirectional model with “Masked LM” objective instead of standard LM
 - ▶ Fine-tune instead of freeze at test time
 - ▶ Operates over word pieces (byte pair encoding)



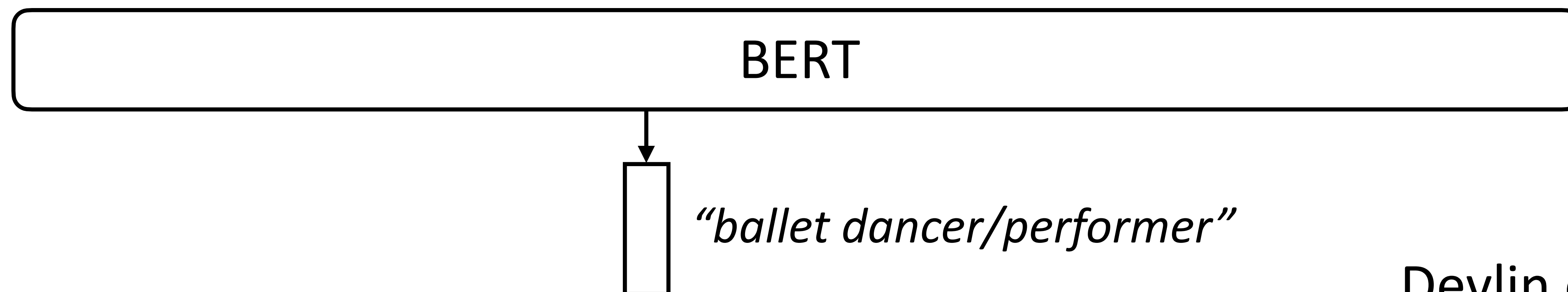
BERT



- ▶ 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.



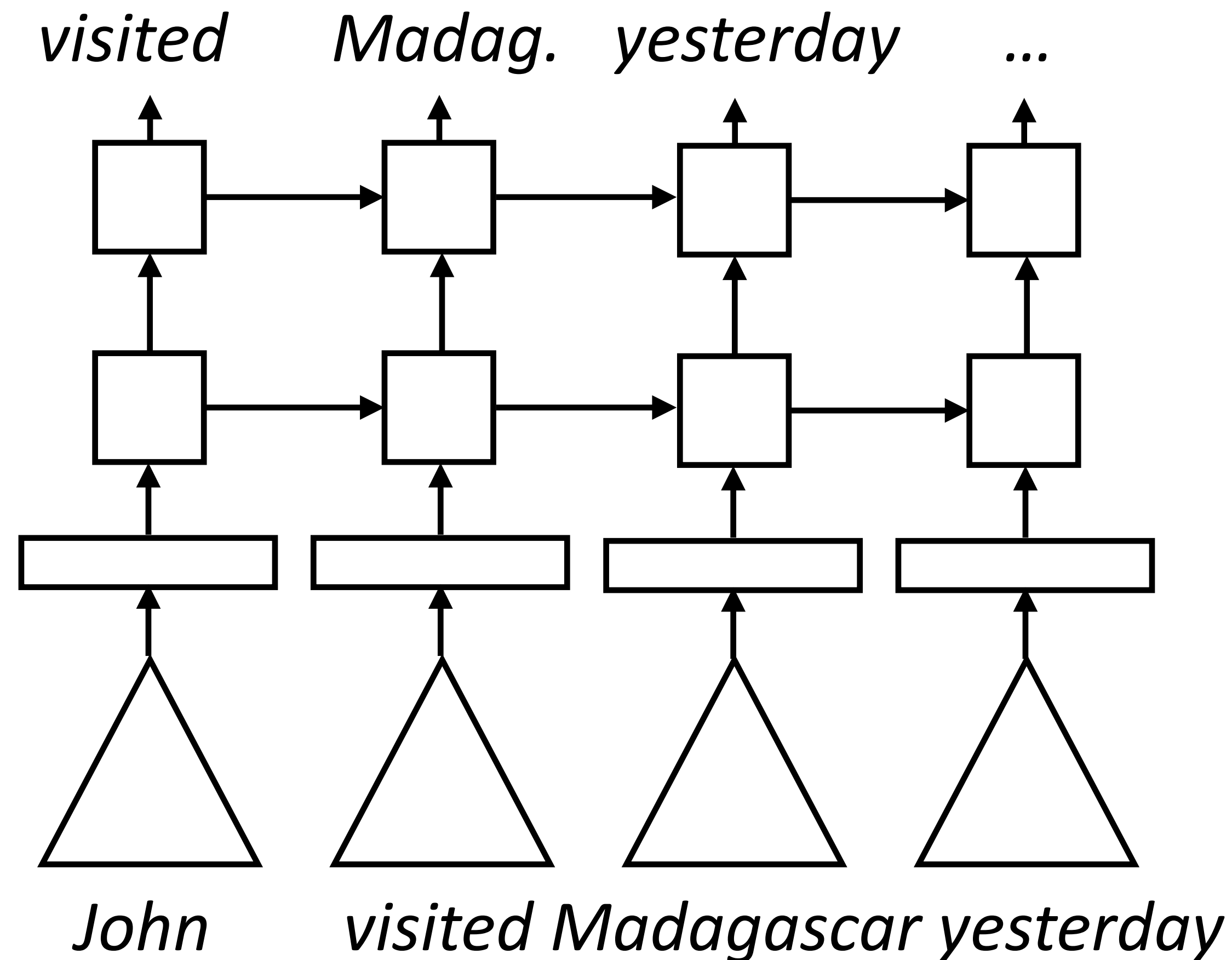


BERT

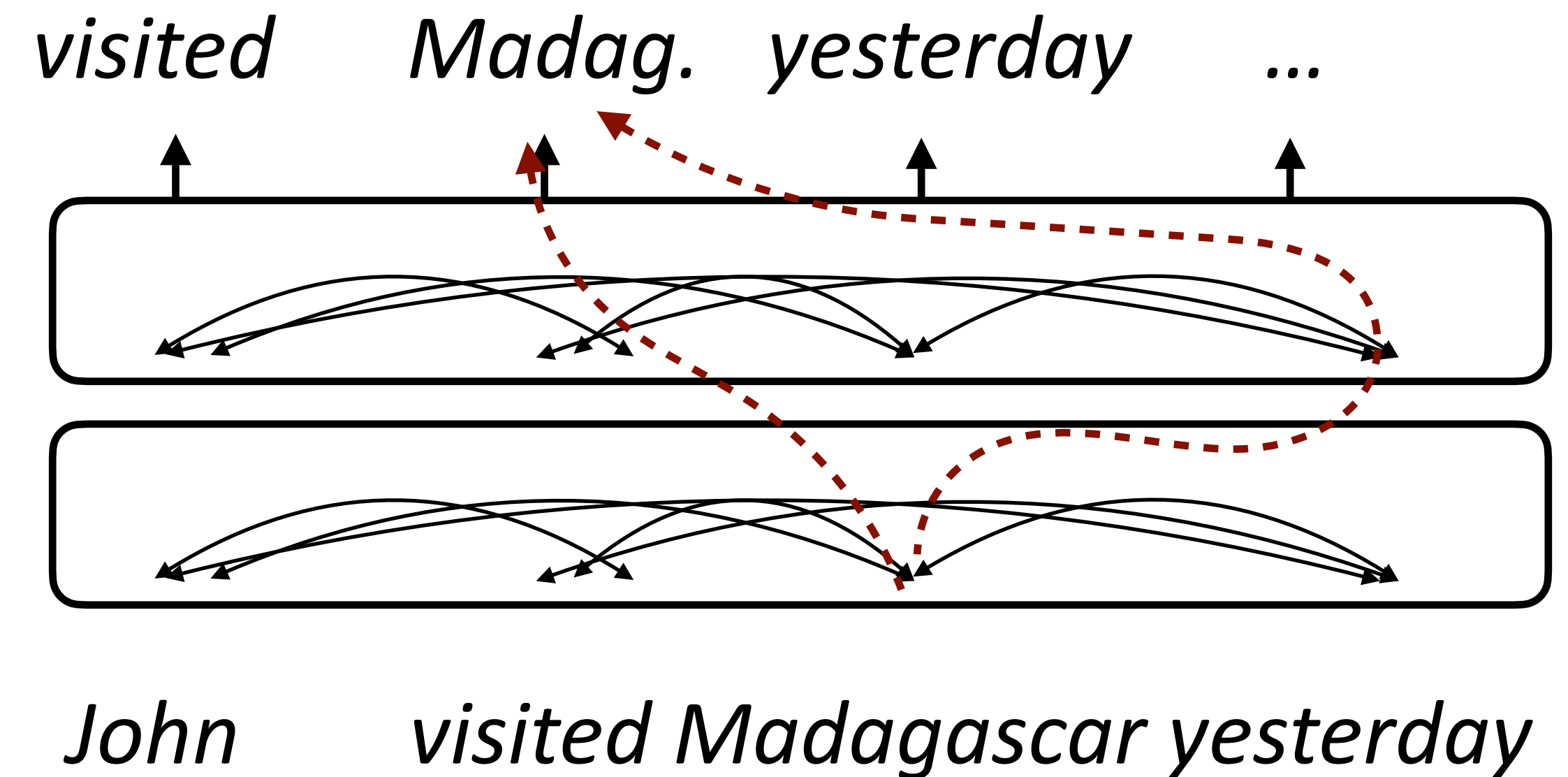


- How to learn a “deeply bidirectional” model? What happens if we just replace an LSTM with a transformer?

