Contextualized Word Representations

Natural Language Processing

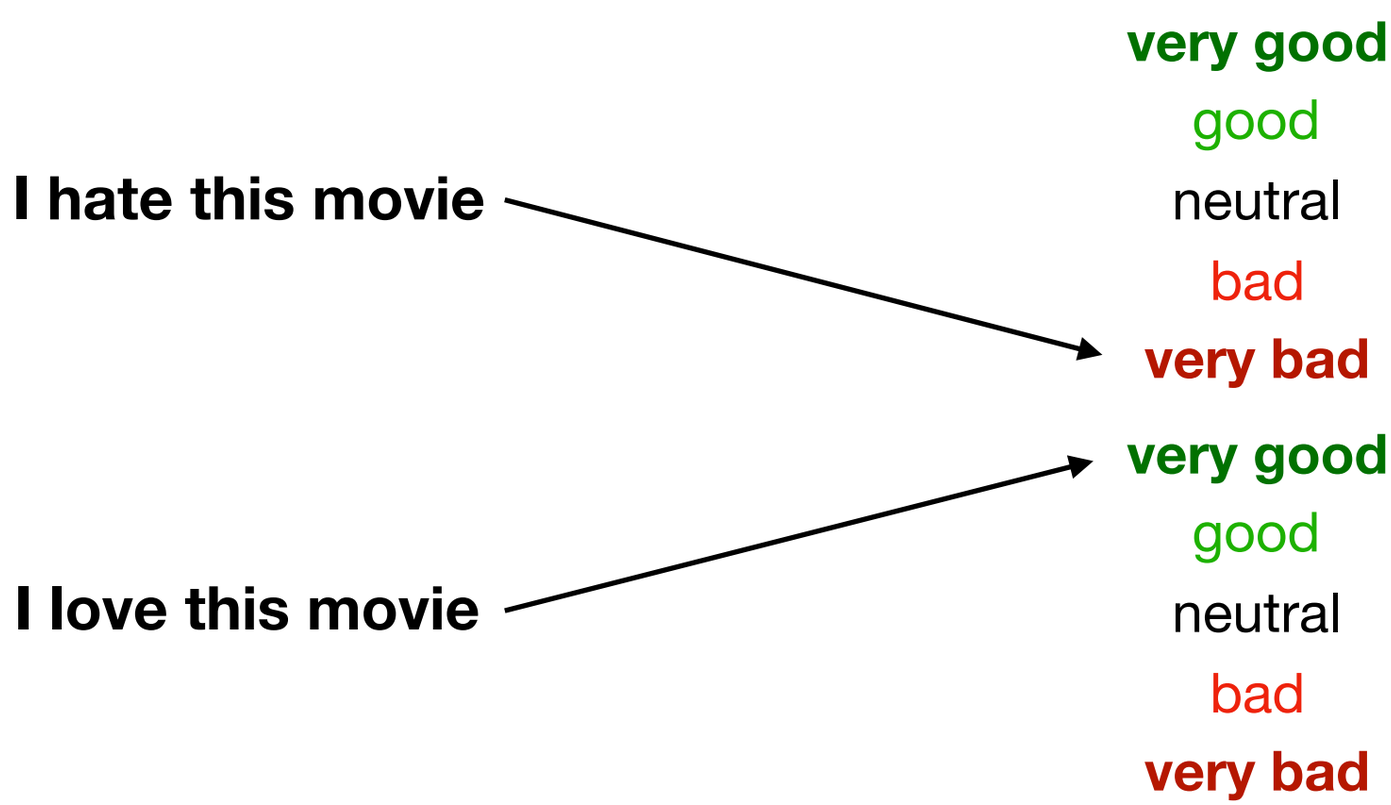
Based on slides from Stanford cs224n (Chris Manning and Abigail See), Graham Neubig and Ashish Vaswani

Slight Detour

- We have described recurrent architectures for language modeling
- But these architectures are more general purpose
- Let’s first see how they can be used for general classification tasks and expand to:
  - Bidirectional RNNs
  - Multi-layer RNNs
  - Transformers
- This will teach us about “contextualized word representations”
- We will then go back to language modeling and how it is used for large-scale representation learning of contextualized words

Sentence classification

- Classify sentences according to various traits
- Topic, sentiment etc.

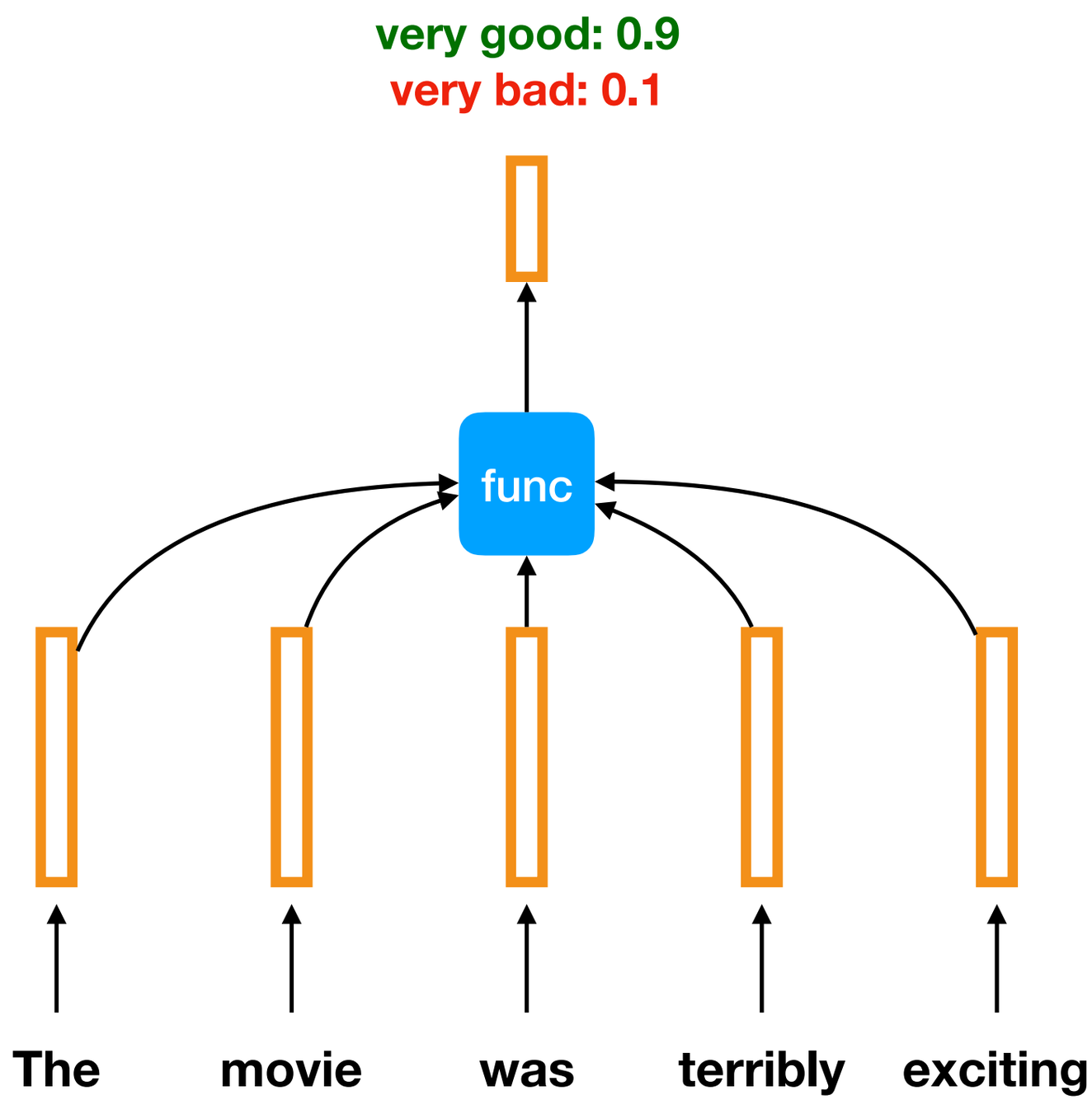


- It would be good to have a **sentence** representation

Sentence classification

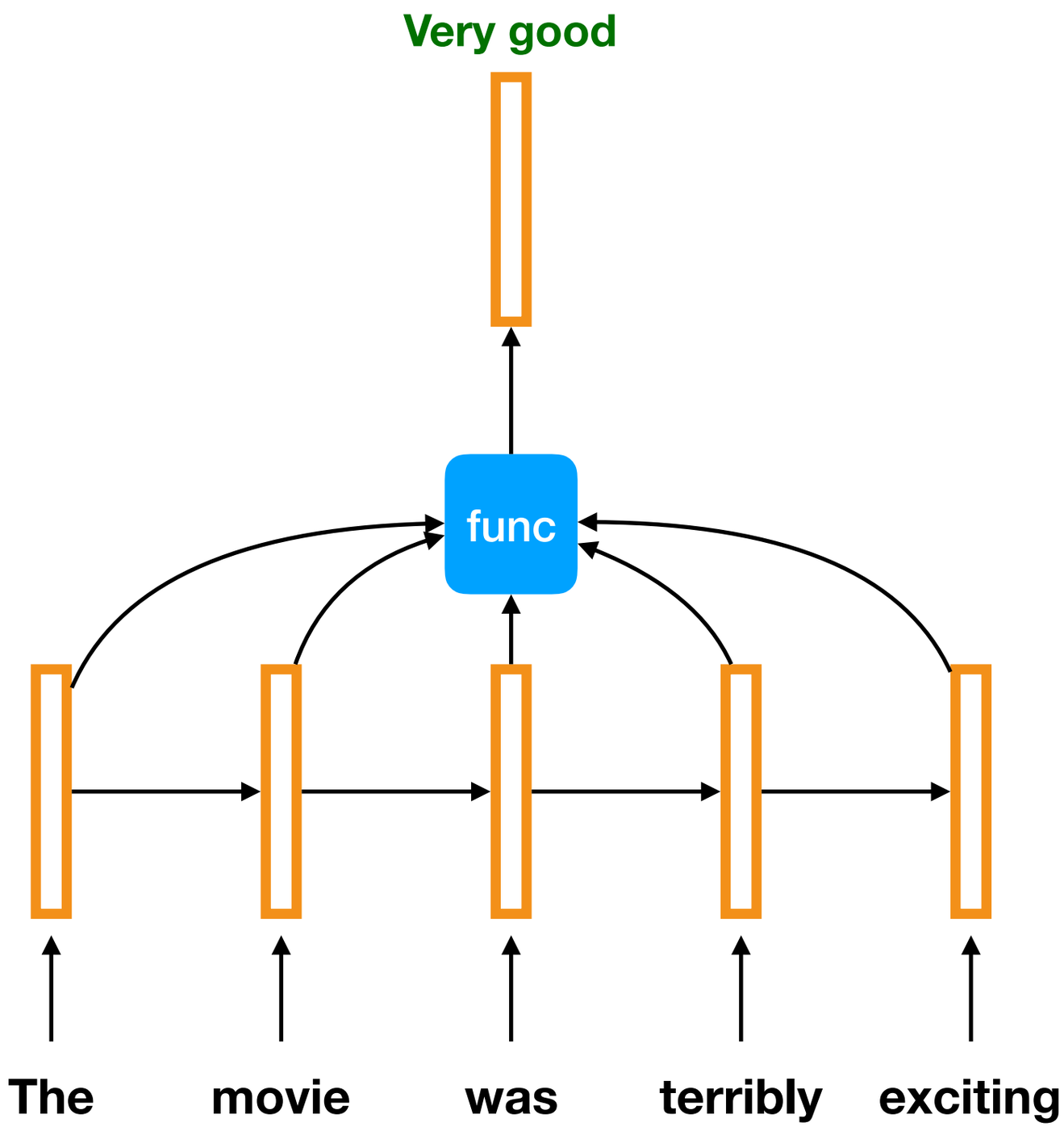
- Many tasks are variants of this:
  - Textual entailment
  - Textual similarity
  - …

