Attention, Self-attention, Transformers, BERT

Manaal Faruqui
Google Assistant

Content Borrowed From:
Ankur Parikh,
Jacob Devlin,
Ashish Vaswani

Attention

Neural Machine Translation

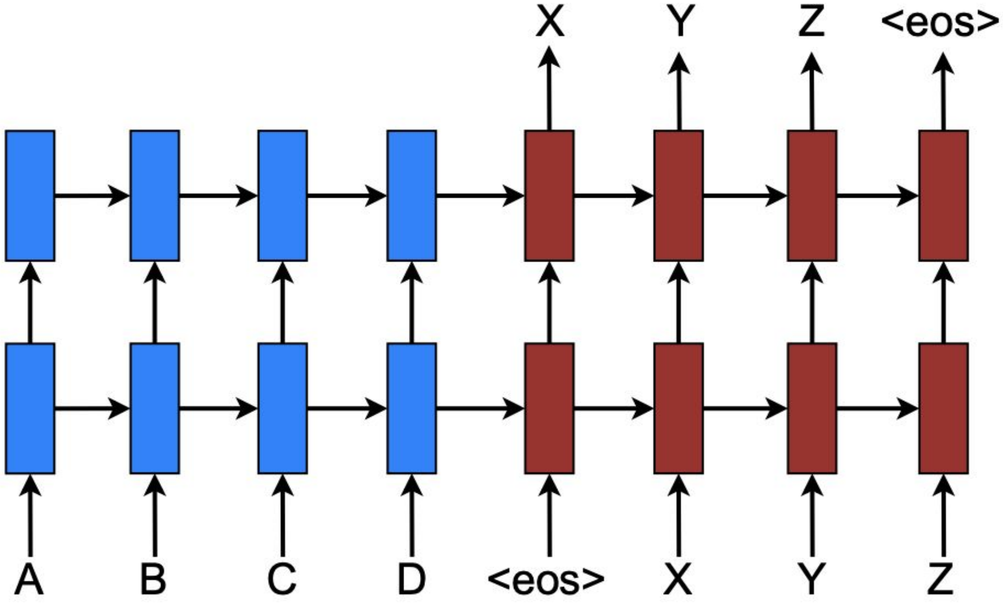


Image: Luong et al (2015)

Sutskever et al (2014)

Attention in Neural Networks

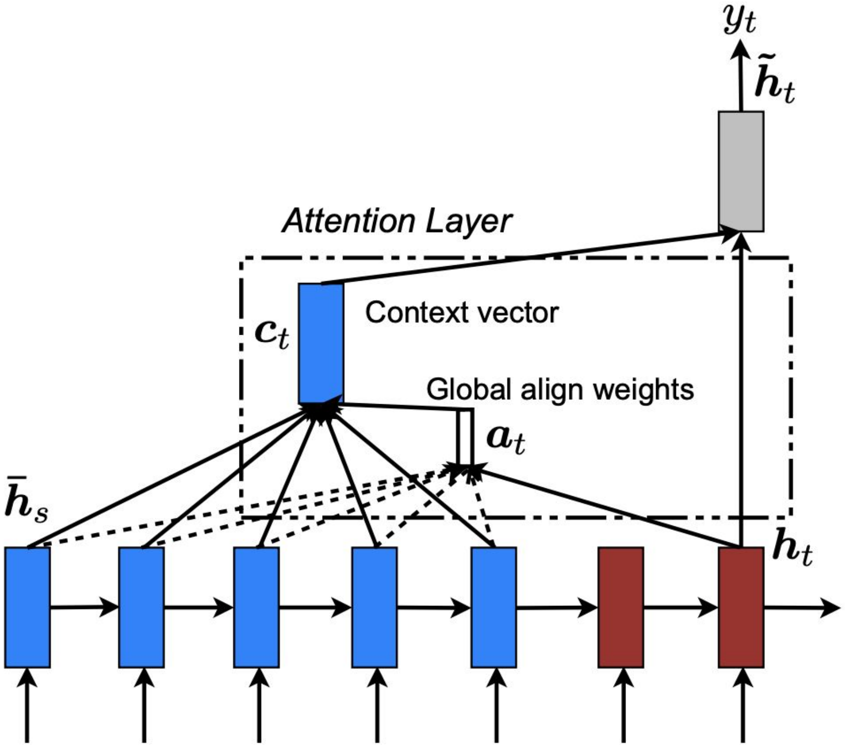


Image: Luong et al (2015)

Neural Machine Translation

$$\log p(y|x) = \sum_{j=1}^{m} \log p\left(y_j | y_{<j}, \boldsymbol{s}\right)$$

$$p\left(y_j | y_{<j}, \boldsymbol{s}\right) = \operatorname{softmax}\left(g\left(\boldsymbol{h}_j\right)\right)$$

Attention in Neural Machine Translation

$$\log p(y|x) = \sum_{j=1}^{m} \log p\left(y_j | y_{<j}, \boldsymbol{s}\right)$$

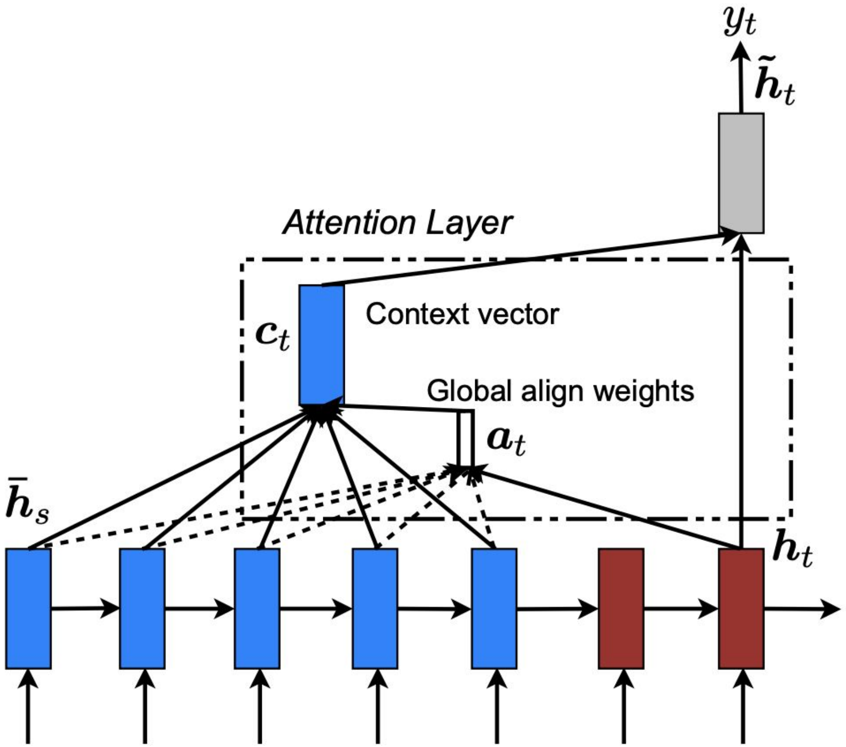
$$p\left(y_j | y_{<j}, \boldsymbol{s}\right) = \operatorname{softmax}\left(g\left(\boldsymbol{h}_j\right)\right)$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

Augment with source context

$$p(y_t | y_{<t}, x) = \operatorname{softmax}(\boldsymbol{W_s} \tilde{\boldsymbol{h}}_t)$$

Attention in Neural Machine Translation



$$p(y_t|y_{<t}, x) = \mathrm{softmax}(\boldsymbol{W_s}\tilde{\boldsymbol{h}}_t)$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

Attention in Neural Networks

$$P(\mathbf{y}|\mathbf{c}_1, ..., \mathbf{c}_T) = \prod_{t=1}^{T} P(\mathbf{y}_t|\mathbf{y}_1, ..., \mathbf{y}_{t-1}, \mathbf{c}_t)$$

$$\mathbf{c}_t = \sum_j \alpha_{tj} \mathbf{h}_j$$

weighted average of source LSTM states

$$\alpha_{tj} = \frac{\exp(\epsilon_{tj})}{\sum_j \exp(\epsilon_{tj})}$$

weights form a probability distribution

$$\epsilon_{tj} = F_{att}(\mathbf{h}_{t-1}, \mathbf{h}_j)$$

some learned function

Parameterizing Attention

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Attention in Neural Networks