ELMo (Language Modeling)



BERT

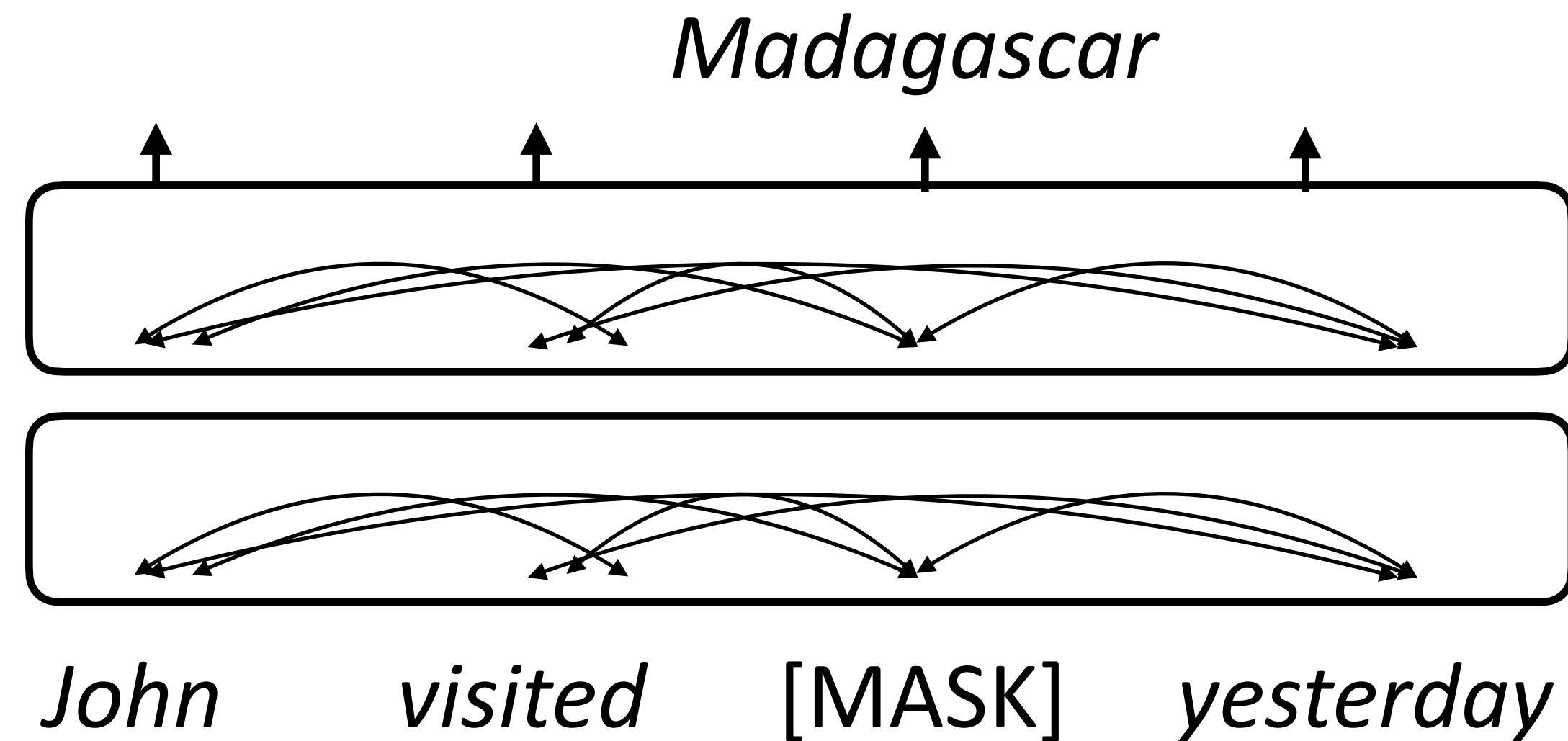


- You could do this with a “one-sided” transformer, but this “two-sided” model can cheat



Masked Language Modeling

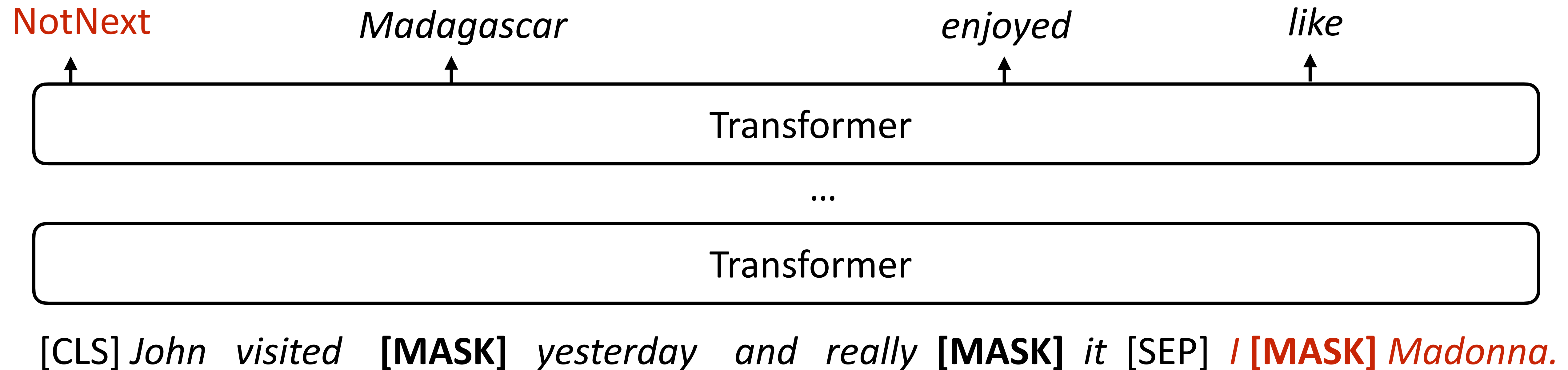
- ▶ How to prevent cheating? Next word prediction fundamentally doesn't work for bidirectional models, instead do *masked language modeling*
- ▶ BERT formula: take a chunk of text, mask out 15% of the tokens, and try to predict them