Word representations



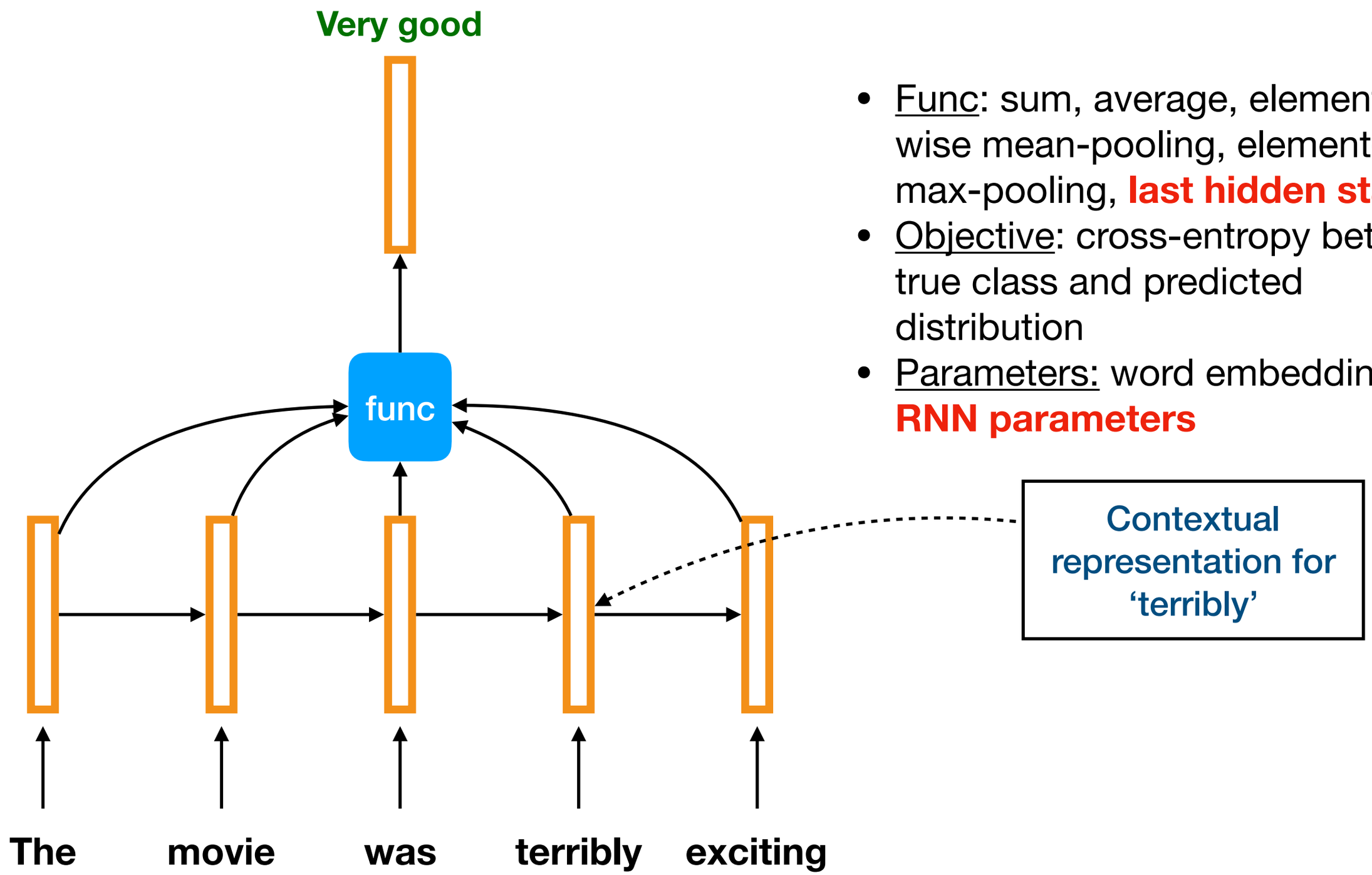
- Func: sum, average, element-wise mean-pooling, element-wise max-pooling
- Objective: cross-entropy between true class and predicted distribution
- Parameters: word embeddings

Contextualized word rep.



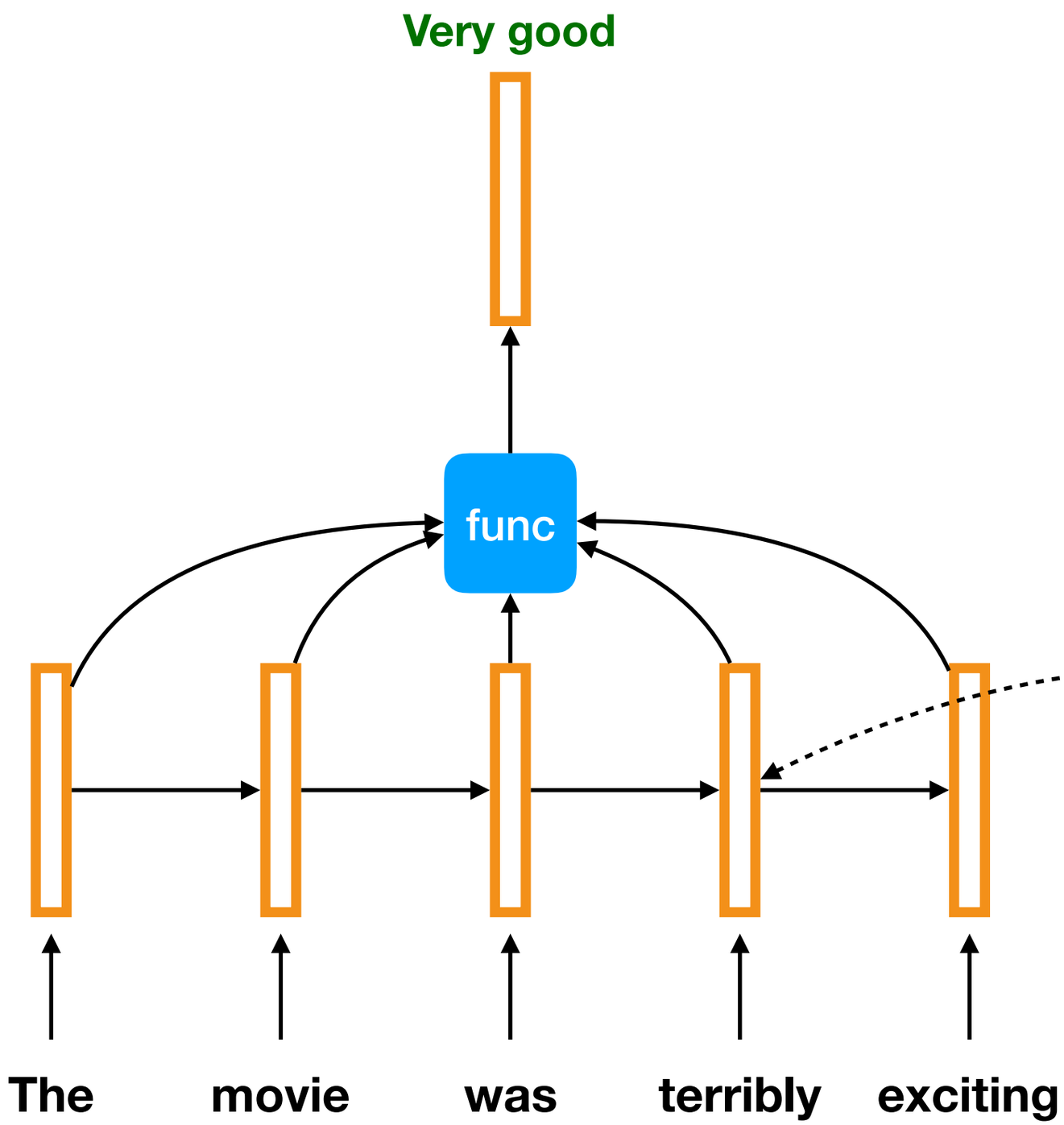
- Func: sum, average, element-wise mean-pooling, element-wise max-pooling, **last hidden state**
- Objective: cross-entropy between true class and predicted distribution
- Parameters: word embeddings **+ RNN parameters**

Contextualized word rep.



- Func: sum, average, element-wise mean-pooling, element-wise max-pooling, **last hidden state**
- Objective: cross-entropy between true class and predicted distribution
- Parameters: word embeddings **+ RNN parameters**

Contextualized word rep.

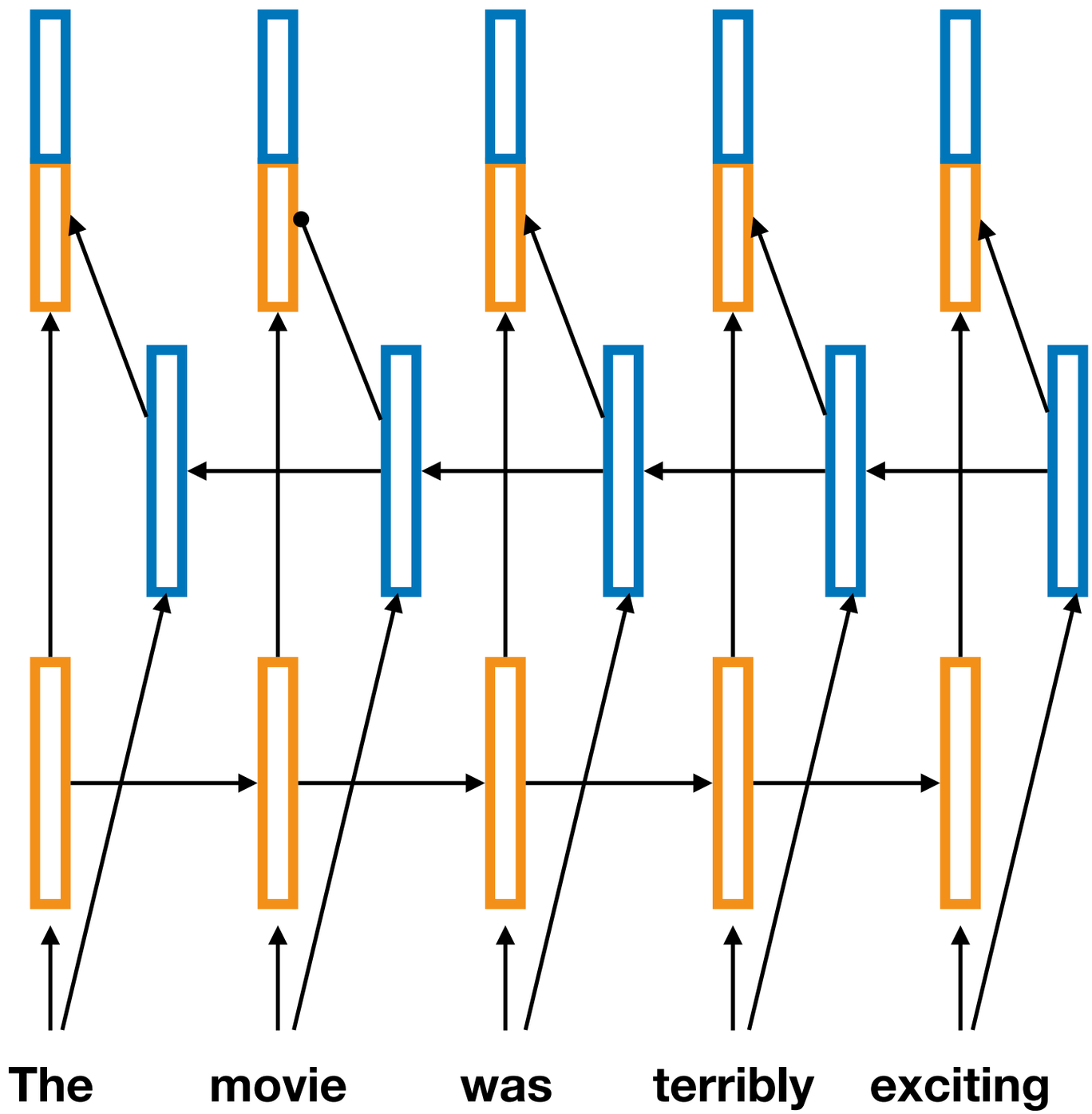


- Func: sum, average, element-wise mean-pooling, element-wise max-pooling, **last hidden state**
- Objective: cross-entropy between true class and predicted distribution
- Parameters: word embeddings **+ RNN parameters**

Contextual representation for 'terribly'

This contains information on left context only!