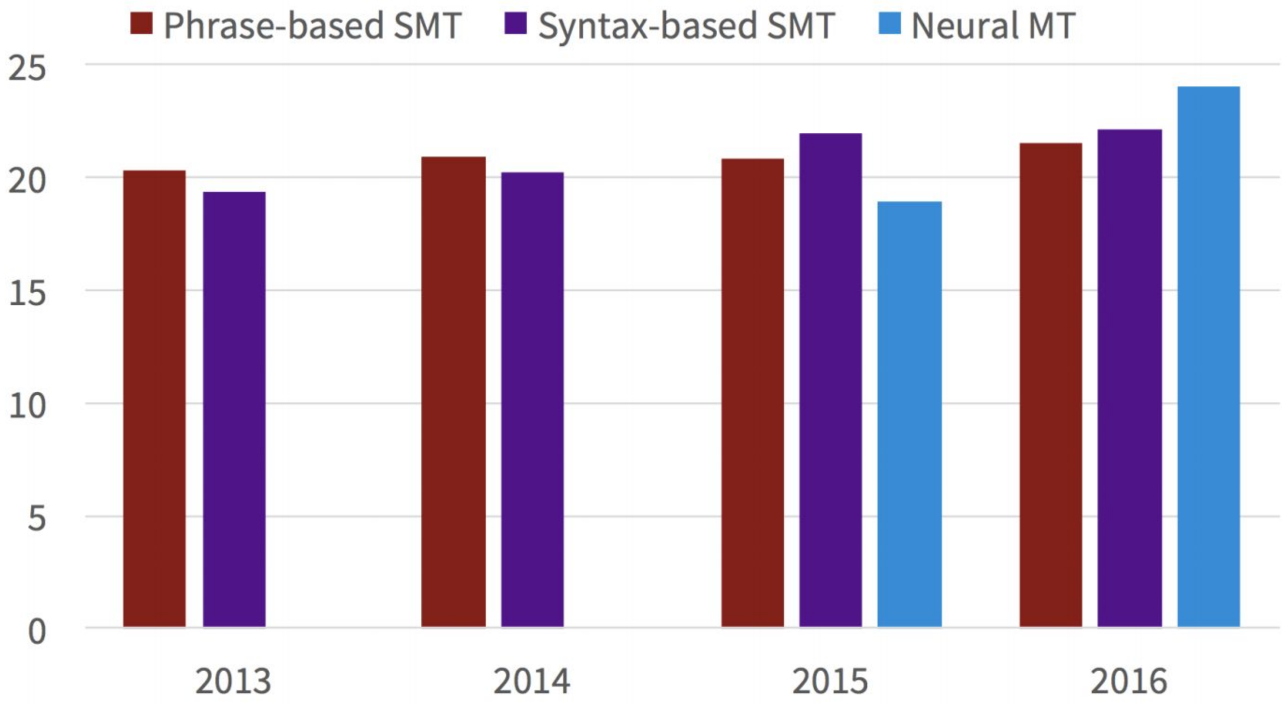
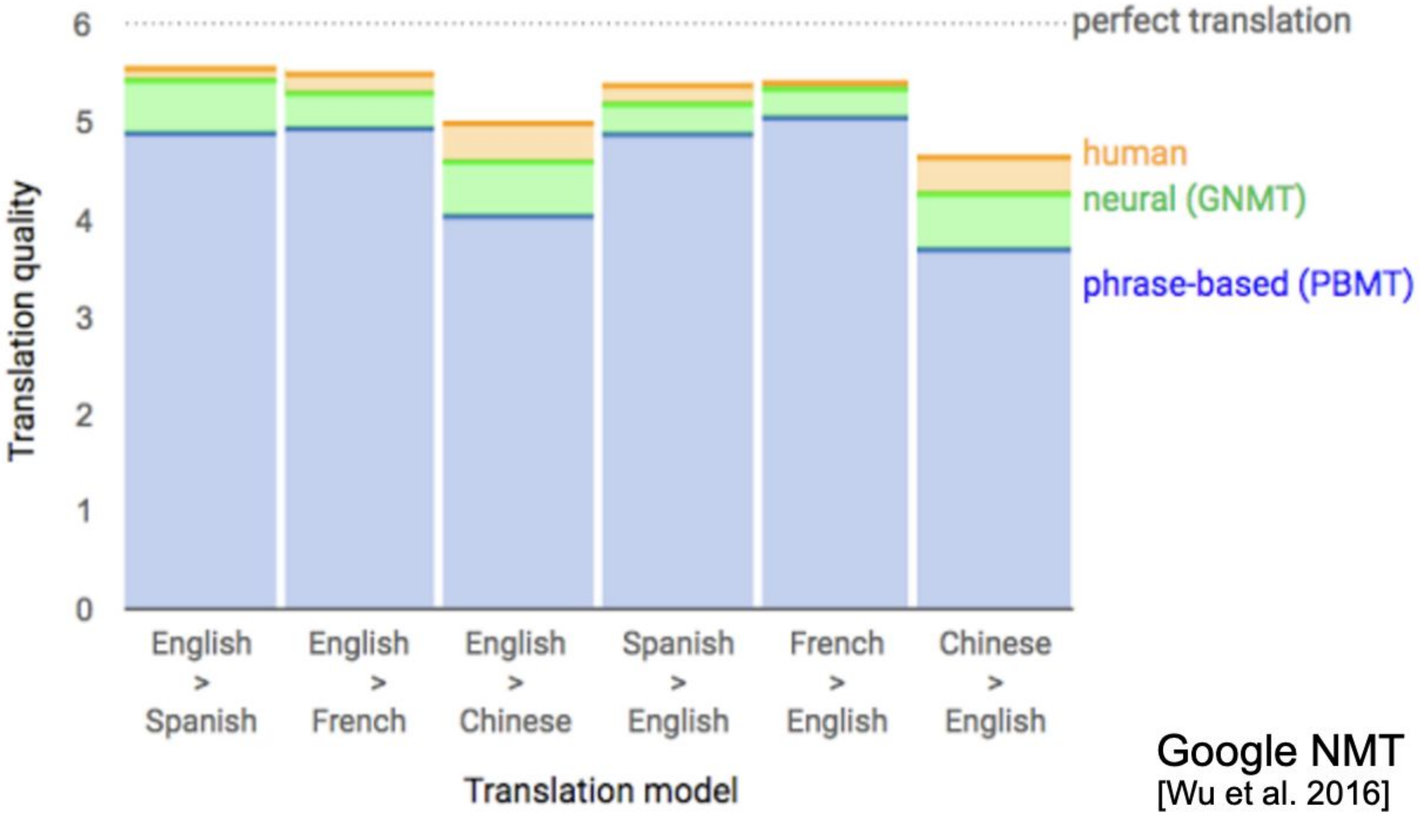


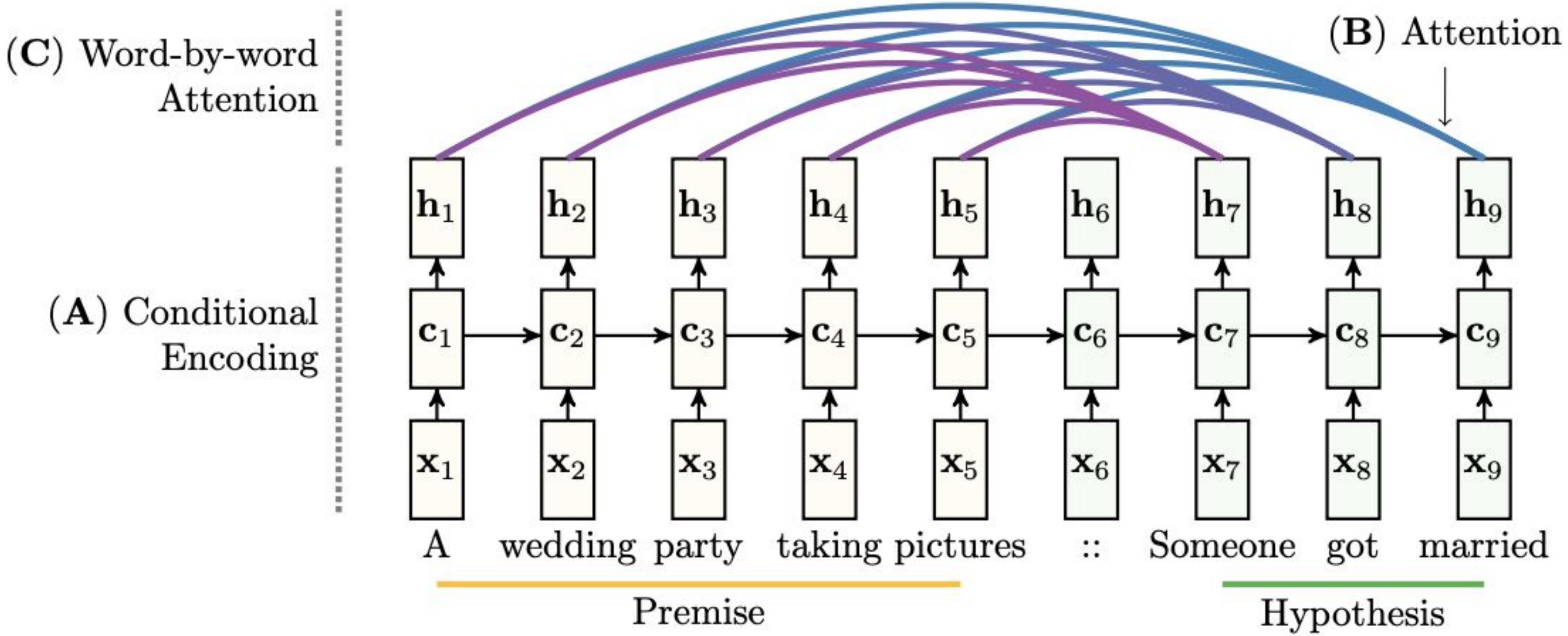
Image: Sennrich (2016)