Next “Sentence” Prediction

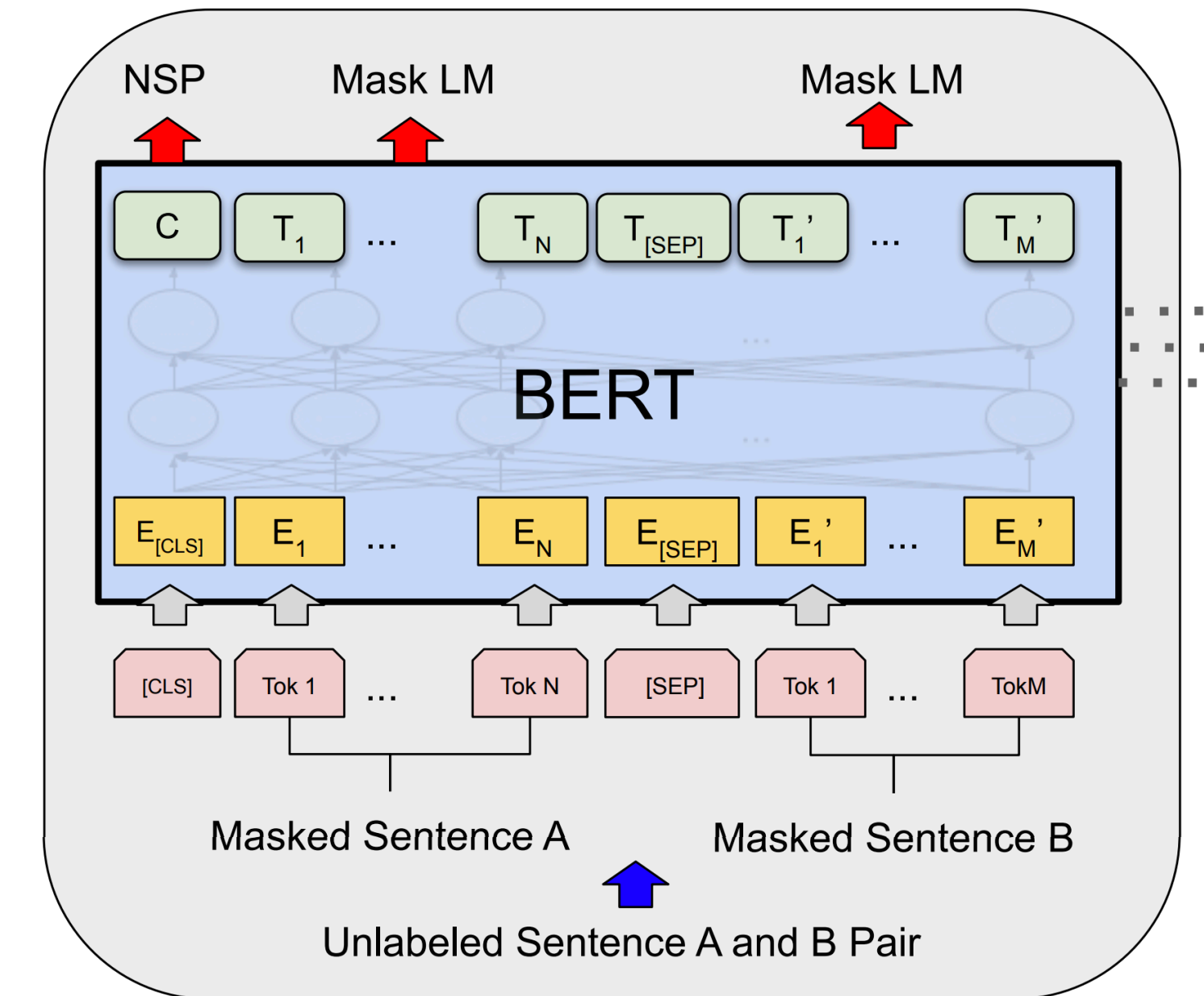
- ▶ Input: [CLS] Text chunk 1 [SEP] Text chunk 2
- ▶ 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk. Predict whether the next chunk is the “true” next
- ▶ BERT objective: masked LM + next sentence prediction





BERT Architecture

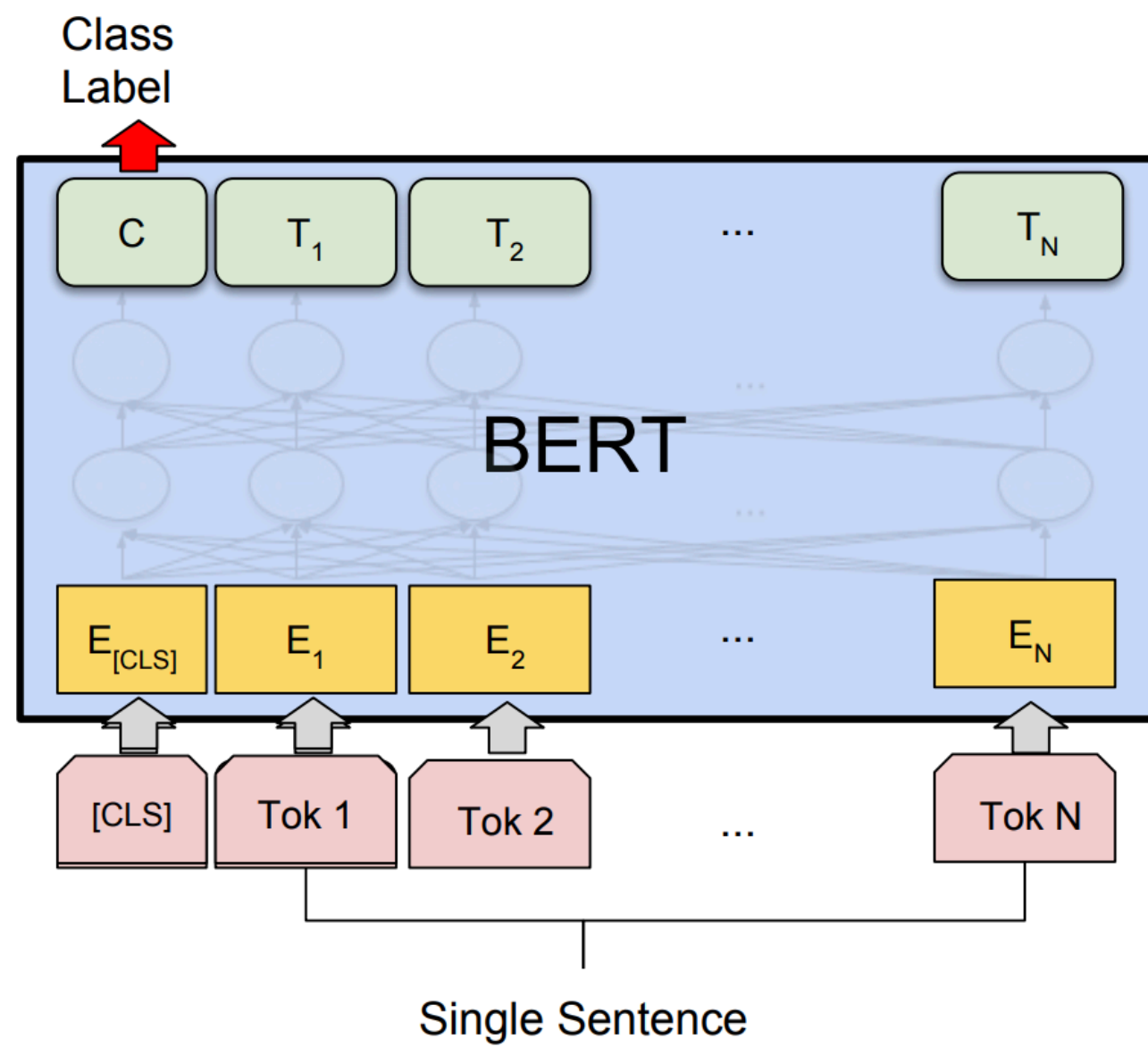
- ▶ BERT Base: 12 layers, 768-dim per wordpiece token, 12 heads. Total params = 110M
- ▶ BERT Large: 24 layers, 1024-dim per wordpiece token, 16 heads. Total params = 340M
- ▶ Positional embeddings and segment embeddings, 30k word pieces
- ▶ This is the model that gets **pre-trained** on a large corpus