Bidirectional RNN



$$\overrightarrow{h}_t = \mathrm{RNN}_{\mathrm{fw}}(\overrightarrow{h}_{t-1}, x_t)$$

$$\overleftarrow{h}_t = \mathrm{RNN}_{\mathrm{bw}}(\overleftarrow{h}_{t+1}, x_t)$$

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$$

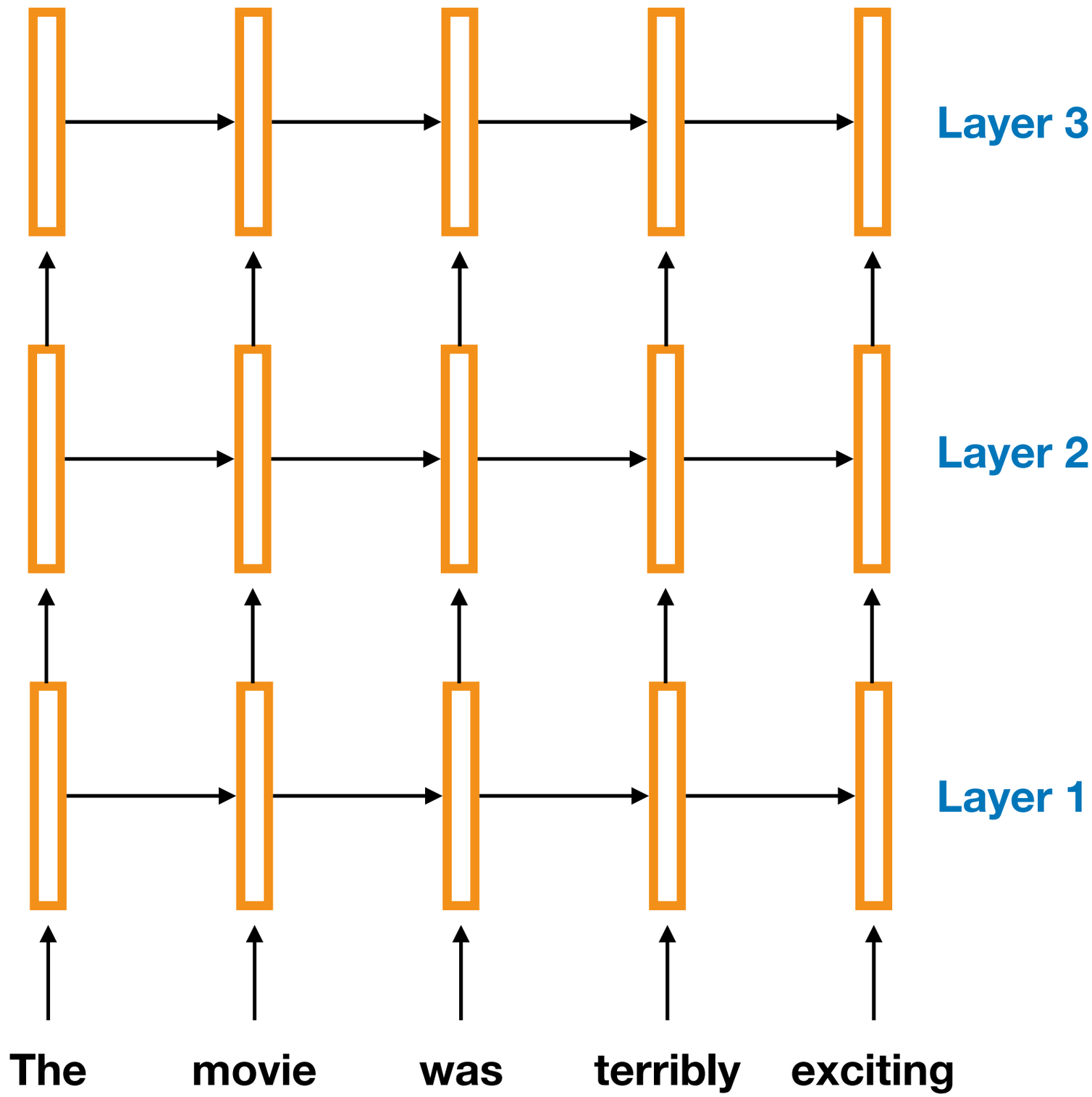
Bidirectional RNN

- RNN here can be LSTM/GRU/whatever
- Each hidden state is now contextualized with both left and right context
- Parameters for forward and backward RNNs are separate
- **These are applicable for the sentence classification task but not for generative LMs where only the history is observed.**
- In general, using both sides improves performance by a lot!
  - **Was the greatest hit of NLP in 2017!**

Multi-layer RNN

- We can make RNNs deeper by having the input of each token in one RNN be the input to a deeper RNN

- These are called multi-layer RNN or stacked RNNs.

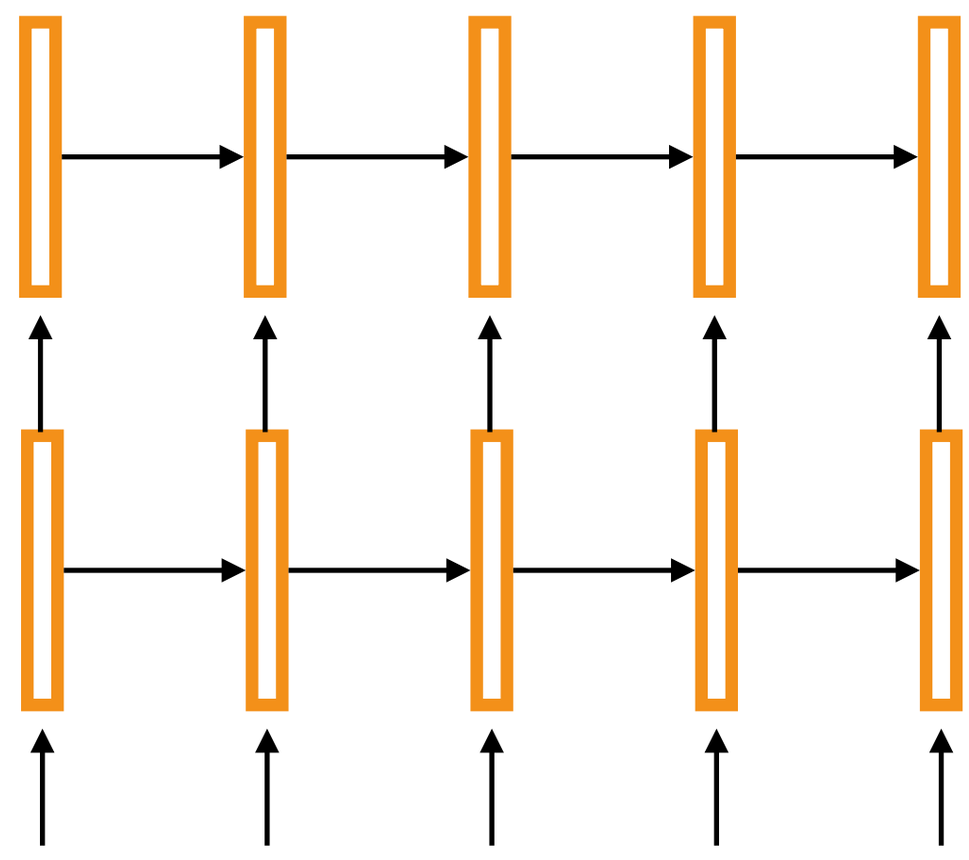


Hidden state/output state of layer *i* is the input to layer *i+1*

Vaswani et al., 2017

Transformers

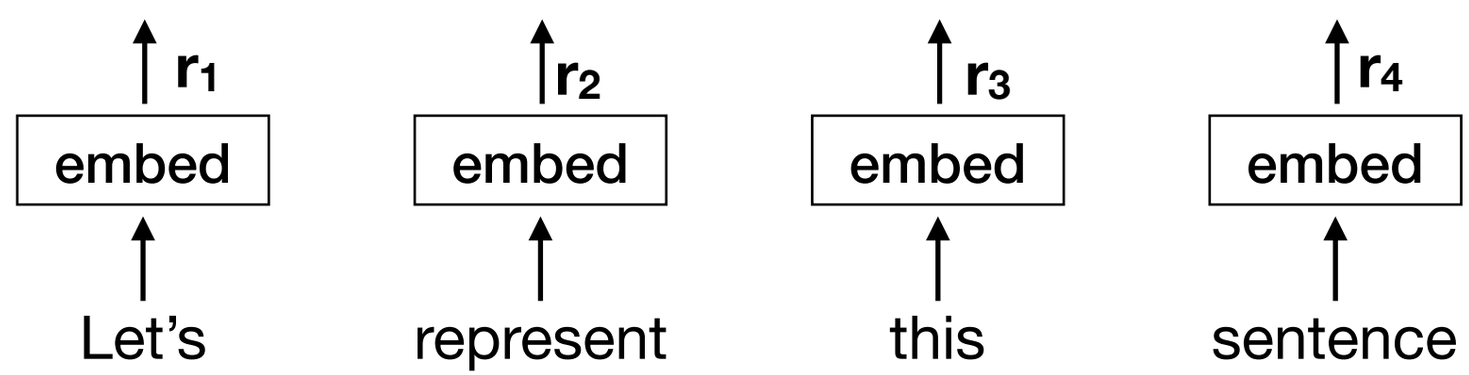
- **Motivation:**
  - RNNs are hard to parallelize so it takes a lot of time to run them on large amounts of data
  - They also don't explicitly model long and close range dependencies
  - Can we get contextualized representations in a way that overcomes these drawbacks?
    - Yes! Attention