Attention in Neural Networks



Google NMT
[Wu et al. 2016]

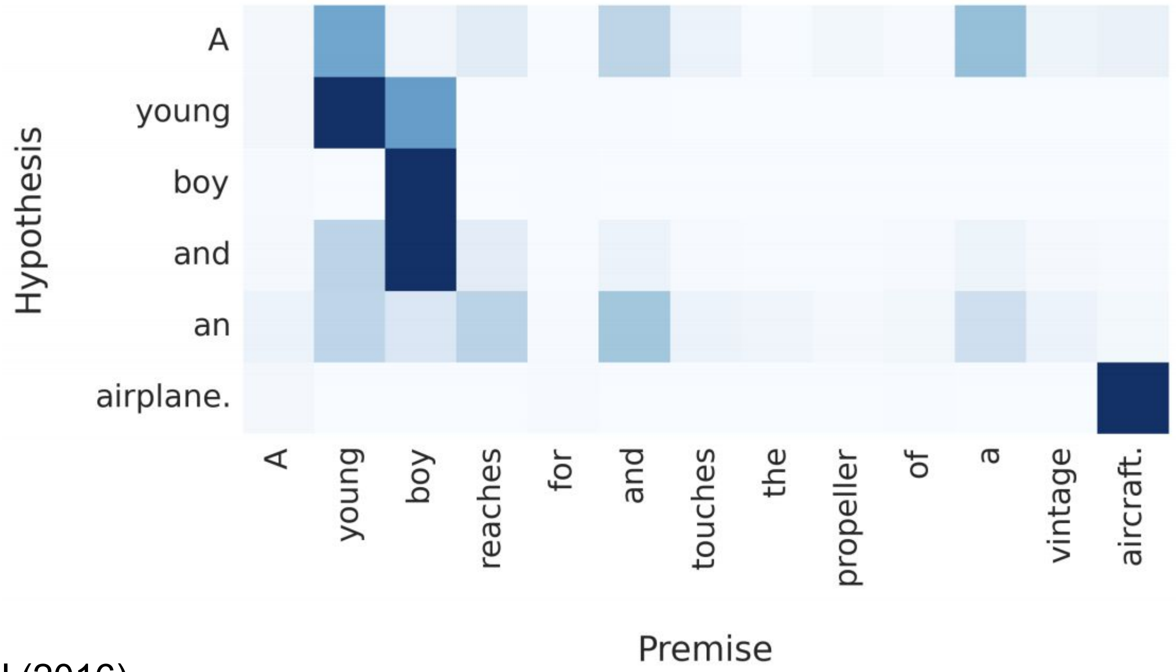
Attention in Neural Networks



Rocktaschel et al (2016)

Interpretability of Attention

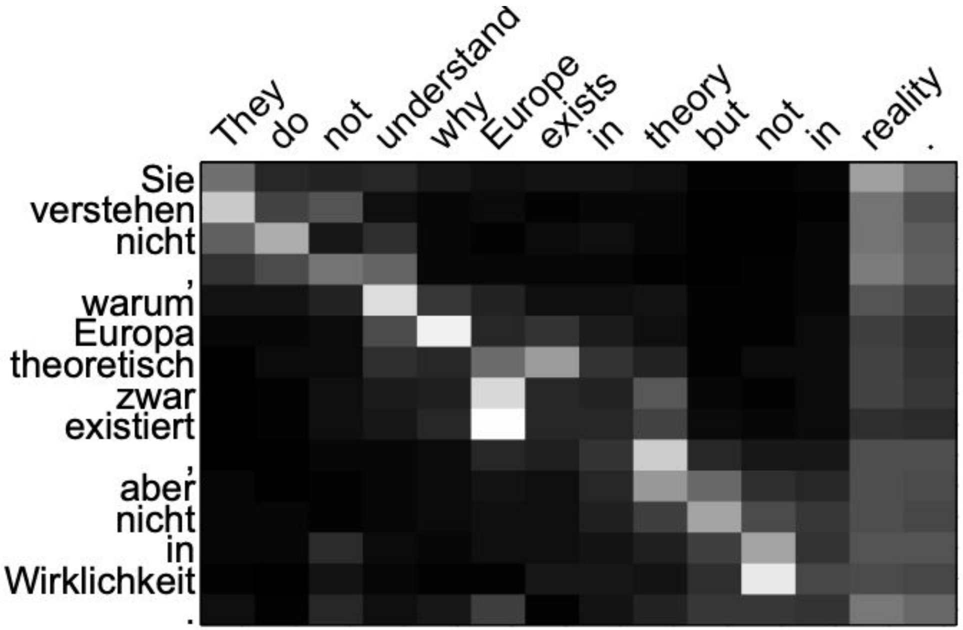
- Attention provides probabilistic weights on input sequence



Rocktaschel et al (2016)

Interpretability of Attention

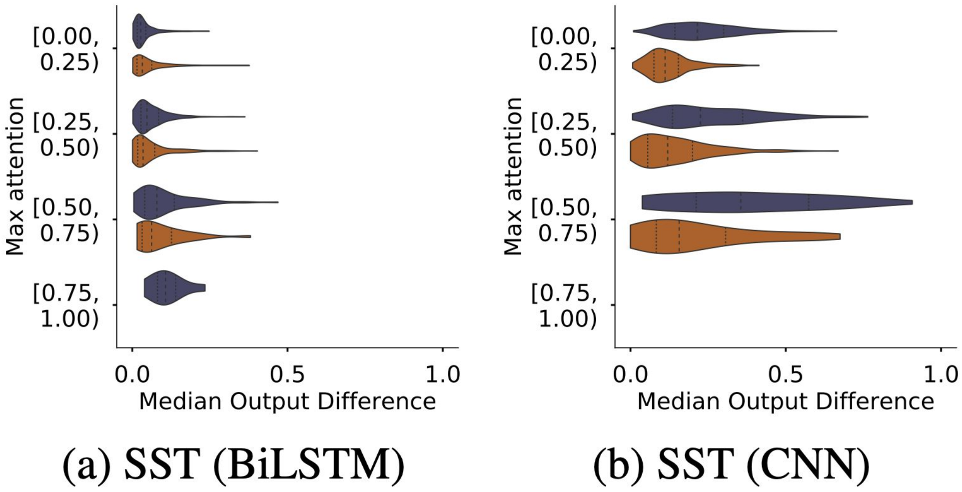
- Attention provides probabilistic weights on input sequence



Luong et al (2016)

Is Attention Interpretable?

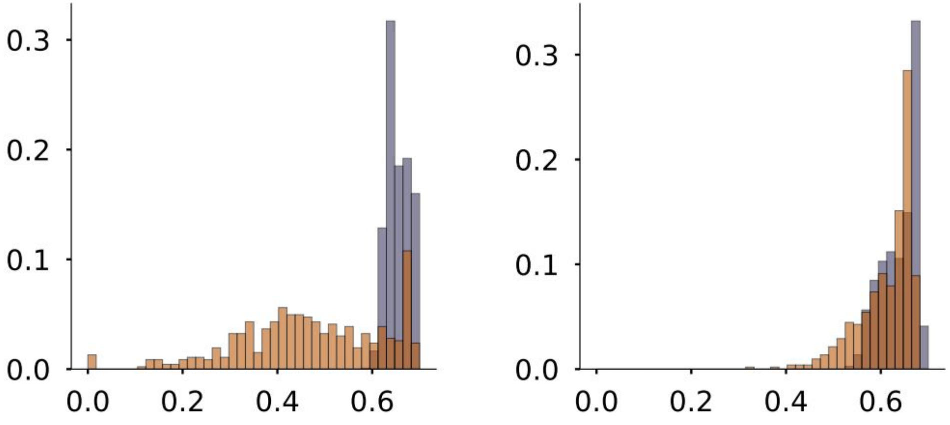
- Replace attention weights at test time by random vector of weights.
  - The results do not change significantly.



Jain and Wallace (2019)

Is Attention Interpretable?

- Find adversarial weights that are maximally divergent while producing very similar result.



(c) Diabetes (BiLSTM) (d) Diabetes (CNN)

Jain and Wallace (2019)

Is Attention Interpretable?

- Zero out the maximum attention weight (and renormalize)
- Zero out a random attention weight (and renormalize)

Remove random: Decision flip? (columns) / **Remove i^*: Decision flip?** (rows)

	Yahoo		IMDB	
	Yes	No	Yes	No
Yes	0.5	8.7	2.2	12.2
No	1.3	89.6	1.4	84.2

	Amazon		Yelp	
	Yes	No	Yes	No
Yes	2.7	7.6	1.5	8.9
No	2.7	87.1	1.9	87.7

Serrano and Smith (2019)

Is Attention Interpretable?