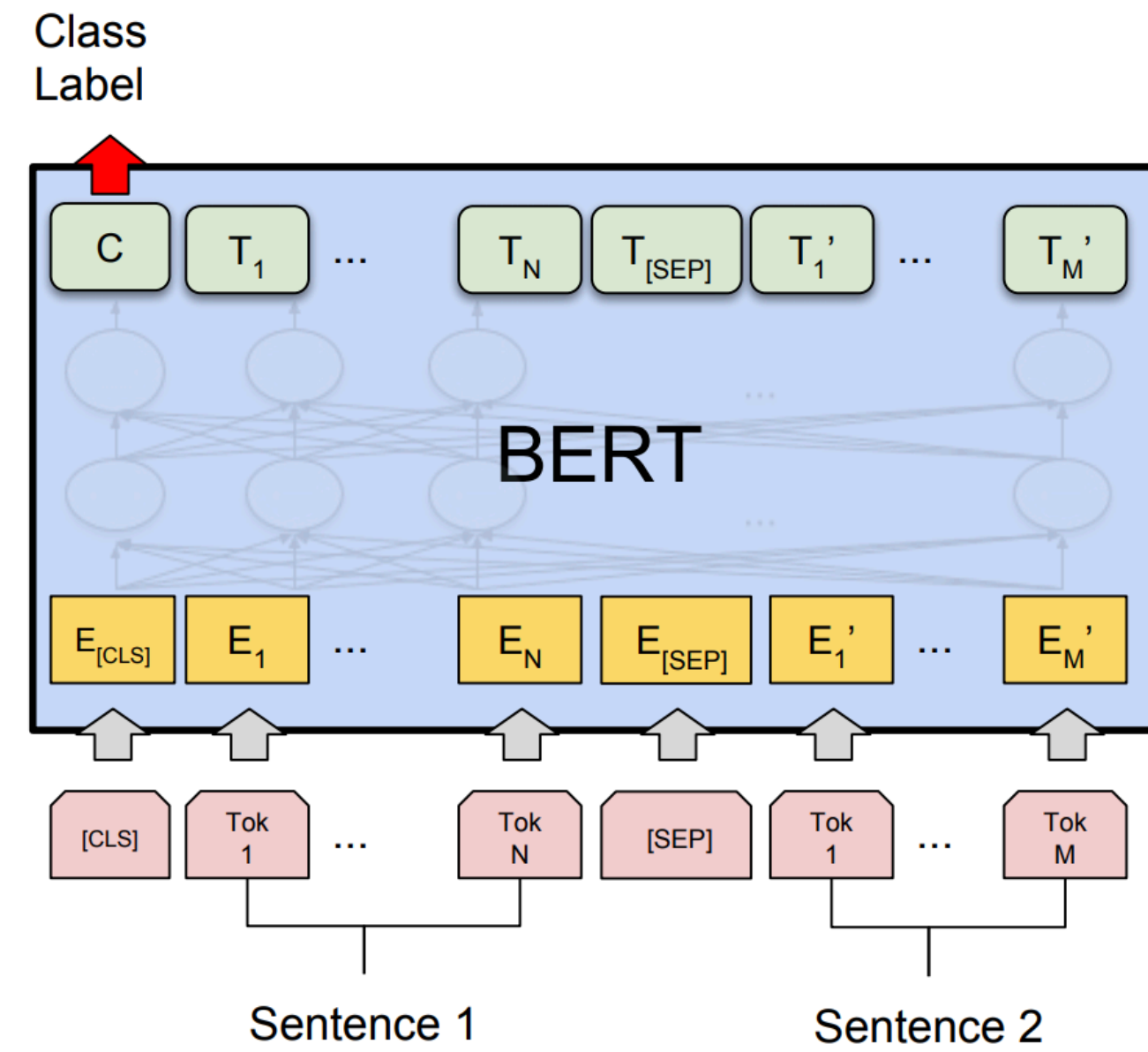
Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{##ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}



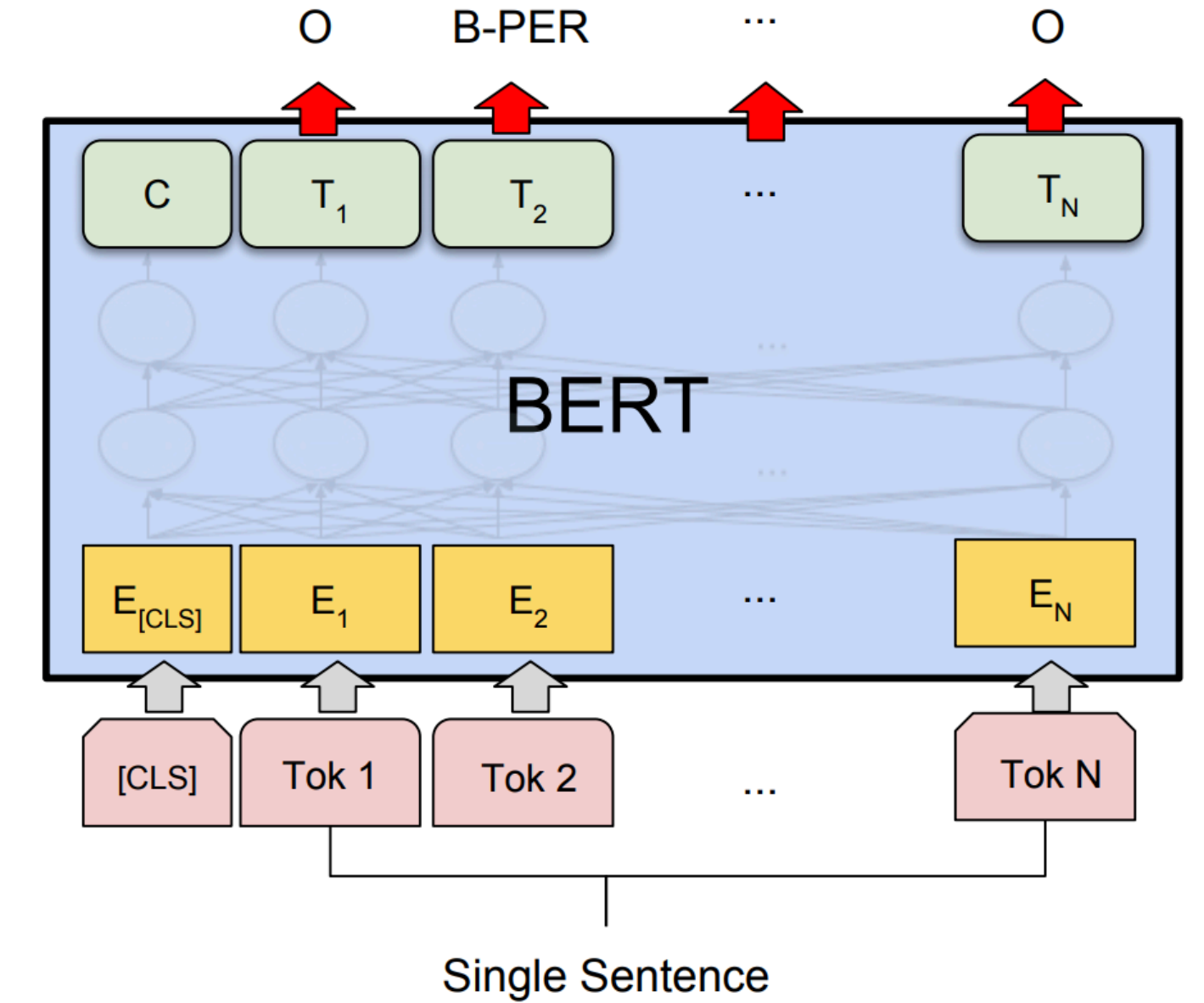
What can BERT do?



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



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



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

- ▶ Artificial [CLS] token is used as the vector to do classification from
 - ▶ 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)



Natural Language Inference

Premise

Hypothesis

A boy plays in the snow

entails

A boy is outside

A man inspects the uniform of a figure

contradicts

The man is sleeping

An older and younger man smiling

neutral

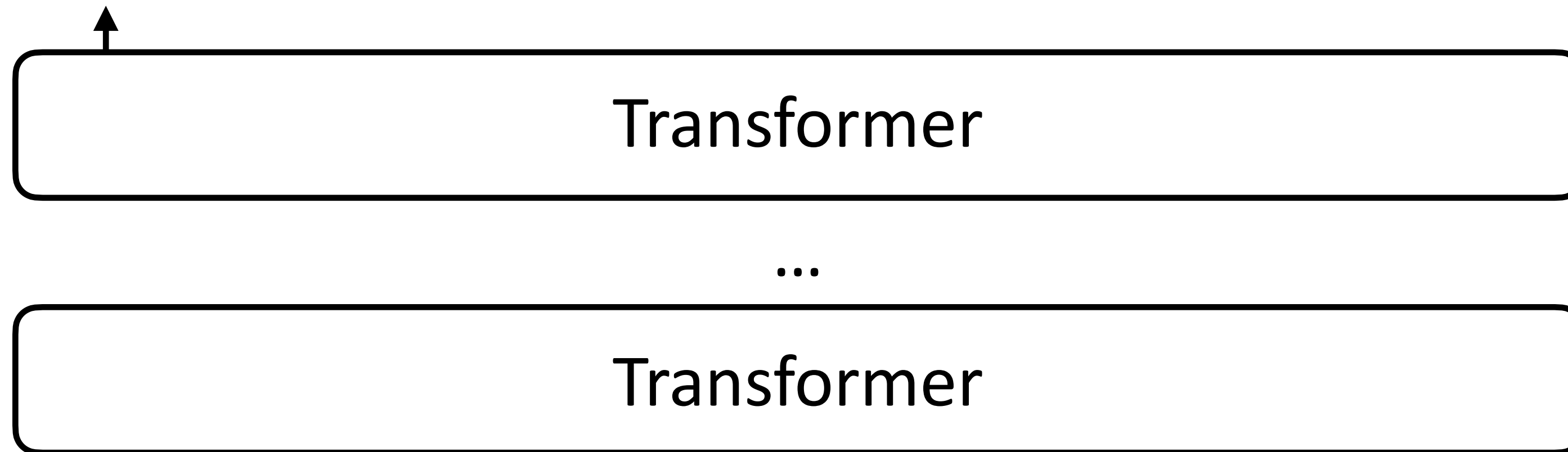
Two men are smiling and
laughing at cats playing