Approximate Transformer Block

- We want to re-represent r_2 as a function of context:

$$r_i = W_{\text{embed}} \cdot e_i$$

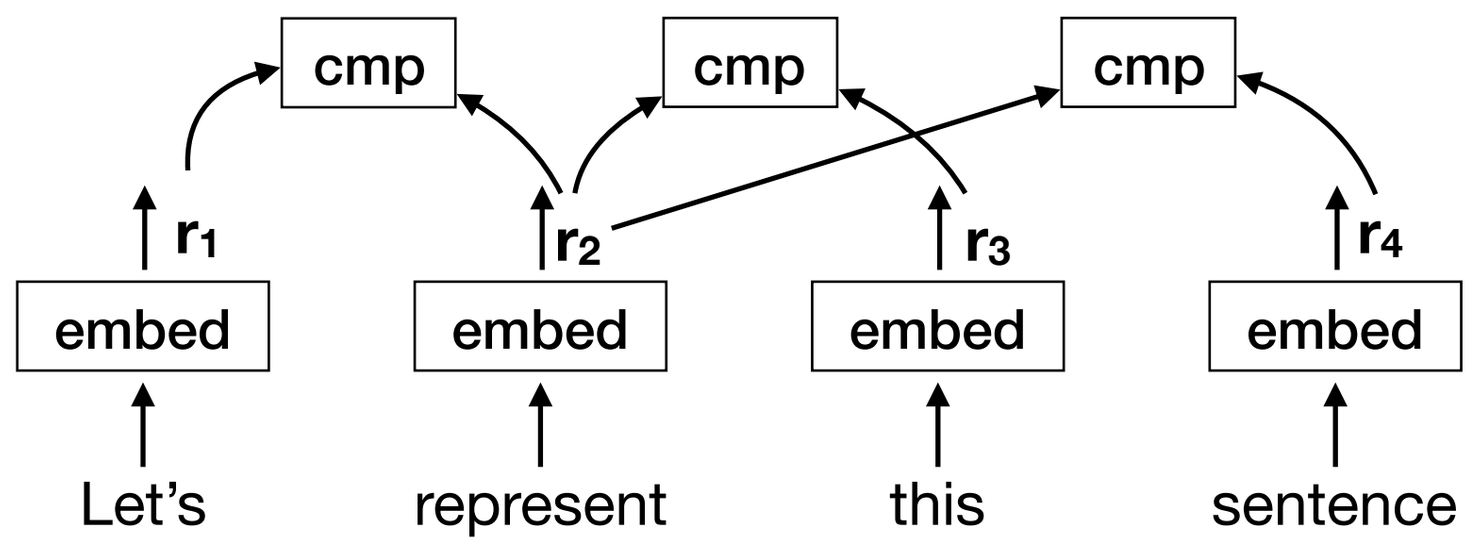


Approximate Transformer Block

- We want to re-represent r_2 as a function of context:

$$s_{ij} = r_i^\top r_j$$

$$r_i = W_{\text{embed}} \cdot e_i$$



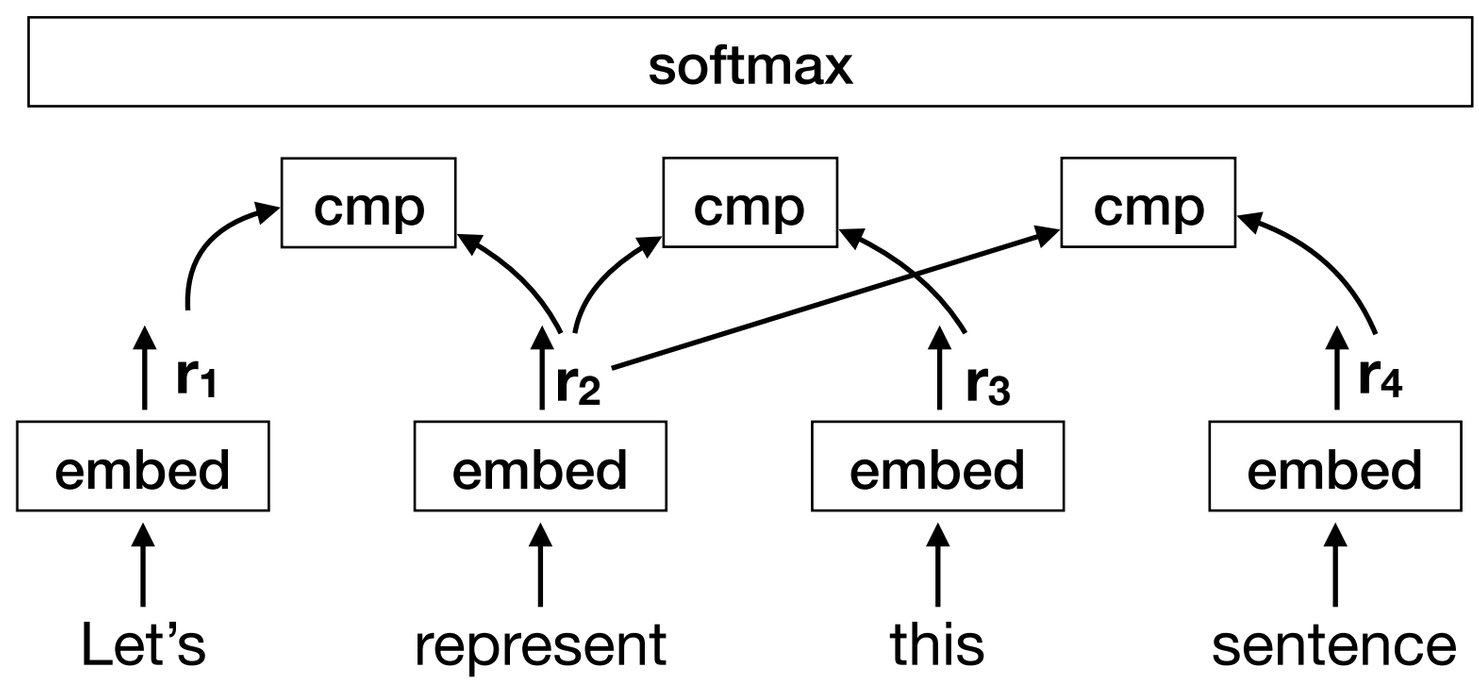
Approximate Transformer Block

- We want to re-represent r_2 as a function of context:

$$p_{ij} = \mathrm{softmax}(s_{i1}, \ldots, s_{i|x|})$$

$$s_{ij} = r_i^\top r_j$$

$$r_i = W_{\mathrm{embed}} \cdot e_i$$



Approximate Transformer Block

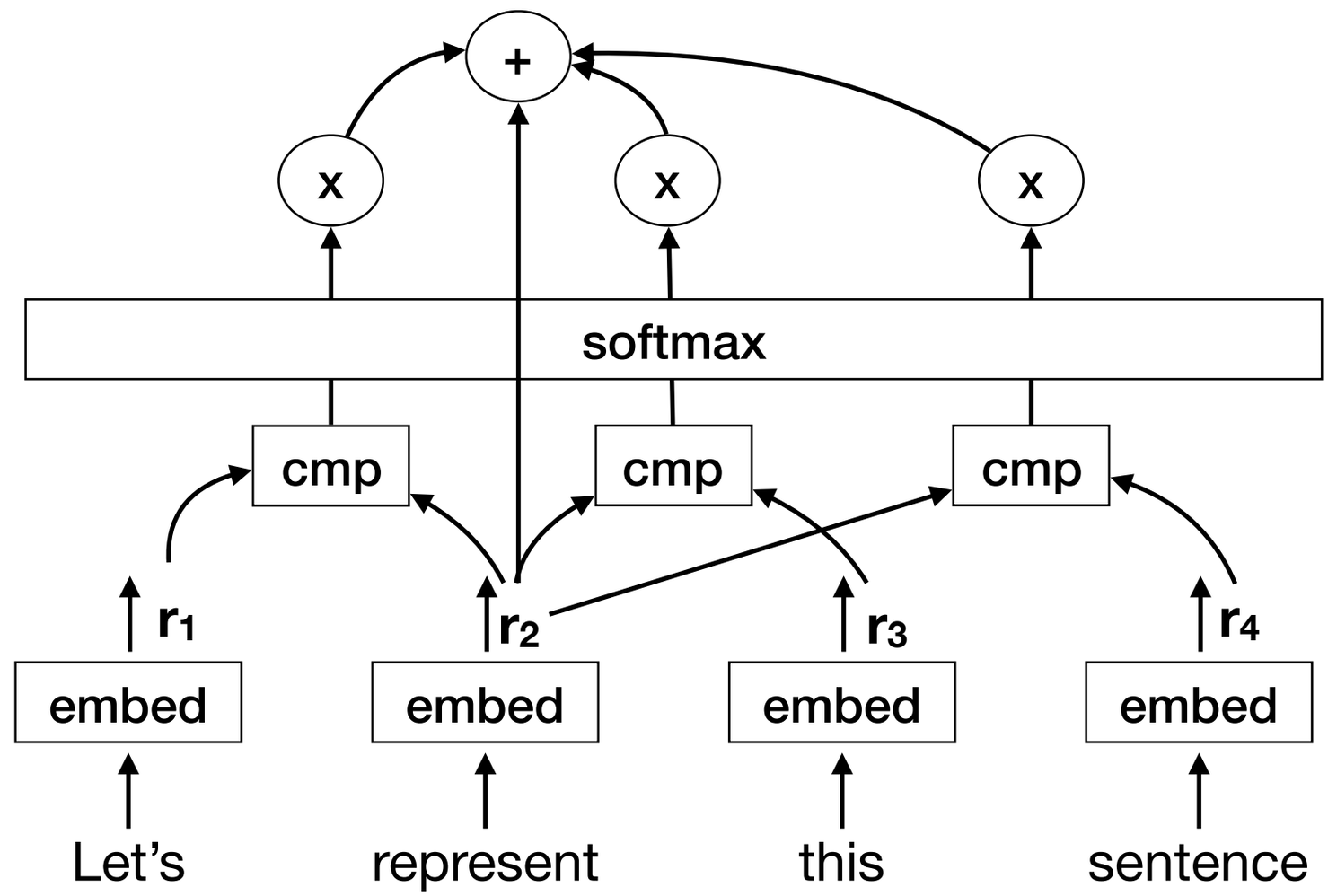
- We want to re-represent r_2 as a function of context:

$$o_i = \sum_j p_{ij} r_j$$

$$p_{ij} = \mathrm{softmax}(s_{i1}, \ldots, s_{i|x|})$$

$$s_{ij} = r_i^\top r_j$$

$$r_i = W_{\mathrm{embed}} \cdot e_i$$



Approximate Transformer Block

- We want to re-represent r_2 as a function of context:

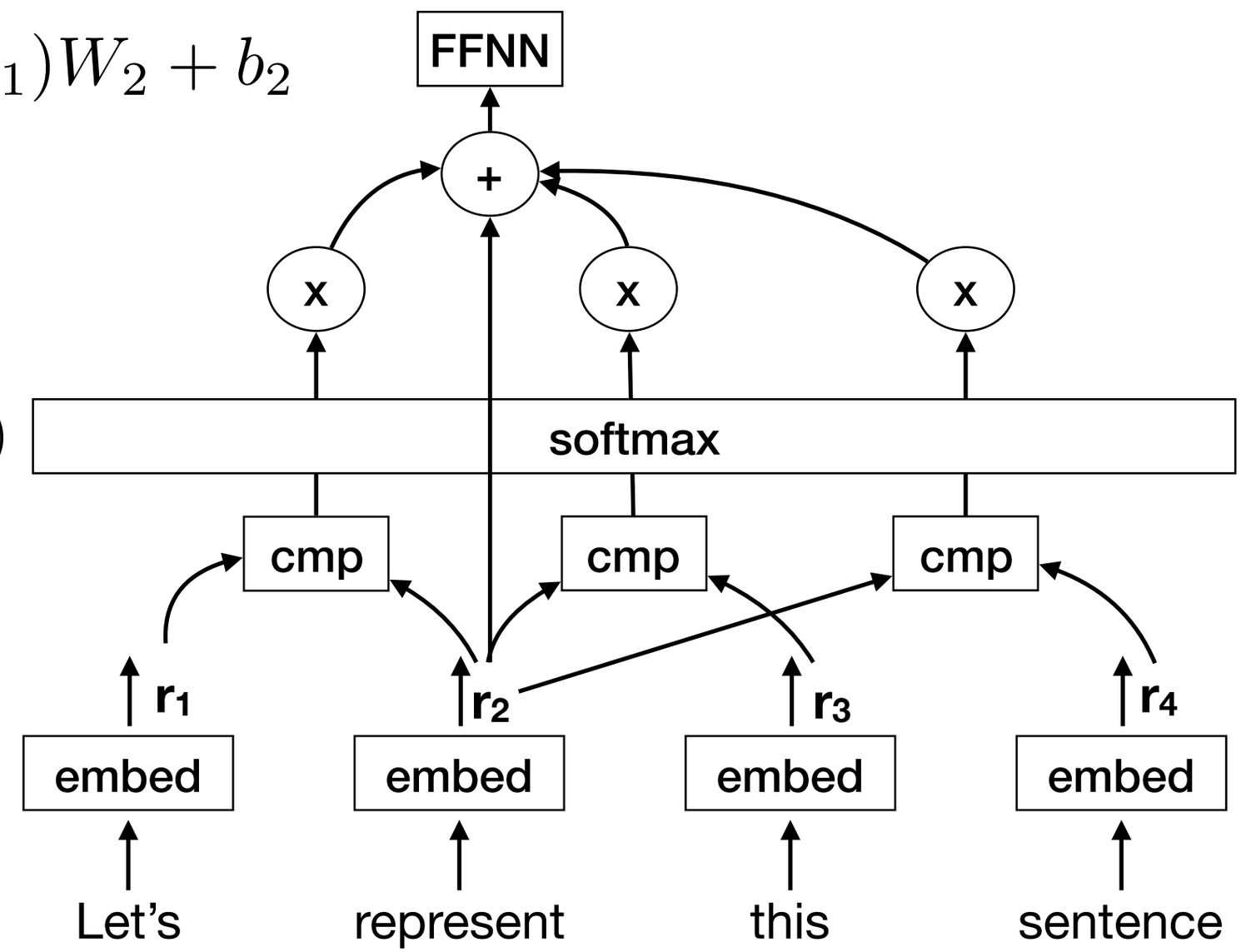
$$\mathrm{FFNN}(o) = \mathrm{relu}(W_1 o + b_1) W_2 + b_2$$

$$o_i = \sum_j p_{ij} r_j$$

$$p_{ij} = \mathrm{softmax}(s_{i1}, \ldots, s_{i|x|})$$

$$s_{ij} = r_i^\top r_j$$

$$r_i = W_{\mathrm{embed}} \cdot e_i$$



Approximate Transformer Block

- We want to re-represent r_2 as a function of context:

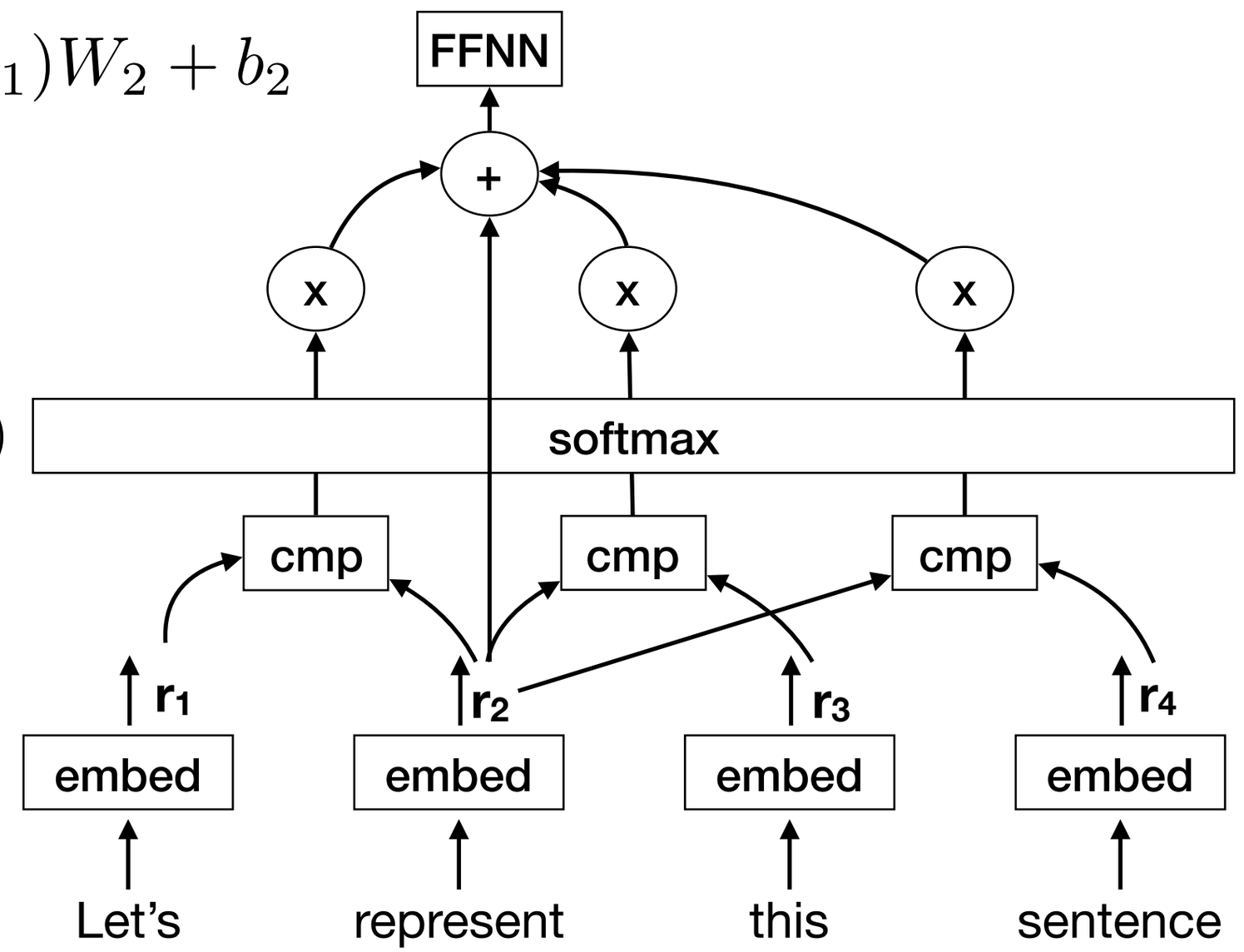
$$\mathrm{FFNN}(o) = \mathrm{relu}(W_1 o + b_1) W_2 + b_2$$

$$o_i = \sum_j p_{ij} r_j$$

$$p_{ij} = \mathrm{softmax}(s_{i1}, \ldots, s_{i|x|})$$

$$s_{ij} = r_i^\top r_j$$

$$r_i = W_{\mathrm{embed}} \cdot e_i$$



(a) This is fast and parallelizable! (b) words attend to entire context!

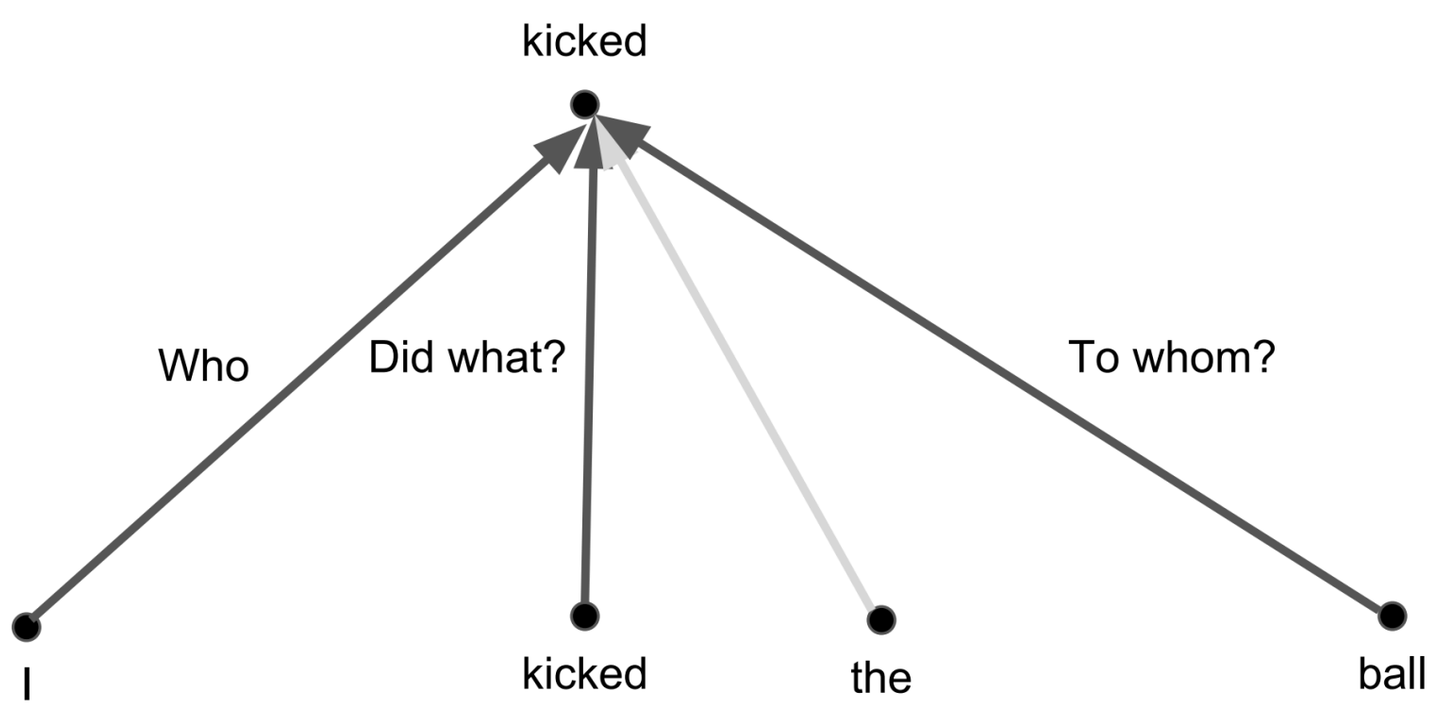
Approximate transformer

- This architecture is just a few matrix multiplications in parallel
- All pairs of words attend to one another

$$A(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Details: Multi-head Attention

- Problem: Each word attends to a single location in previous layer
- Solution: Have *K* attention heads



Detail: Multi-head Attention

- In our case Q, K, V are all just the representations of the inputs to the transformer block

$$\mathrm{MultiHeadAtt}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)W_3$$
$$\mathrm{head}_i = A(QW_q^i, KW_k^i, VW_v^i)$$

- We use the linear projections W_q, W_k, W_v to reduce the dimensionality of the input by factor of h, perform attention h times, and then concatenate the output to obtain a representation that has the same dimensionality as the input

Detail: Residual Connections and Normalization

- We can call both the attention layer and the feed-forward network that follows it a sublayer.

- The output of each sublayer is defined as: *LayerNorm(x + sublayer(x)*

- *x + sublayer(x)* is a “residual connection”. It allows gradients to flow in deep networks similar to what we saw in LSTMs and GRUs

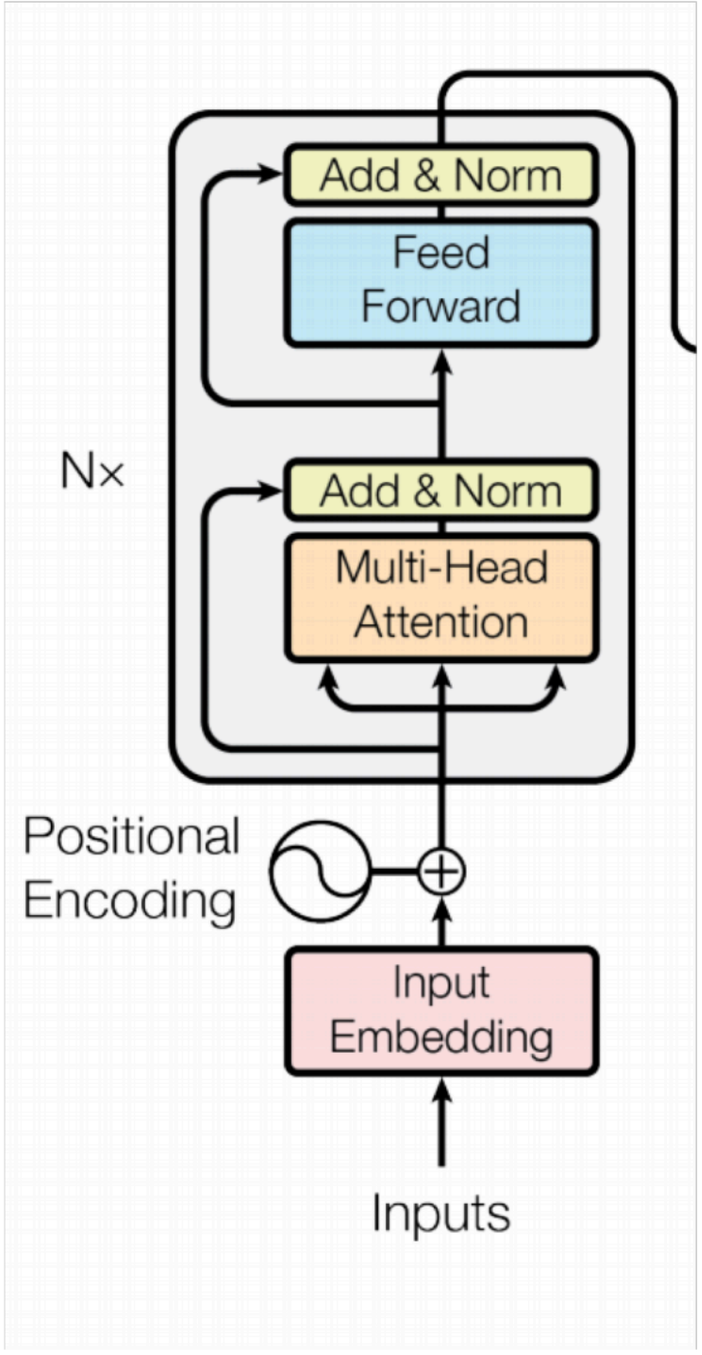
- *LayerNorm* normalizes the inputs to the layer and improves optimization