- Attention is interpretable when it's model critical [Ongoing work, Vashishta et al 2019]
- Train models with standard attention layer or random attention layer.

	BLEU
Luong	20.2
Bahdanau	18.2
Random	7.3

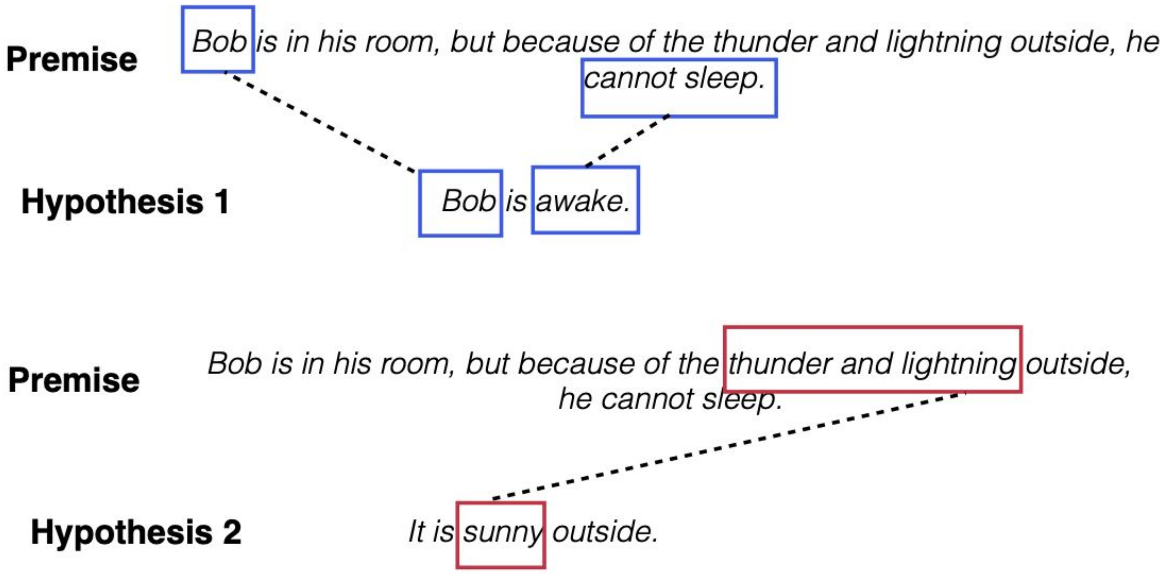
NMT

	Accuracy
Luong	Similar
Bahdanau	Similar
Random	Similar

Text classification

Attention in Neural Networks

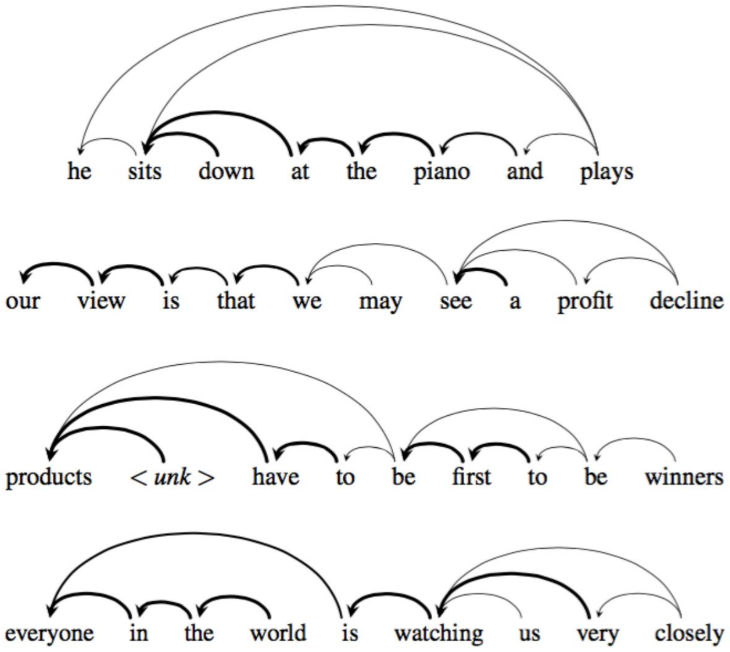
- If attention is so important, how much can we get with just attention?
- Often times, alignment is sufficient, do not need sentence representation



Self-Attention

Self-attention

- Construct a context by attending to your-self: Intra-sentence attention



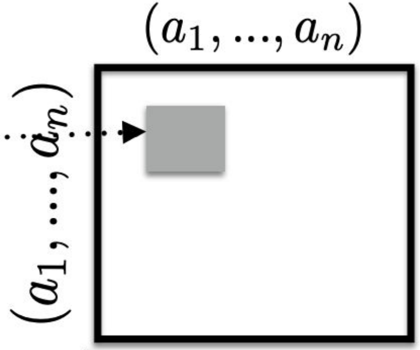
Language modeling

Cheng et al (2016)

Self-attention

- Self Attention without LSTMs
- Use weak word order information via distance bias

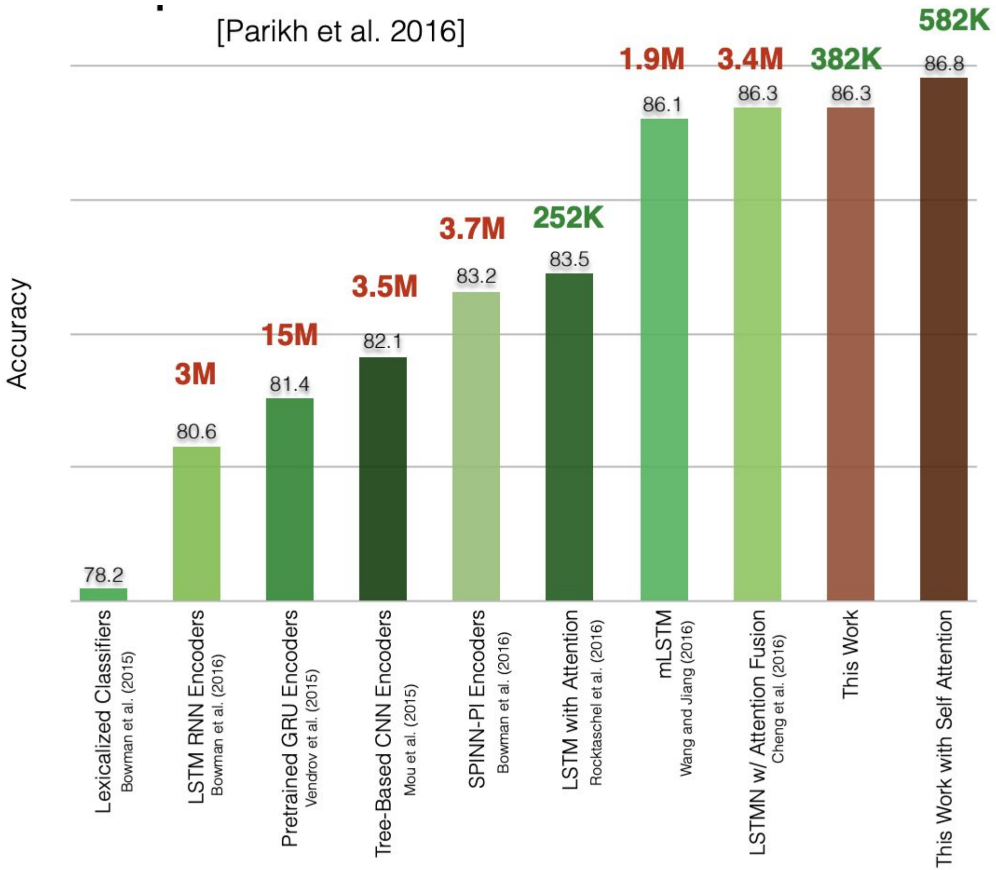
$$f_{ij} = F_{\text{intra}}(a_i)^\top F_{\text{intra}}(a_j)$$



$$a'_i = \sum_{j=1}^{n} \frac{\exp(f_{ij} + d_{i-j})}{\sum_{k=1}^{n} \exp(f_{ik} + d_{i-j})} a_j \quad \Rightarrow \quad \bar{a}_i = [a_i, a'_i]$$

Parikh et al (2016), Cheng et al (2016)