- ▶ Long history of this task: “Recognizing Textual Entailment” challenge in 2006 (Dagan, Glickman, Magnini)
- ▶ Early datasets: small (hundreds of pairs), very ambitious (lots of world knowledge, temporal reasoning, etc.)

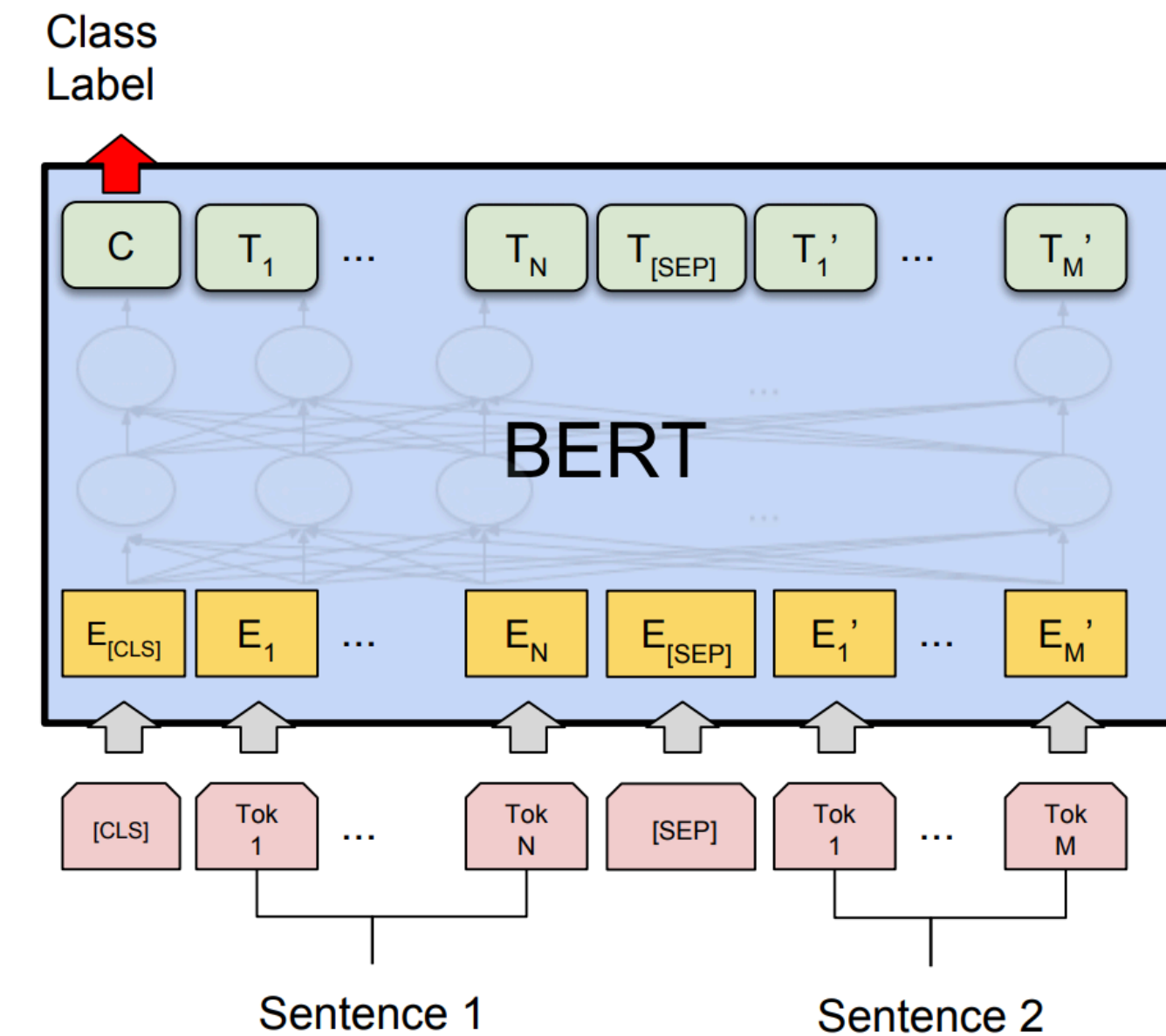


What can BERT do?

Entails (first sentence implies second is true)



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



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

- ▶ 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



SQuAD

Q: What was Marie Curie the first female recipient of?

Passage: One of the most famous people born in Warsaw was Marie Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the **Nobel Prize**. Famous musicians include Władysław Szpilman and Frédéric Chopin. Though Chopin was born in the village of Żelazowa Wola, about 60 km (37 mi) from Warsaw, he moved to the city with his family when he was seven months old. Casimir Pulaski, a Polish general and hero of the American Revolutionary War, was born here in 1745.

Answer = Nobel Prize

- ▶ Assume we know a passage that contains the answer. More recent work has shown how to retrieve these effectively (will discuss when we get to QA)



SQuAD

Q: What was Marie Curie the first female recipient of?

Passage: One of the most famous people born in Warsaw was Marie Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the **Nobel Prize**. ...

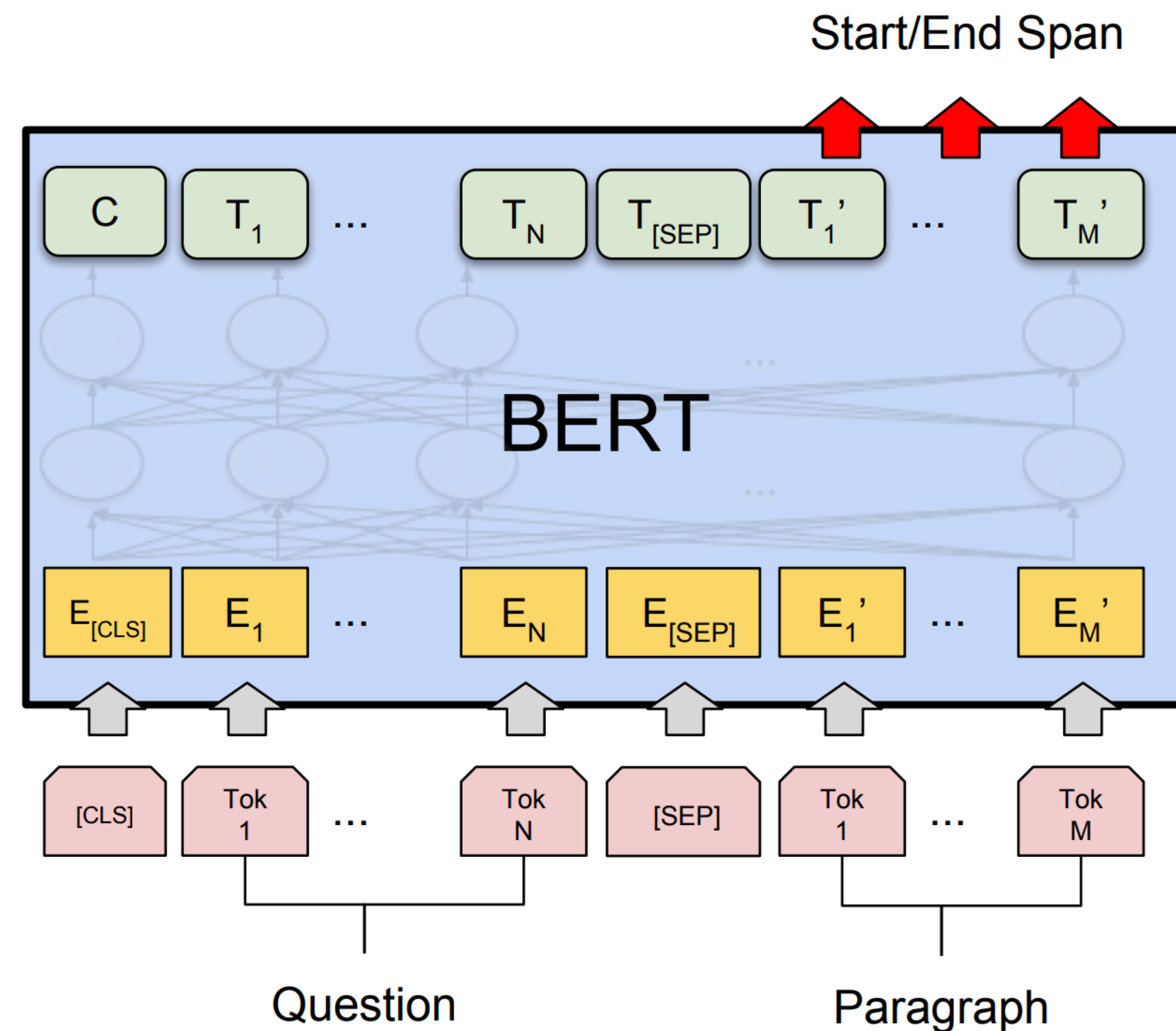
- Predict answer as a pair of (start, end) indices given question q and passage p; compute a score for each word and softmax those