Detail: Positional embeddings

- Problem:
  - Everything was permutation invariant so far!
  - You can take a training corpus, and apply a fixed permutation to every sentence and you will get the same result
  - Order seems to matter in language
- Solution: add positional embeddings
  - Input to (first) transformer block is $r\_i + p\_i$, where p_i is a representation of the integer i.

Transformer Summary

- Blocks are stacked (6, 12, 24 times)
- The encoding of a word is the output of the top-most layer
- This representation is very contextualized and good for many tasks

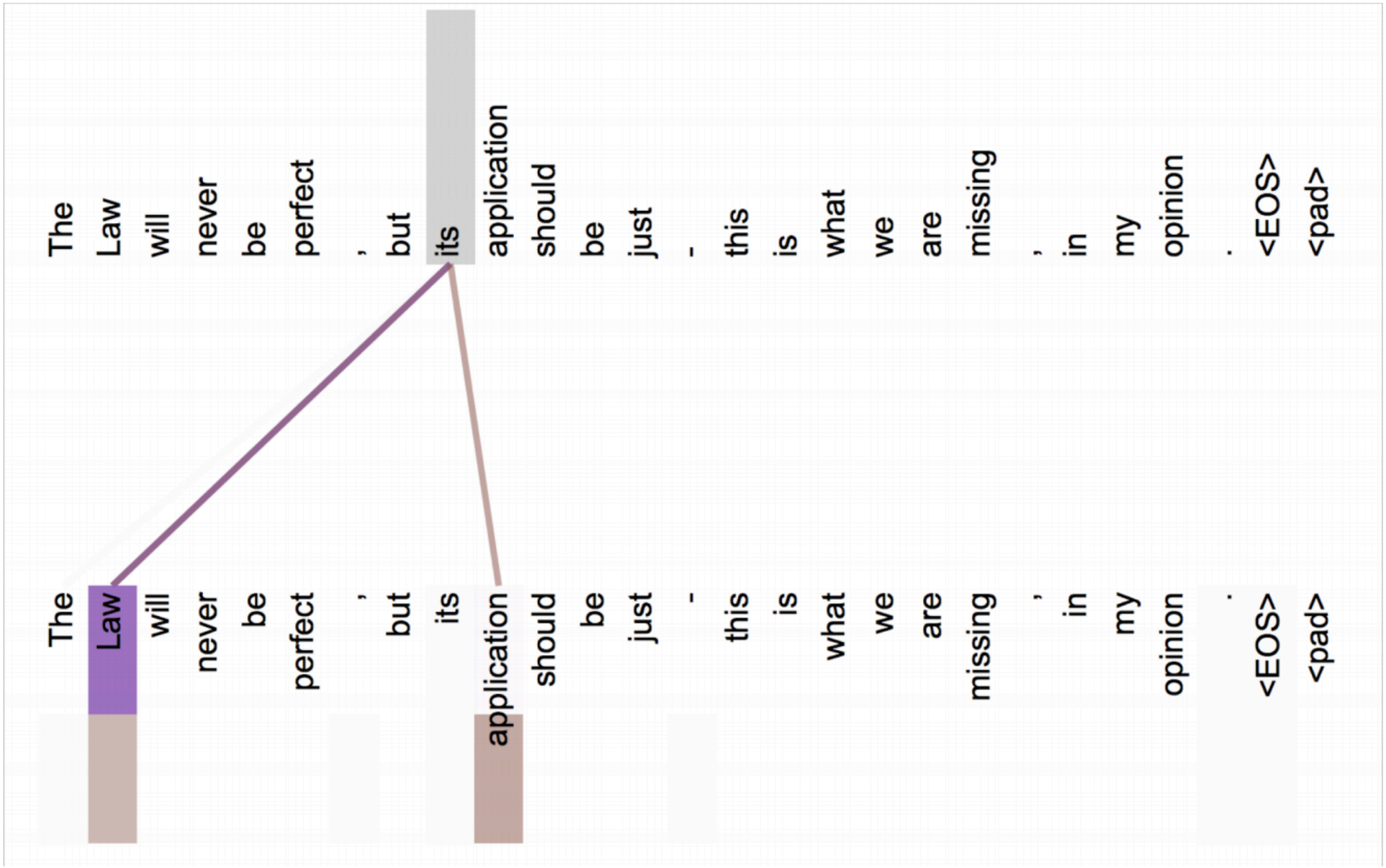


Results: Machine Translation

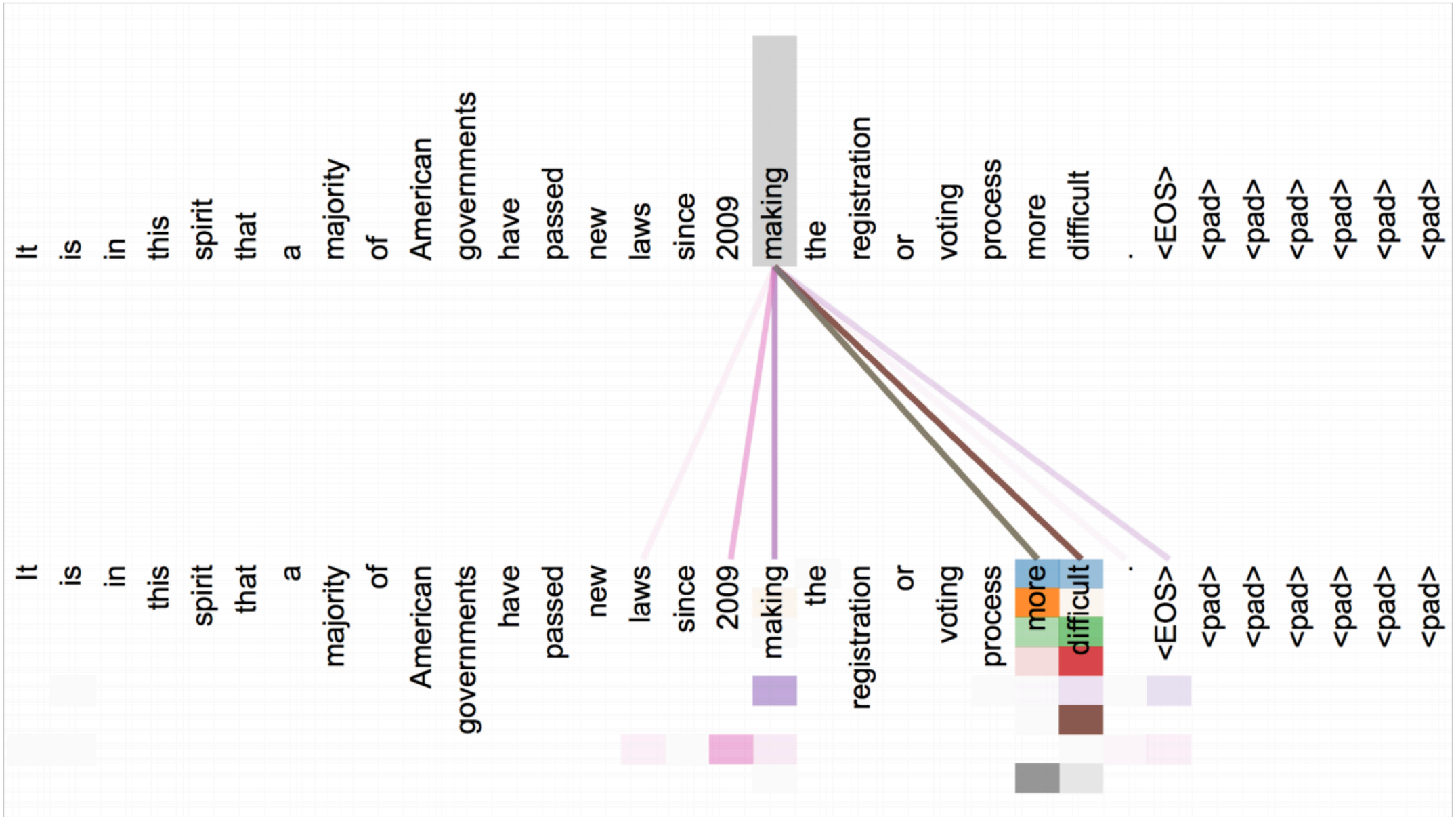
	EN-DE	EN-FR
GNMT (orig)	24.6	39.9
ConvSeq2Seq	25.2	40.5
Transformer*	**28.4**	**41.8**

*Transformer models trained >3x faster than the others.