Self-attention in NLI



Performance on SNLI dataset

Transformers

Learning Representations of Variable Length Data

Basic building block of sequence-to-sequence learning

Neural machine translation, summarization, QA, …

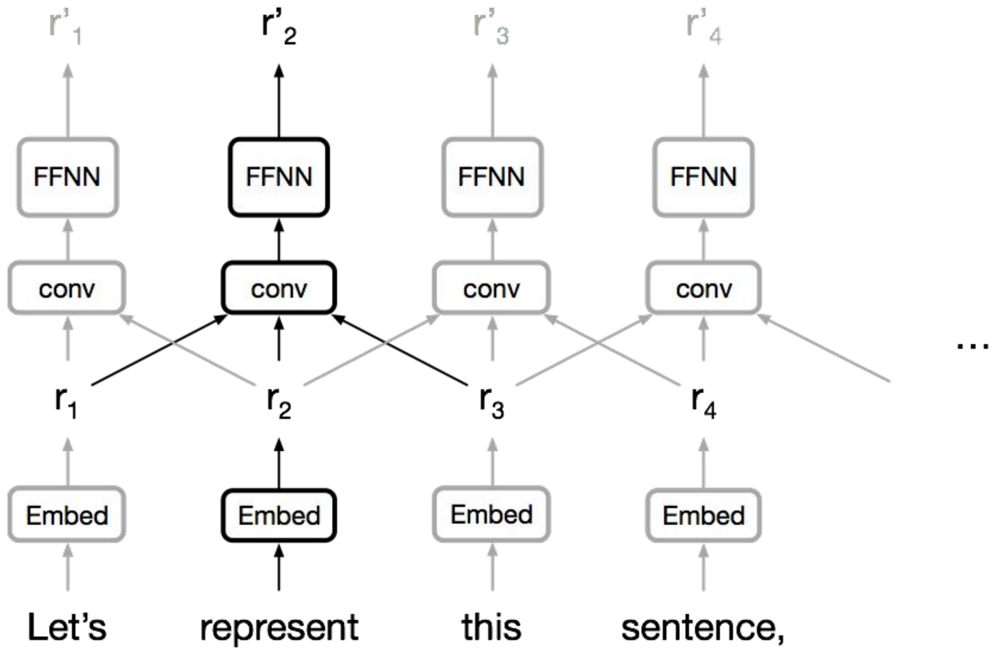
Recurrent Neural Networks

- Model of choice for learning variable-length representations.
- Natural fit for sentences and sequences of pixels.
- LSTMs, GRUs and variants dominate recurrent models.

But…

- Sequential computation inhibits parallelization.
- No explicit modeling of long and short range dependencies.
- We want to model hierarchy.

Convolutional Neural Networks?



Convolutional Neural Networks?

Trivial to parallelize (per layer).

Exploits local dependencies

‘Interaction distance’ between positions linear or logarithmic.

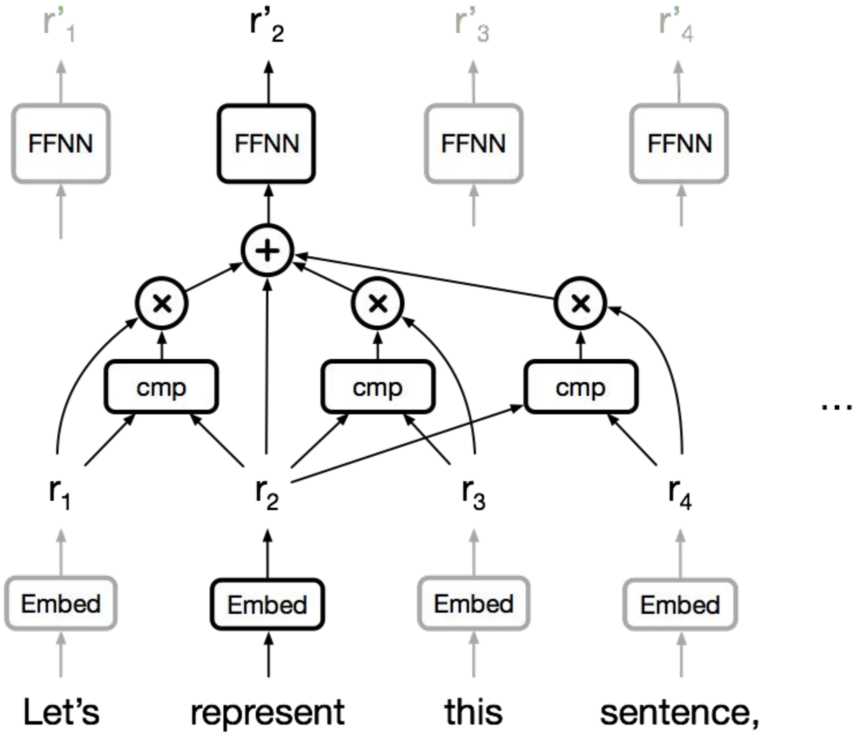
Long-distance dependencies require many layers.

Attention

Attention between encoder and decoder is crucial in NMT.

Why not use attention for representations?

Self-Attention



Self-Attention

Constant ‘path length’ between any two positions.

Gating/multiplicative interactions.

Trivial to parallelize (per layer).

Stacking Self-Attention

- Just like CNNs and LSTMs can be stacked together, so can self-attention
- Build iteratively transformed representations