$$P(\text{start} \mid q, p) = \begin{array}{ccccc} 0.01 & 0.01 & 0.01 & 0.85 & 0.01 \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ & \text{recipient of the } \mathbf{Nobel Prize} . \end{array}$$

$P(\text{end} \mid q, p)$ = same computation but different params



QA with BERT



What was Marie Curie the first female recipient of ? [SEP] One of the most famous people born in Warsaw was Marie ...



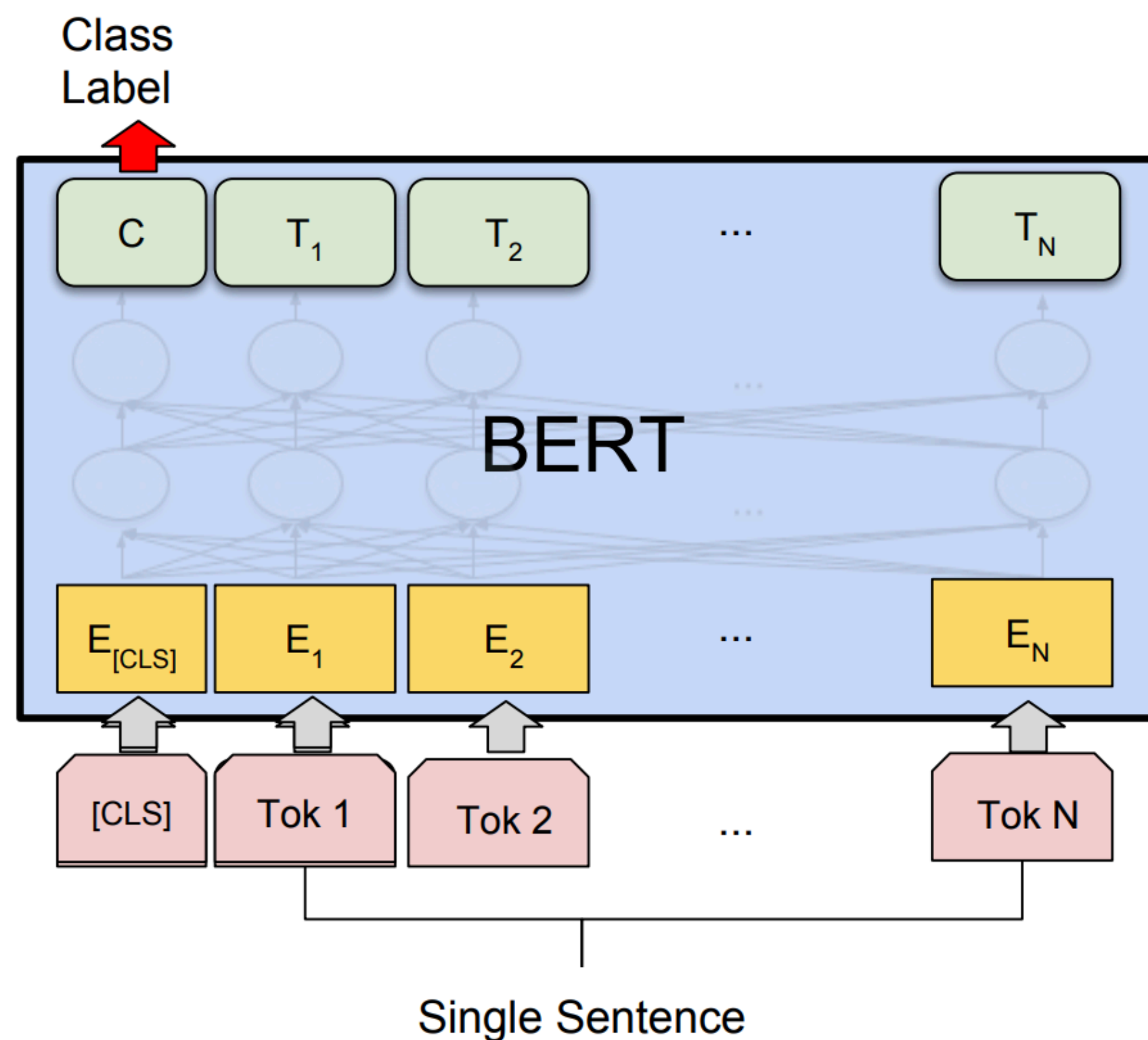
What can BERT NOT do?

- ▶ BERT **cannot** generate text (at least not in an obvious way)
 - ▶ Can fill in MASK tokens, but can't generate left-to-right (well, you could put MASK at the end repeatedly, but this is slow)
- ▶ 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
- ▶ Often requires tricky learning rate schedules (“triangular” learning rates with warmup periods)

BERT Results



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

Wang et al. (2019)



Results

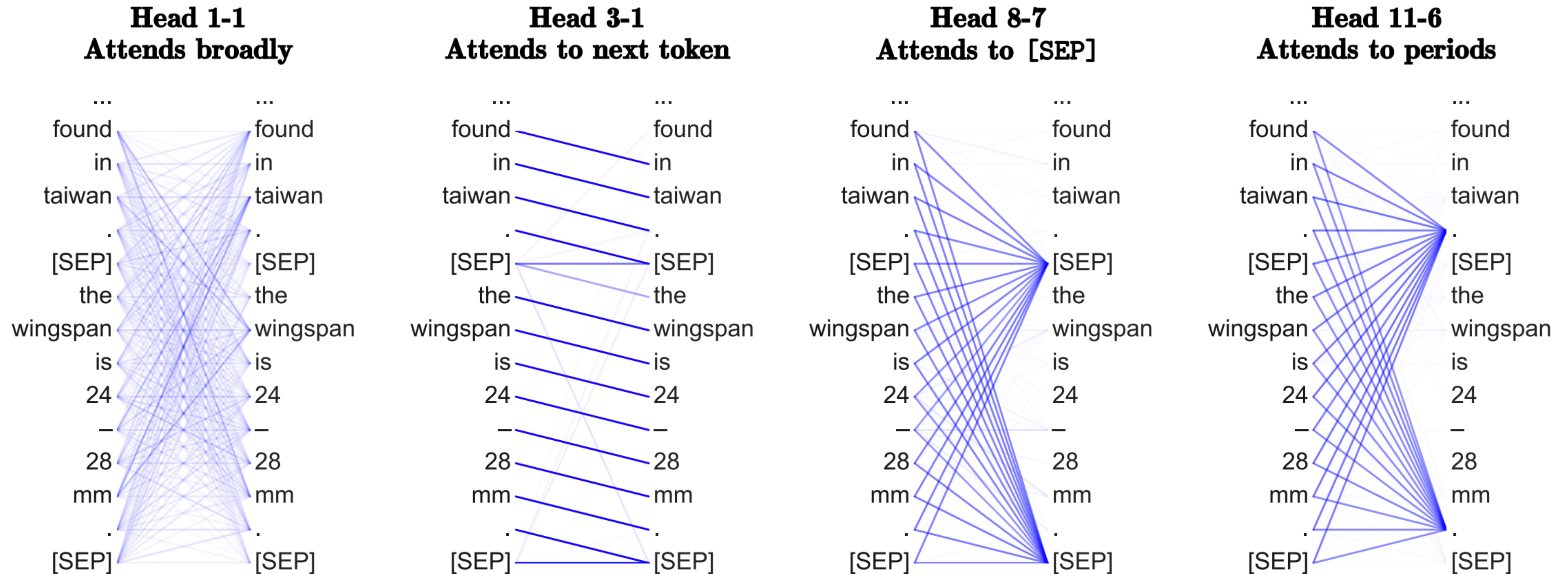
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT _{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

Devlin et al. (2018)



What does BERT learn?