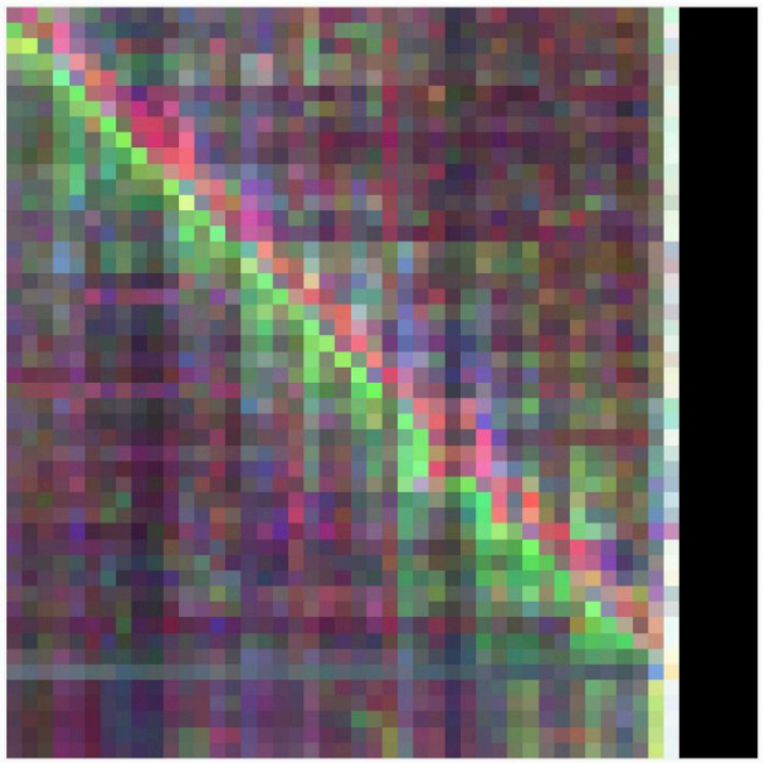
Attention Visualization



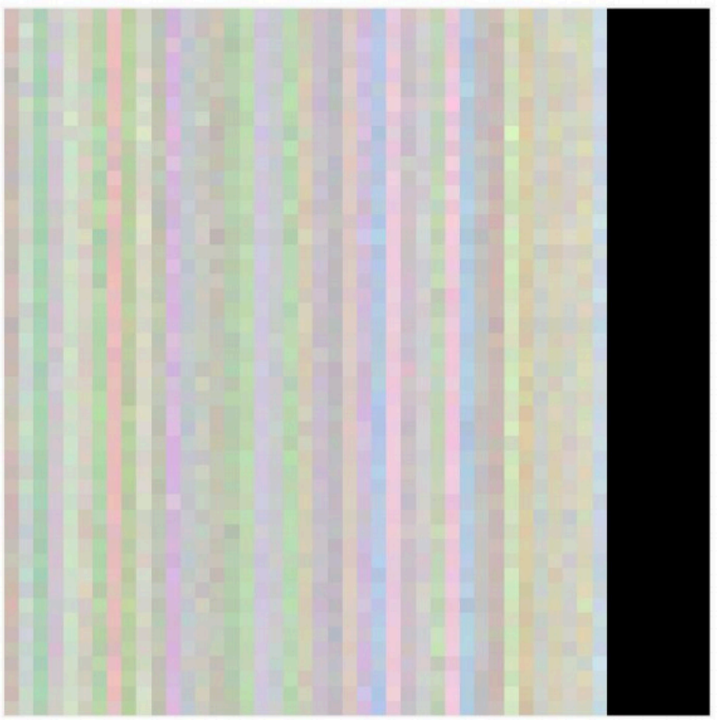
Attention Visualization



Importance of Residuals



With residuals



Without residuals

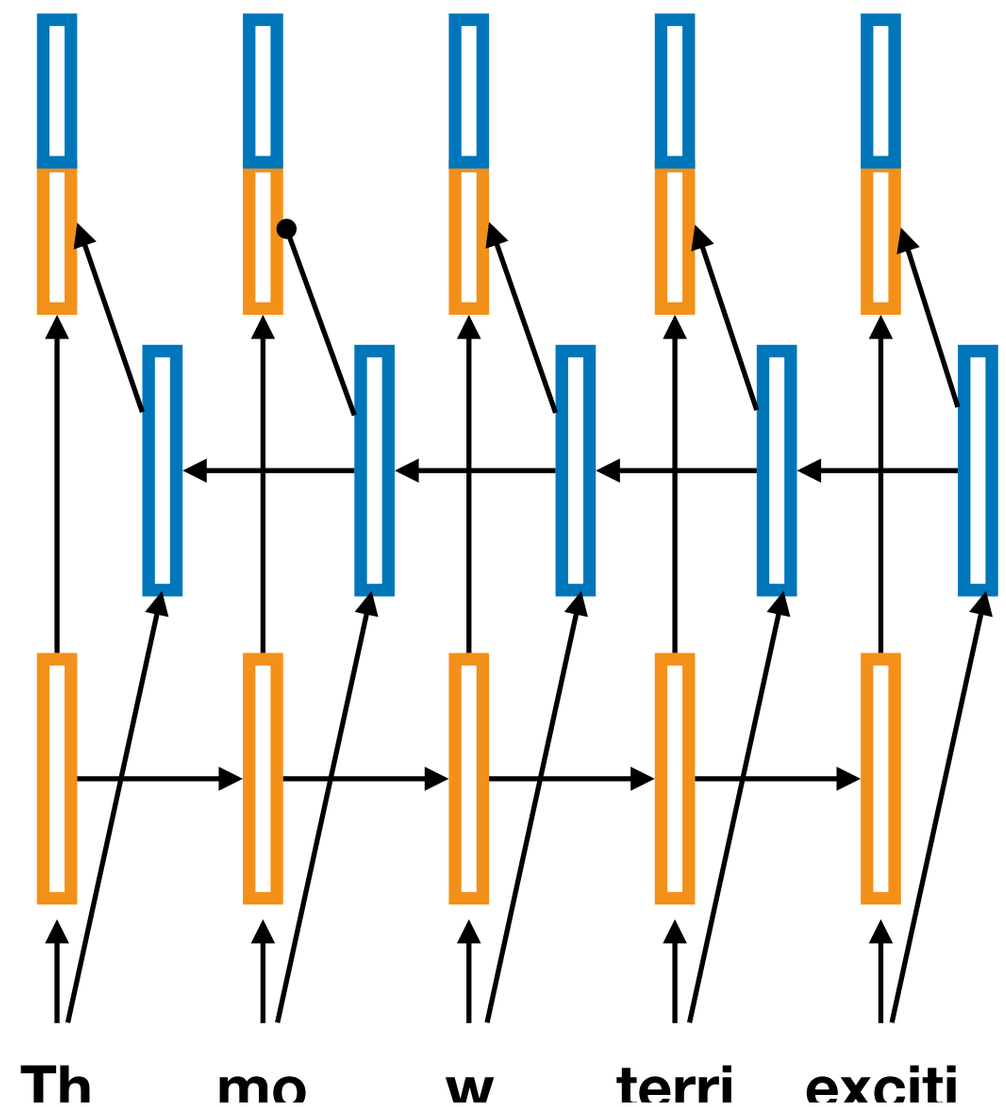
Pre-training Contextualized Word Representations

Thought

- Pre-trained word representations are nice:
  - Can be trained on tons of data
- But:
  - They don’t take context into account
- Can we pre-train representations for words in context on tons of data and use them like word embeddings?
  - **YES! Let’s just use language models!**

TagLM + ELMO (Peters et al. 2017, 2018)

- Train a forward LM and a backward LM (L-Layers)
  - In practice, 2-layer LSTM
- This gives for every for every token at index *k*, a *2L+1* representation: *2L* are contextualized and one is not.



ELMO

- How are these representations used?
- Learn how to combine the different layers per task

$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

$$
\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.
$$

- s^{task}: softmax normalized logits over the layers
- γ^{task}: scaling factor for the entire ELMO vector
- The vector per token can be used by any task

ELMO Results

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%

Radford et al., 2018

GPT-2

- Replaced LSTM with large transformer
- Trained on tons of text
- Got the beautiful unicorn example we saw last time

BERT

- BERT: Bidirectional Encoder Represenations from Transformers (Devlin et al., 2019)

- <u>Problem</u>:

  - LM are uni-directional, but our understanding of language can be based on both directions

  - Reason: LM are a generative model defining a probability distribution for language

    - But we don’t care about that when we just want to learn good contextualized word representations.

BERT

- <u>Change the objective</u>: predict a word based on both left and right context (**masked LM,** cross-entropy loss)

- Randomly mask k(=15%) of the words and predict them from the context

store **gallon**

The man went to the [mask] to buy a [mask] of milk.

- Too little masking: expensive to train

- Too much masking: Not enough context

BERT

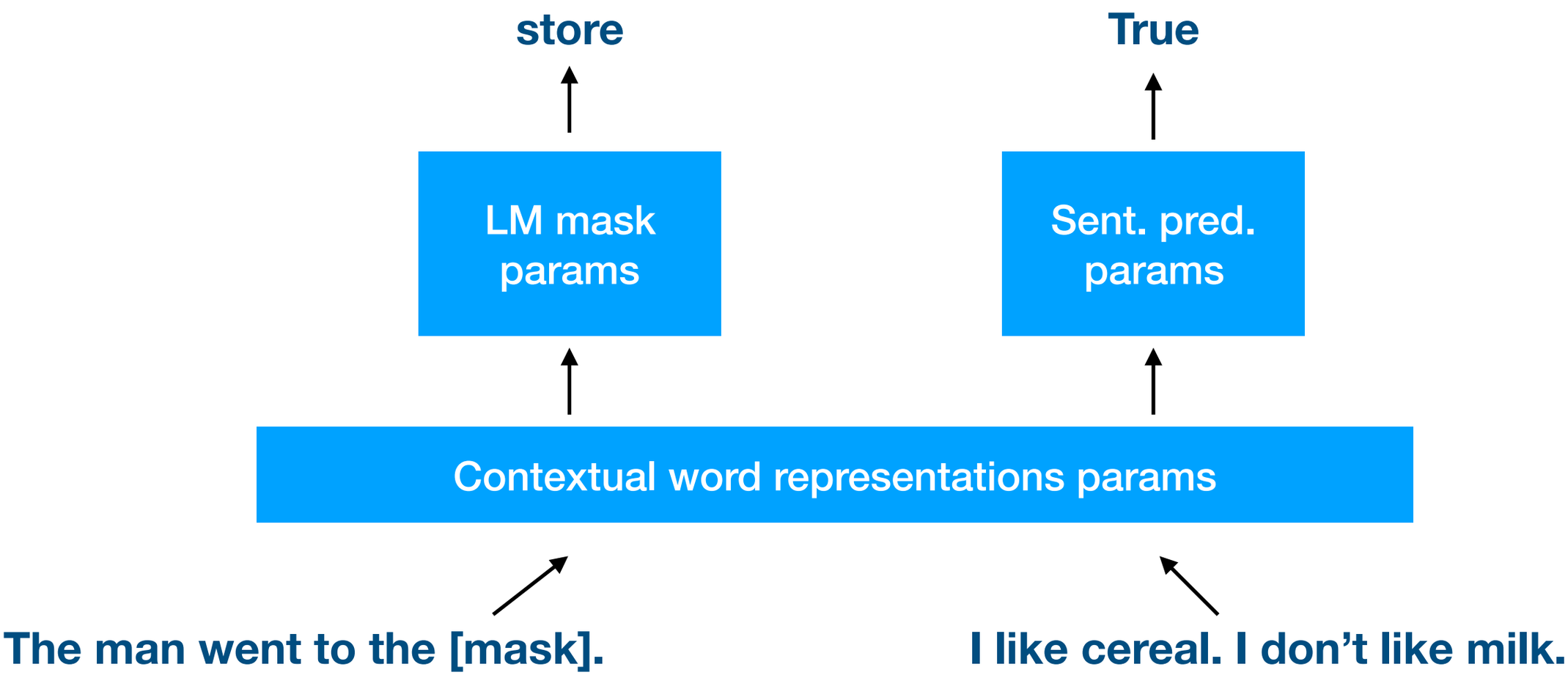
- Change the objective: guess the next sentence