$$\mathbf{a}^{(0)} = (a_1, ..., a_n)$$

$$\Downarrow$$

$$\mathbf{a}^{(1)} = (a_1^{(1)}, ..., a_n^{(1)})$$

$$\Downarrow$$

$$.......$$

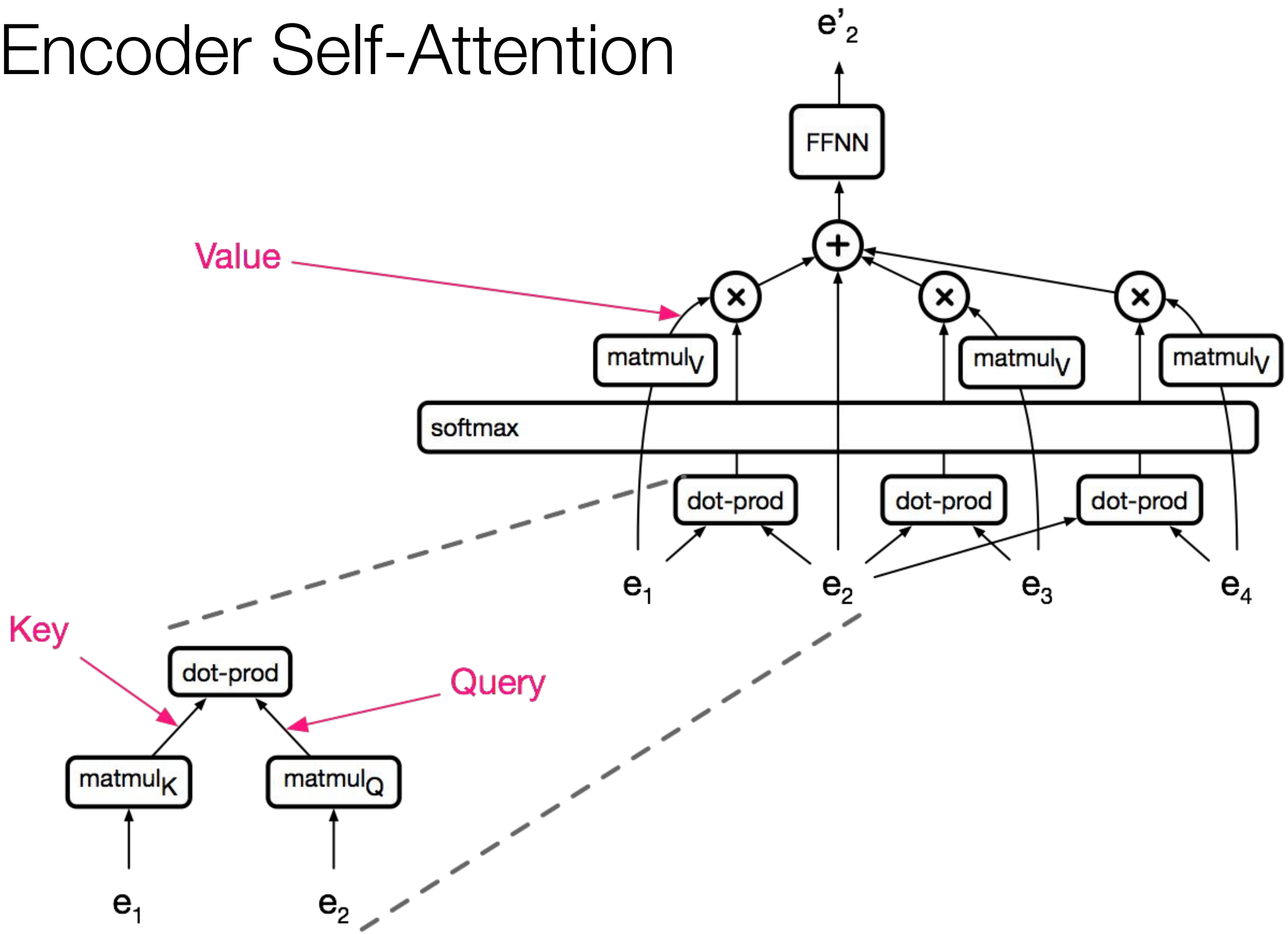
$$\Downarrow$$

layer ℓ

$$\mathbf{a}^{(\ell)} = (a_1^{(\ell)}, ..., a_n^{(\ell)})$$

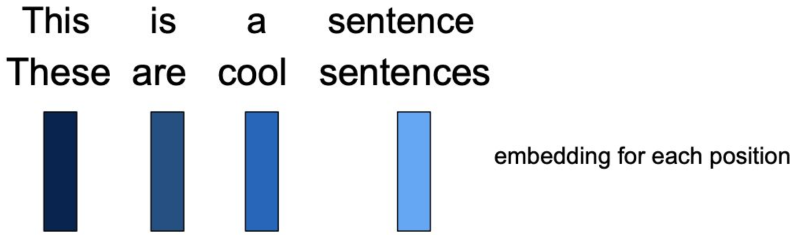
$$\Downarrow$$

Encoder Self-Attention

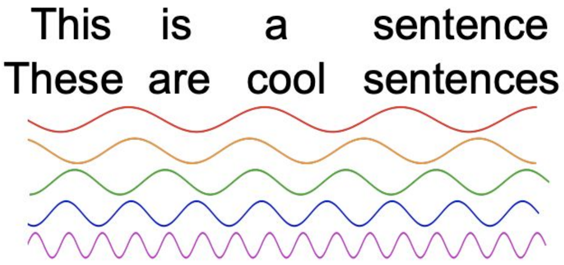


Position Embeddings

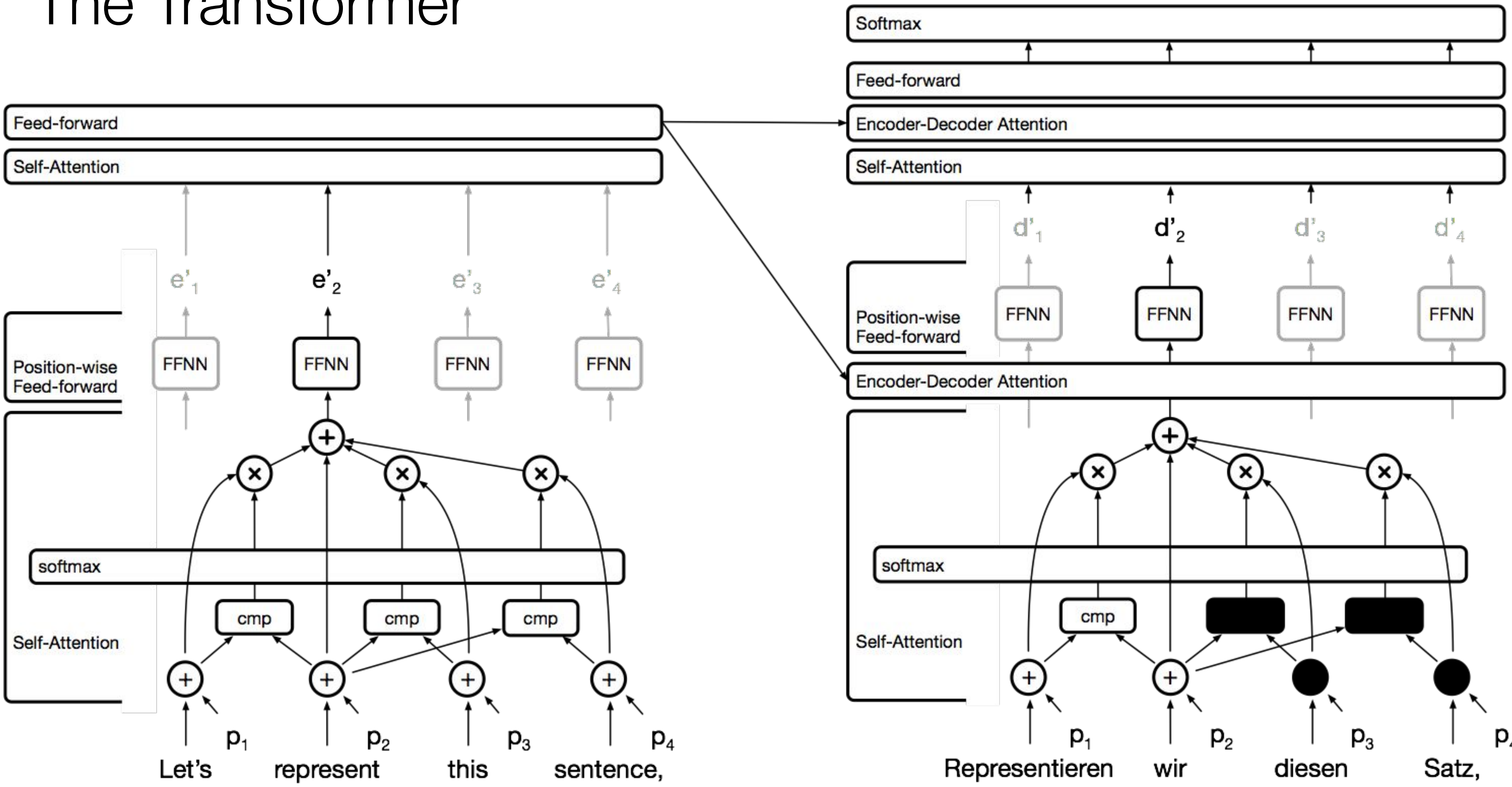
- Position encoding. A weakness of (vanilla) self attention is that unlike CNNs/LSTMs, it has no notion of position.
- Option 1: Learned position embeddings [Gehring et al. 2017]



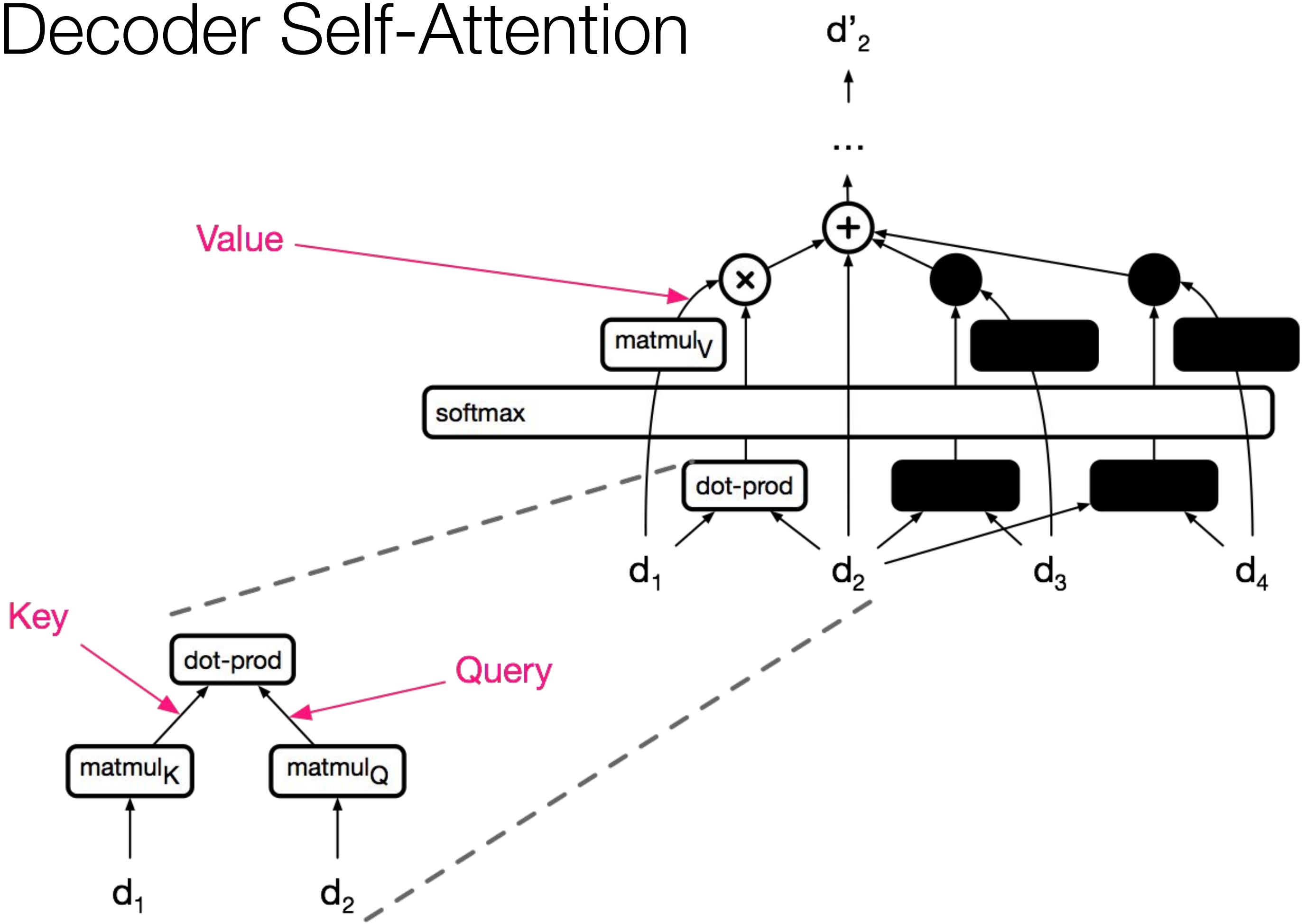
- Option 2: Fixed sinusoids of various frequencies [Vaswani et al 2017]



The Transformer



Decoder Self-Attention



Attention is Cheap!