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

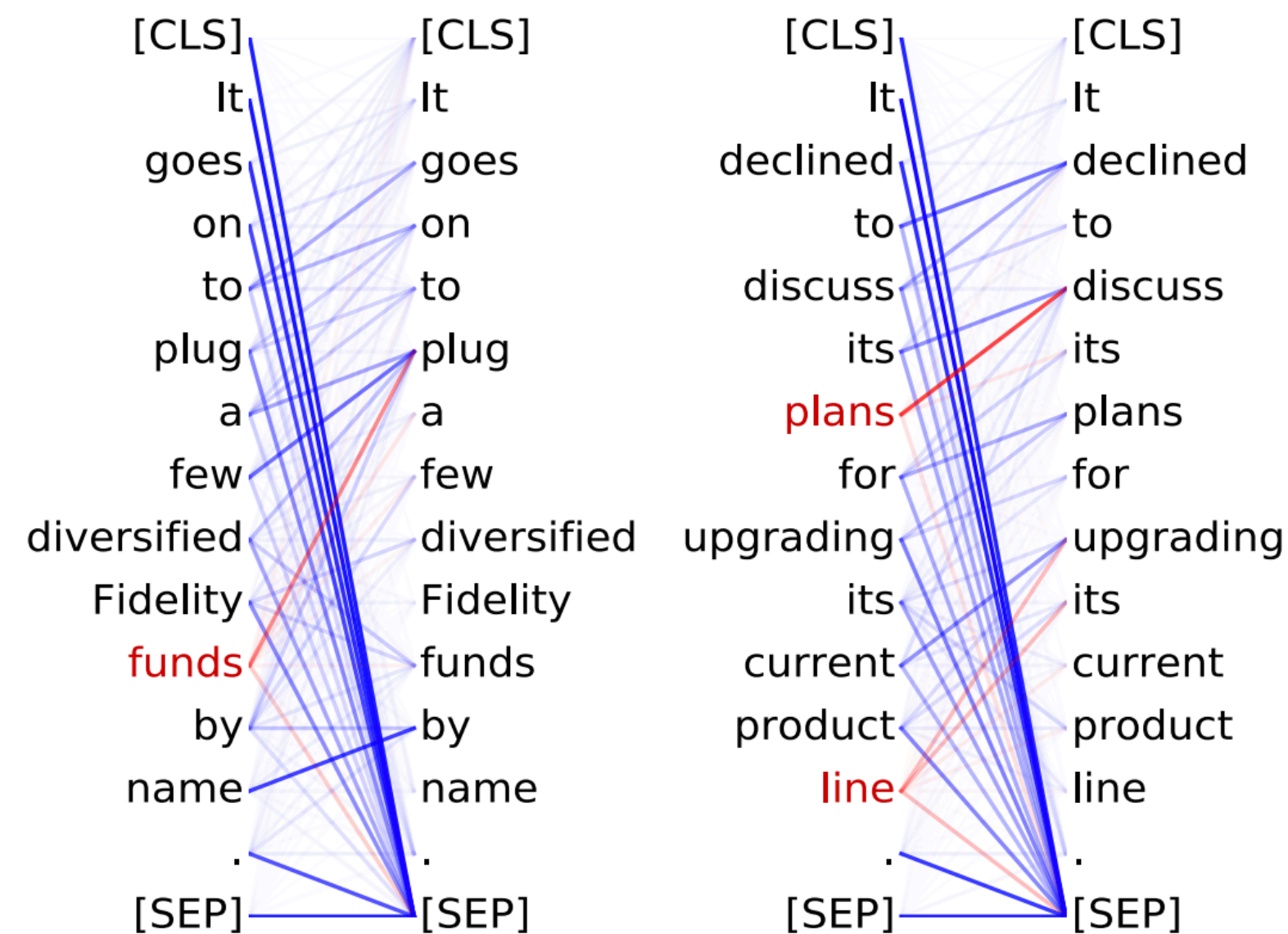
Clark et al. (2019)



What does BERT learn?