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

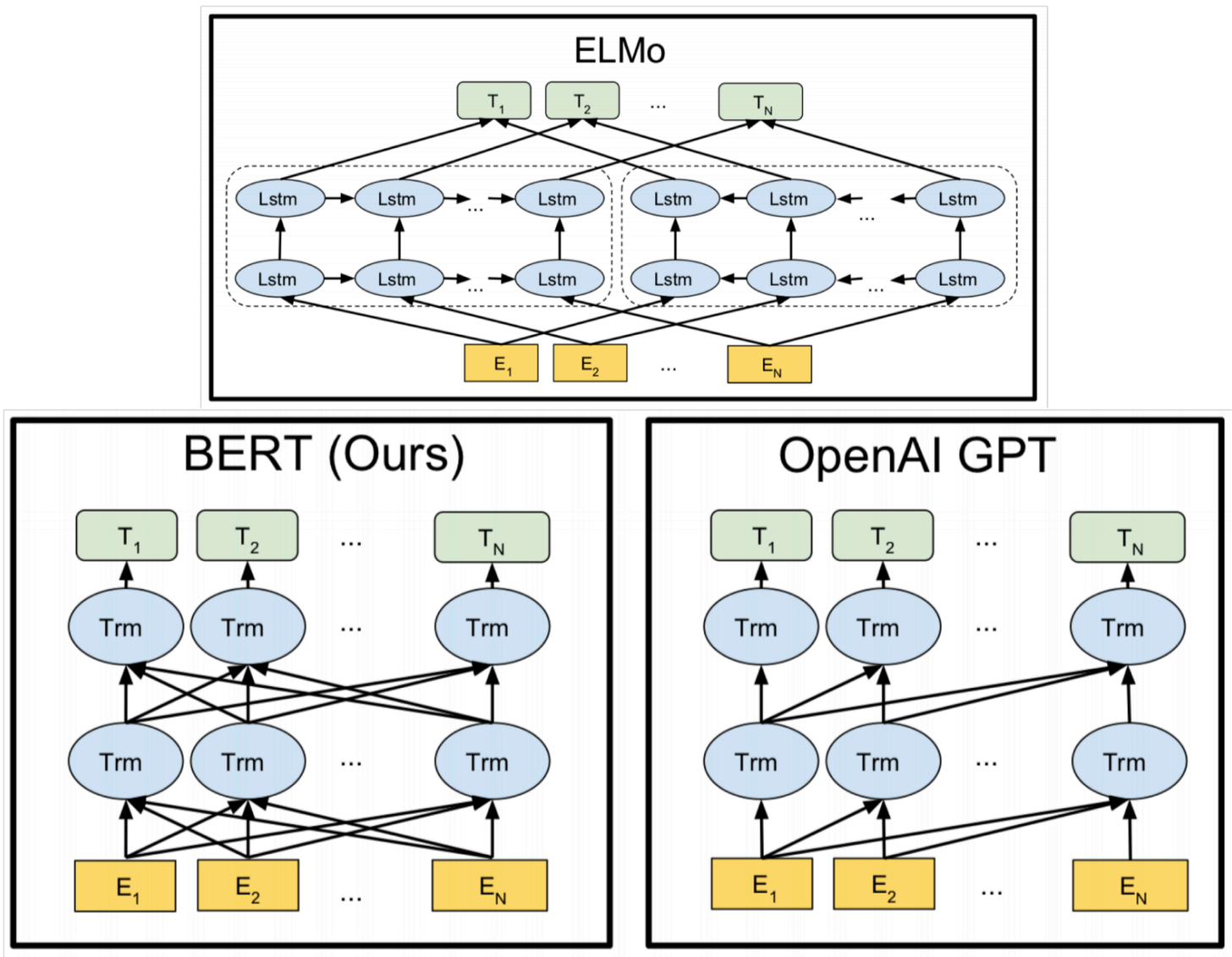
BERT

- <u>Multi-task objective:</u>
  - The contextualized word representations are computed for both objectives (masked LM and sentence guessing) with the same weights



BERT

- Change the model: replace two **unidirectional** RNNs with **bidirectional** transformer

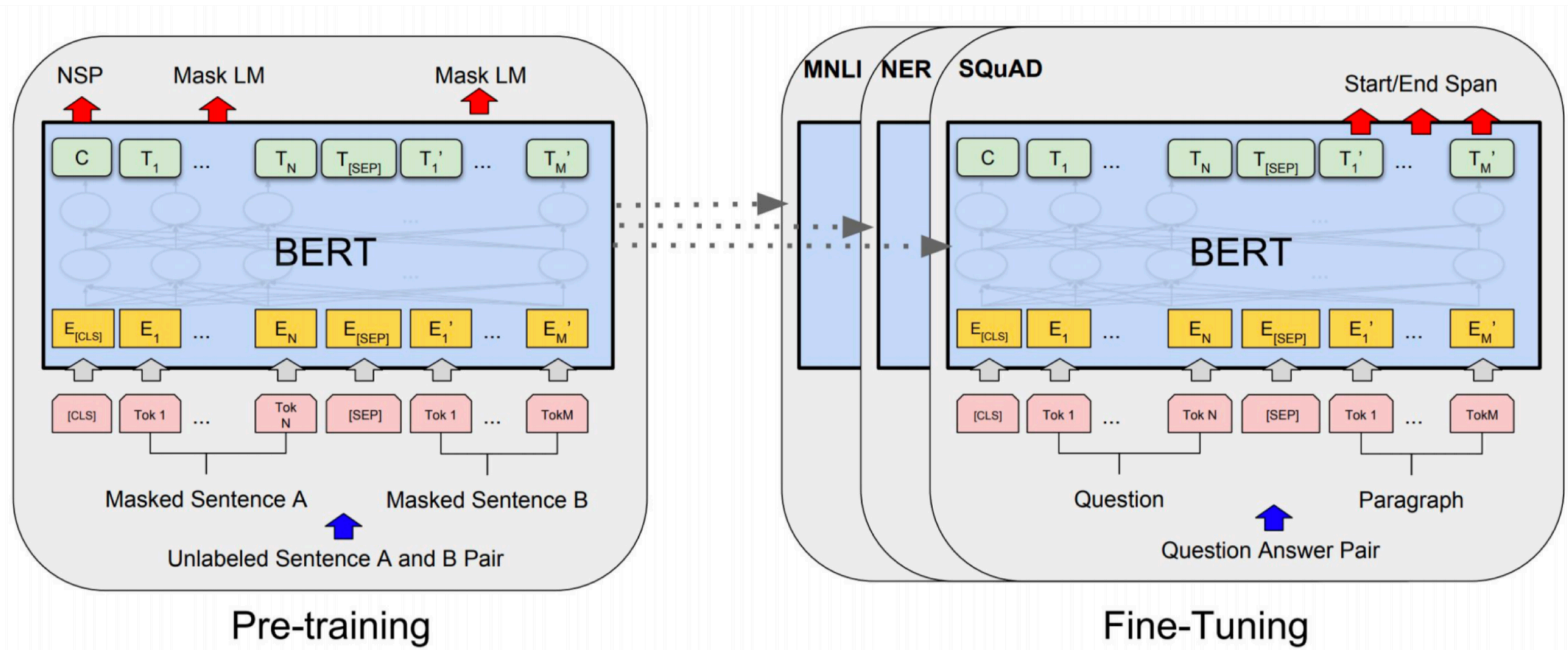


BERT Details

- Transformer architecture
  - BERT-base: 12 layers, 768 hidden, 12-head attention
  - BERT-large: 24 layers, 1024-hidden, 16-head attention
- Trained on Wikipedia + BookCorpus
- BERT-large takes some time to train…

BERT Fine-tuning

- The entire BERT model is fine-tuned for downstream tasks with very little parameters added



BERT Sentence Classification Results

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

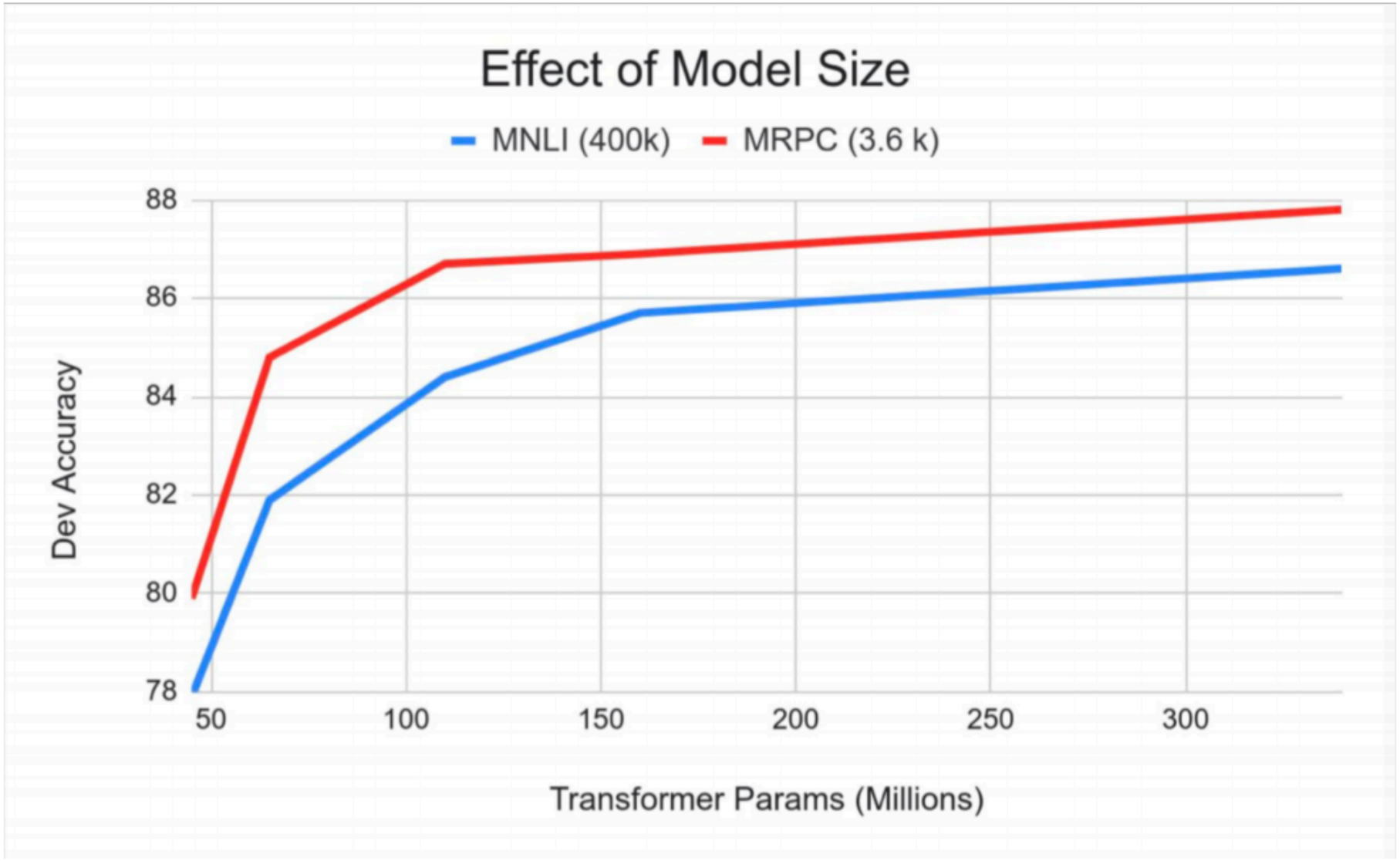
BERT QA Results

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	**87.433**	**93.160**
2 Oct 05, 2018	BERT (single model) *Google AI Language* https://arxiv.org/abs/1810.04805	85.083	91.835
2 Sep 26, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.954	91.677
5 Sep 09, 2018	nlnet (single model) *Microsoft Research Asia*	83.468	90.133
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490

BERT QA Results

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

Model Size


Effect of Model Size
MNLI (400k)
MRPC (3.6 k)
88
86
84
82
80
78
Dev Accuracy
50
100
150
200
250
300
Transformer Params (Millions)

Since BERT…

- RoBERTa: The same as BERT but trained better and on more data
  - Substantially better!
- Albert: The same as BERT but model is smaller by tying transformer layers
  - Again - substantially Better
- SenseBERT, SpanBERT, …

Disclaimer

- I glossed over a lot of the details
  - Exact details of training/pre-training/multi-task learning
  - Sentences are represented with word-pieces and characters, not words
  - ...
- If you are interested read the paper and think of a project!
- But this covered the main ideas - which are simple!

History

- Variants of this was proposed before ELMO, but the true power was cleanly shown in the ELMO paper
  - context2vec (Melamud et al., 2016)
  - CoVe (McMann et al., 2017)
  - ULMfit (Howard and Ruder, 2018)

Scaling

ULMfit	GPT	BERT	GPT-2
Jan 2018	June 2018	Oct 2018	Feb 2019
Training:	Training	Training	Training
1 GPU day	240 GPU days	256 TPU days	~2048 TPU v3 days according to a reddit thread
		~320–560 GPU days	

Summary

- Contextualized representations are replacing word representations
- Can be trained on tons of data
- Tuned for downstream tasks
- Lead to large improvements
- Based on LM in two ways:
  - Trained often from a LM-like objective
  - Use architectures from the LM world
- <u>Recommended reading</u>