Self-Attention	$O(\text{length}^2 \cdot \text{dim})$
RNN (LSTM)	$O(\text{length} \cdot \text{dim}^2)$
Convolution	$O(\text{length} \cdot \text{dim}^2 \cdot \text{kernel\_width})$

Attention is Cheap!

Self-Attention	$O(\text{length}^2 \cdot \text{dim})$	$= 4{\cdot}10^9$
RNN (LSTM)	$O(\text{length} \cdot \text{dim}^2)$	$= 16{\cdot}10^9$
Convolution	$O(\text{length} \cdot \text{dim}^2 \cdot \text{kernel\_width})$	$= 6{\cdot}10^9$

length=1000   dim=1000   kernel_width=3

Transformers

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

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

(**B**idirectional **E**ncoder **R**epresentations from **T**ransformers)

Pre-training in NLP

- Word embeddings are the basis of deep learning for NLP

king

[-0.5, -0.9, 1.4, …]

queen

[-0.6, -0.8, -0.2, …]

- Word embeddings (`word2vec`, `GloVe`) are often *pre-trained* on text corpus from co-occurrence statistics

Inner Product

the king wore a crown

Inner Product

the queen wore a crown

Contextual Representations

- **Problem**: Word embeddings are applied in a context free manner

```
open a bank account                 on the river bank

              [0.3, 0.2, -0.8, …]
```

- **Solution**: Train *contextual* representations on text corpus

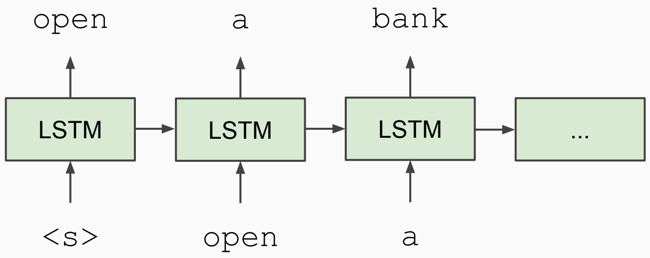
```
[0.9, -0.2, 1.6, …]                 [-1.9, -0.4, 0.1, …]

open a bank account                 on the river bank
```

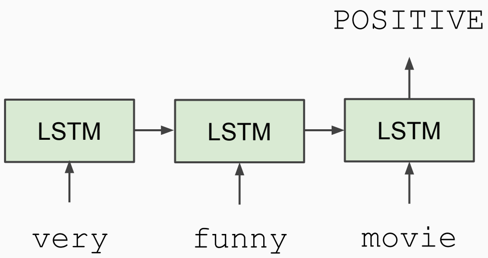
History of Contextual Representations

- *Semi-Supervised Sequence Learning*, Google, 2015

Train LSTM Language Model



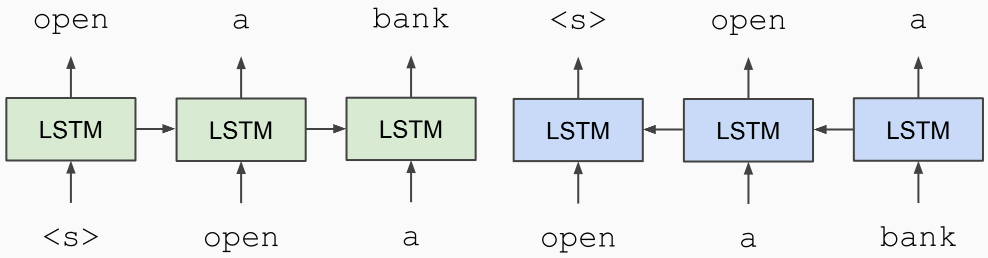
Fine-tune on Classification Task



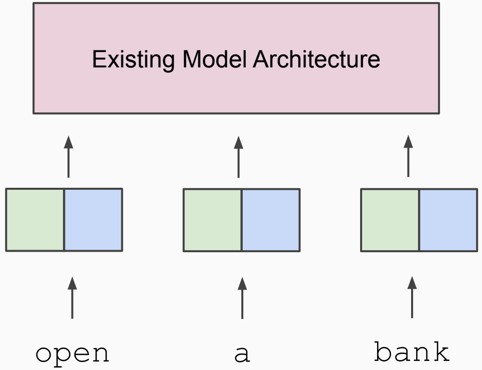
History of Contextual Representations

- *ELMo: Deep Contextual Word Embeddings*, AI2 & University of Washington, 2017

Train Separate Left-to-Right and Right-to-Left LMs



Apply as "Pre-trained Embeddings"

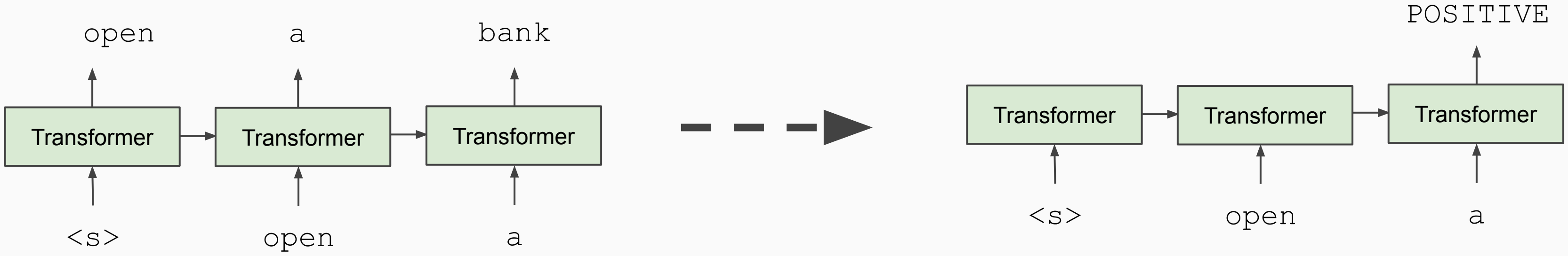


History of Contextual Representations

- *Improving Language Understanding by Generative Pre-Training*, OpenAI, 2018

Train Deep (12-layer) Transformer LM

Fine-tune on Classification Task

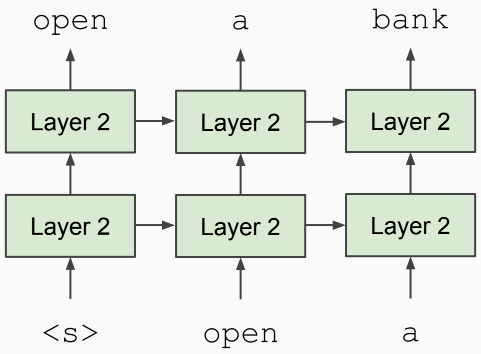


Problem with Previous Methods

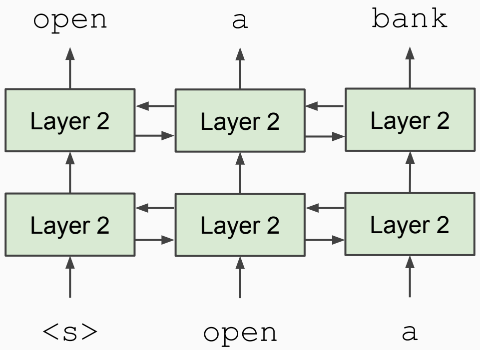
- **Problem**: Language models only use left context *or* right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
- Reason 1: Directionality is needed to generate a well-formed probability distribution.
    - We don't care about this.
- Reason 2: Words can "see themselves" in a bidirectional encoder.