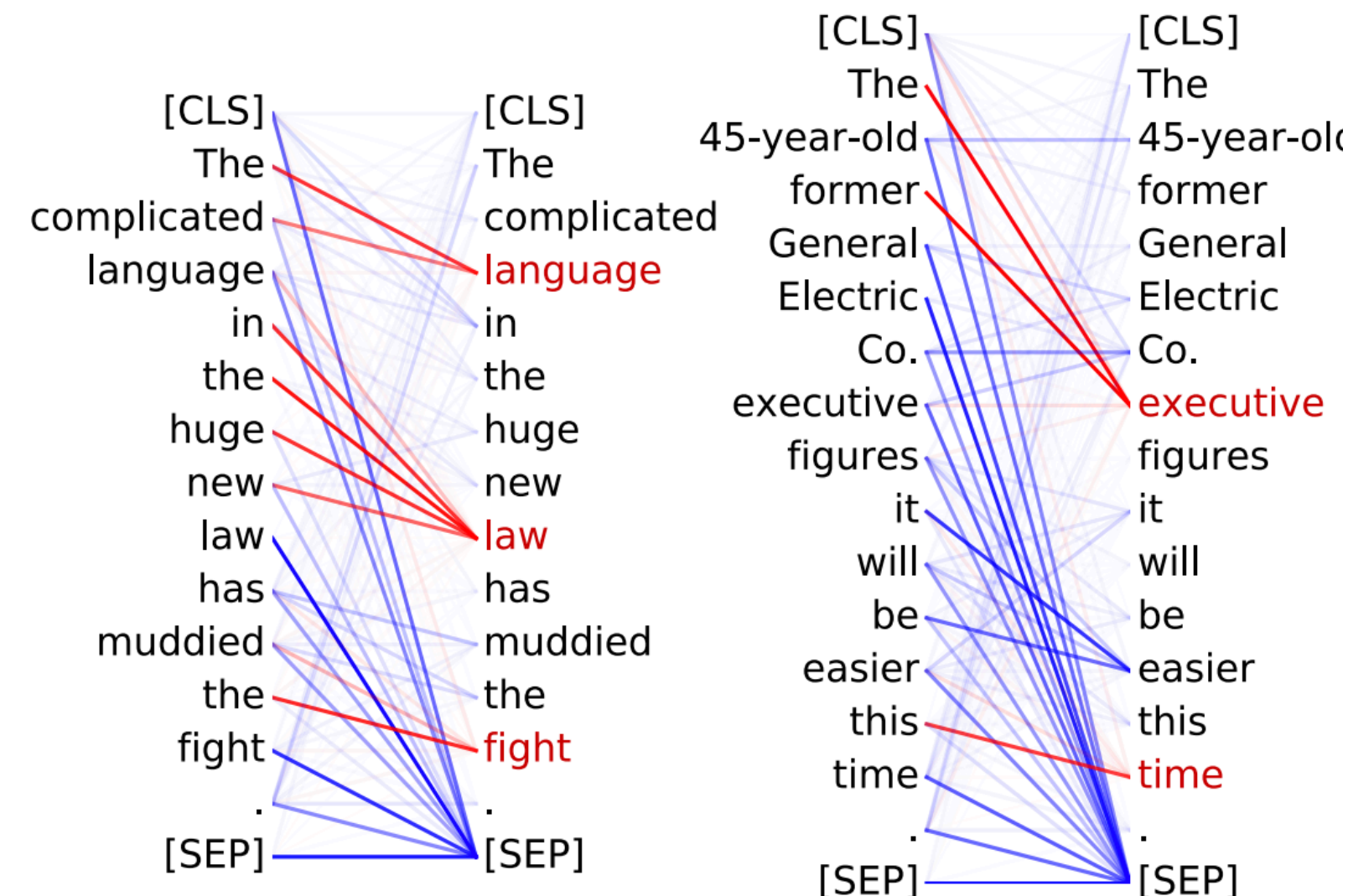
Head 8-10

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



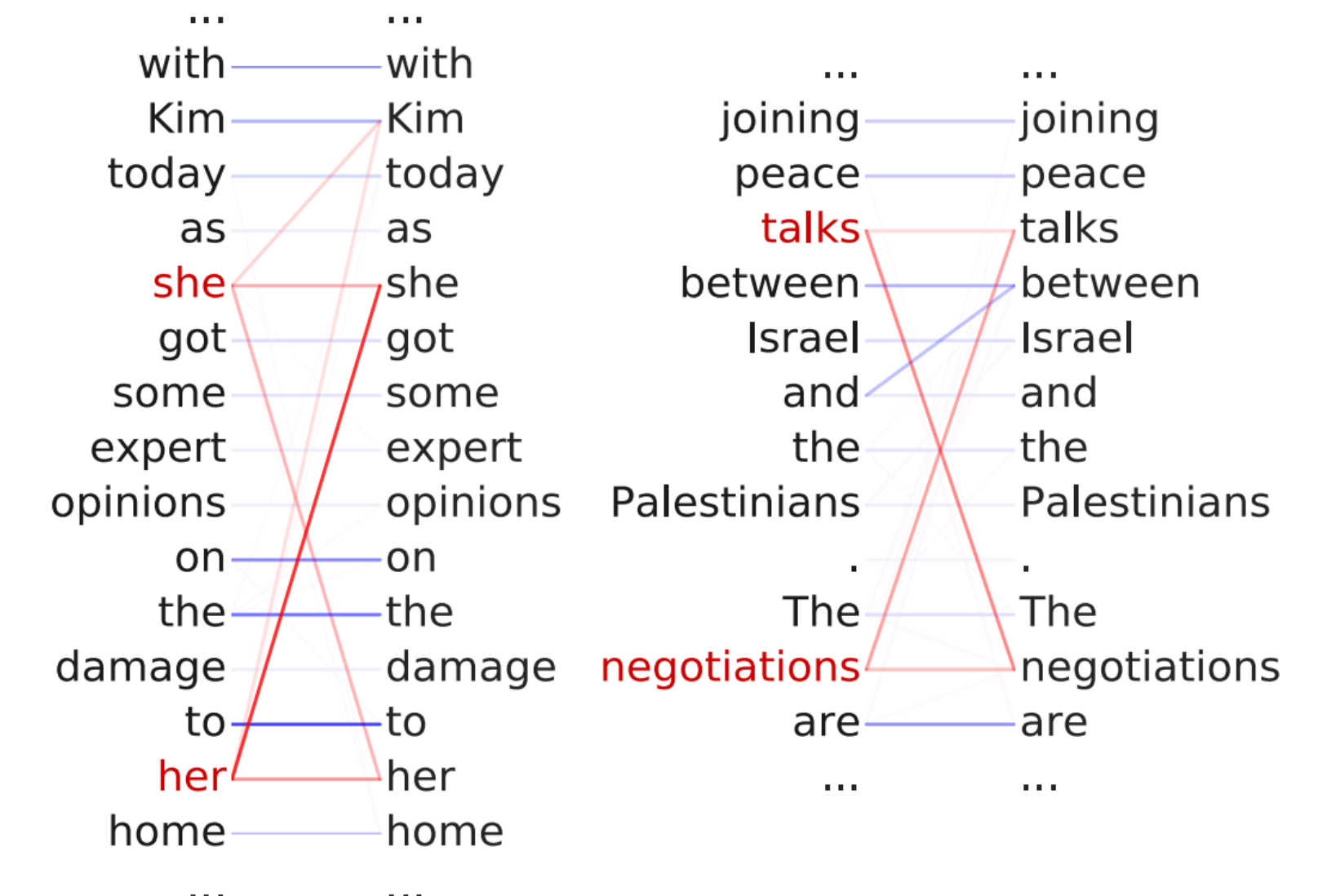
Head 8-11

- **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

Subword Tokenization



Handling Rare Words

- ▶ Words are a difficult unit to work with. Why?
 - ▶ When you have 100,000+ words, the final matrix multiply and softmax start to dominate the computation, many params, still some words you haven't seen, doesn't take advantage of morphology, ...
- ▶ Character-level models were explored extensively in 2016-2018 but simply don't work well — becomes very expensive to represent sequences



Subword Tokenization

- ▶ Subword tokenization: wide range of schemes that use tokens that are **between characters and words** in terms of granularity
- ▶ These “word pieces” may be full words or parts of words

`_the _eco tax _port i co _in _Po nt - de - Bu is ...`

- ▶ `_` indicates the word piece starting a word (can think of it as the space character).



Subword Tokenization

- ▶ Subword tokenization: wide range of schemes that use tokens that are **between characters and words** in terms of granularity
- ▶ These “word pieces” may be full words or parts of words

_the _**eco tax** _port i co _in _Po nt - de - Bu is...

Output: _le _port ique _**éco taxe** _de _Pont - de - Bui s

A diagram illustrating subword tokenization. The input text is "_the _eco tax _port i co _in _Po nt - de - Bu is...". The output text is "_le _port ique _éco taxe _de _Pont - de - Bui s". Two lines connect the input tokens "_eco" and "tax" to the output token "_éco taxe", showing how they are merged into a single subword token. A dashed box encloses the output tokens "_Pont - de - Bui s", which correspond to the input tokens "_Po nt - de - Bu is...".

- ▶ Can achieve transliteration with this, subword structure makes some translations easier to achieve



Byte Pair Encoding (BPE)

- ▶ Start with every individual byte (basically character) as its own symbol

```
for i in range(num_merges):  
    pairs = get_stats(vocab)  
    best = max(pairs, key=pairs.get)  
    vocab = merge_vocab(best, vocab)
```

- ▶ Count bigram character cooccurrences
- ▶ Merge the most frequent pair of adjacent characters

- ▶ Doing 8k merges => vocabulary of around 8000 word pieces. Includes many whole words
- ▶ Most SOTA NMT systems use this on both source + target



Byte Pair Encoding (BPE)

Original:	furiously		Original:	tricycles
BPE:	_fur iously	(b)	BPE:	_t ric y cles
Unigram LM:	_fur ious ly		Unigram LM:	_tri cycle s
Original:	Completely preposterous suggestions			
BPE:	_Comple t ely	_prep ost erous	_suggest ions	
Unigram LM:	_Complete ly	_pre post er ous	_suggestion s	

- ▶ What do you see here?
- ▶ BPE produces less linguistically plausible units than another technique based on a unigram language model: rather than greedily merge, find chunks which make the sequence look likely under a unigram LM
- ▶ Unigram LM tokenizer leads to slightly better BERT

Bostrom and Durrett (2020)



What's in the token vocab?



Matthew Watkins
@SoC_trilogy

I've just found out that several of the anomalous GPT tokens ("TheNitromeFan", " SolidGoldMagikarp", " davidjl", " Smartstocks", " RandomRedditorWithNo",) are handles of people who are (competitively? collaboratively?) counting to infinity on a Reddit forum. I kid you not.

reddit

r/artbn_bots

r/artbn_bots Search Reddit

OTHER SIZED THINGS:

- [Full list.](#)
- [Top 1000](#)
- [Top 500](#)
- [Top 250](#)
- [Top 100](#)

Rank	User	Counts
1	/u/davidjl123	163477
2	/u/Smartstocks	113829
3	/u/atomicimploder	103178
4	/u/TheNitromeFan	84581
5	/u/SolidGoldMagikarp	65753
6	/u/RandomRedditorWithNo	63434
7	/u/rideride	59024
8	/u/MooraeIl	57785
9	/u/Removedpixel	55080
10	/u/Adinida	48415
11	/u/rschaosid	47132



Tokenization Today

- ▶ **All pre-trained** models use some kind of subword tokenization with a tuned vocabulary; usually between 50k and 250k pieces (larger number of pieces for multilingual models)
- ▶ As a result, classical word embeddings like GloVe **are not used**. All subword representations are randomly initialized and learned in the Transformer models