Unidirectional vs. Bidirectional Models

Unidirectional context
Build representation incrementally



Bidirectional context
Words can “see themselves”



Masked LM

- **Solution:** Mask out *k*% of the input words, and then predict the masked words
  - Use $k = 15\%$

store gallon

the man went to the [MASK] to buy a [MASK] of milk

- Too little masking: Too expensive to train
- Too much masking: Not enough context

Masked LM

- Problem: Mask token never seen at fine-tuning
- Solution: 15% of the words to predict, but don't replace with `[MASK]` 100% of the time. Instead:
- 80% of the time, replace with `[MASK]`

  `went to the store → went to the [MASK]`
- 10% of the time, replace random word

  `went to the store → went to the running`
- 10% of the time, keep same

  `went to the store → went to the store`

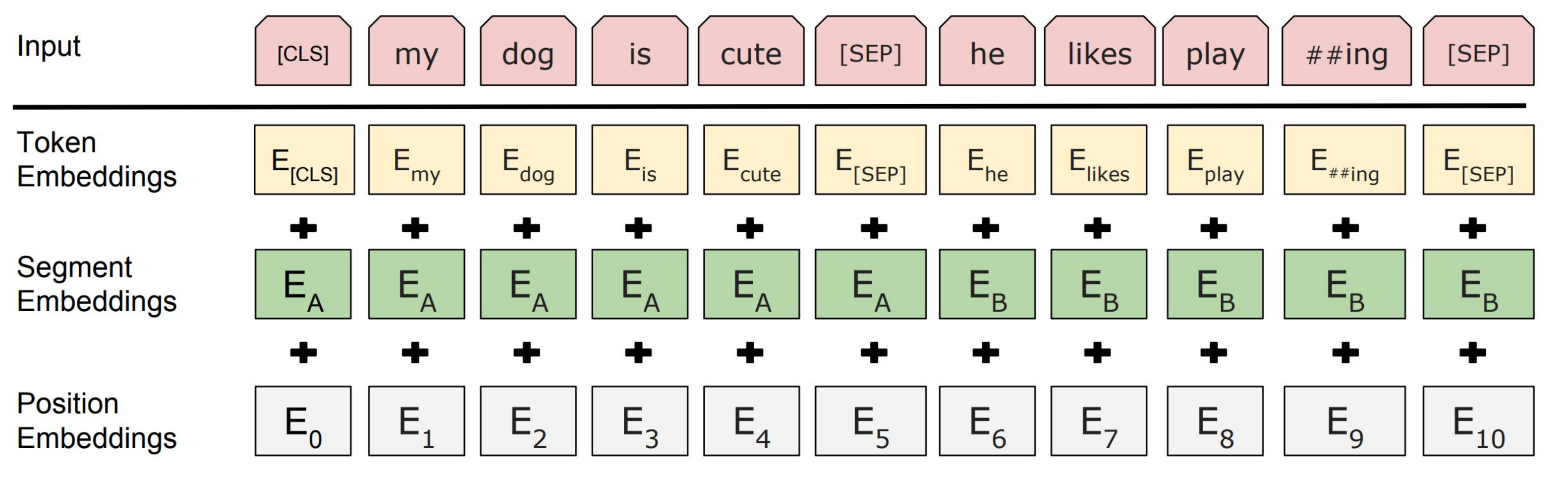
Next Sentence Prediction

- To learn *relationships* between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random 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`

Input Representation

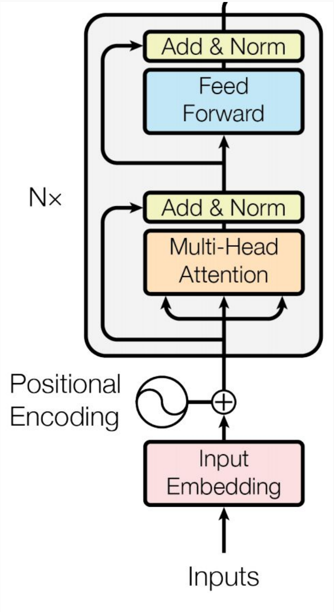


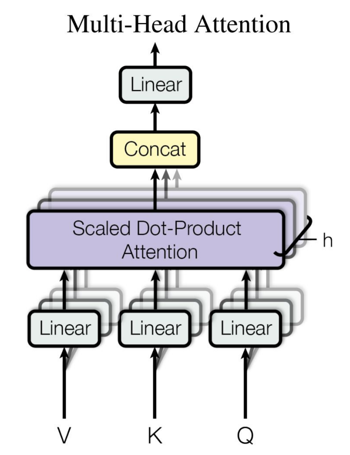
- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings

Model Architecture

Transformer encoder

- Multi-headed self attention
  - Models context
- Feed-forward layers
  - Computes non-linear hierarchical features
- Positional embeddings
  - Allows model to learn relative positioning



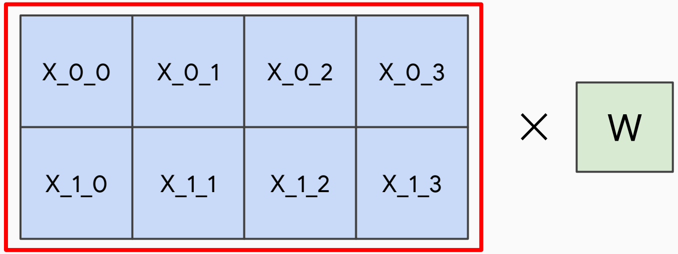


Model Architecture

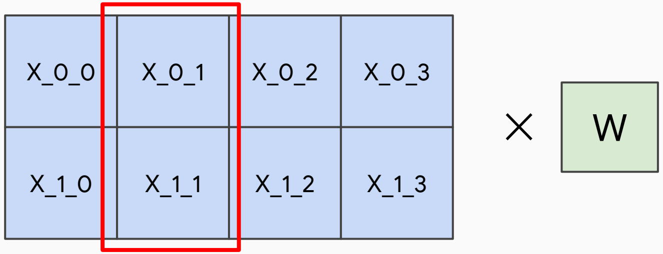
- Empirical advantages of Transformer vs. LSTM:

1. Self-attention == no locality bias
    - Long-distance context has “equal opportunity”
2. Single multiplication per layer == efficiency on TPU
    - Effective batch size is number of *words*, not *sequences*

Transformer



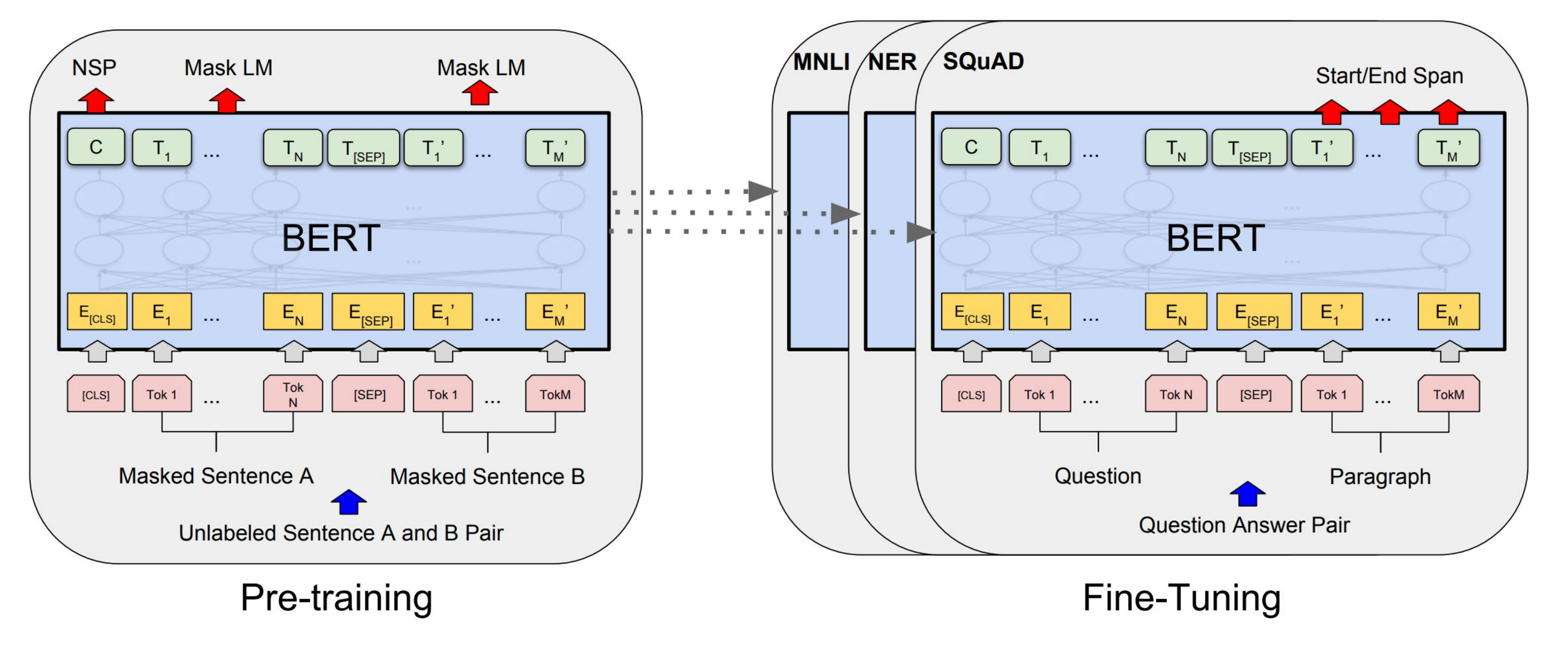
LSTM



Model Details

- Data: Wikipedia (2.5B words) + BookCorpus (800M words)
- Batch Size: 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- Training Time: 1M steps (~40 epochs)
- Optimizer: AdamW, 1e-4 learning rate, linear decay
- `BERT-Base`: 12-layer, 768-hidden, 12-head
- `BERT-Large`: 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPU slice for 4 days

Fine-Tuning Procedure



GLUE 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**

MultiNLI

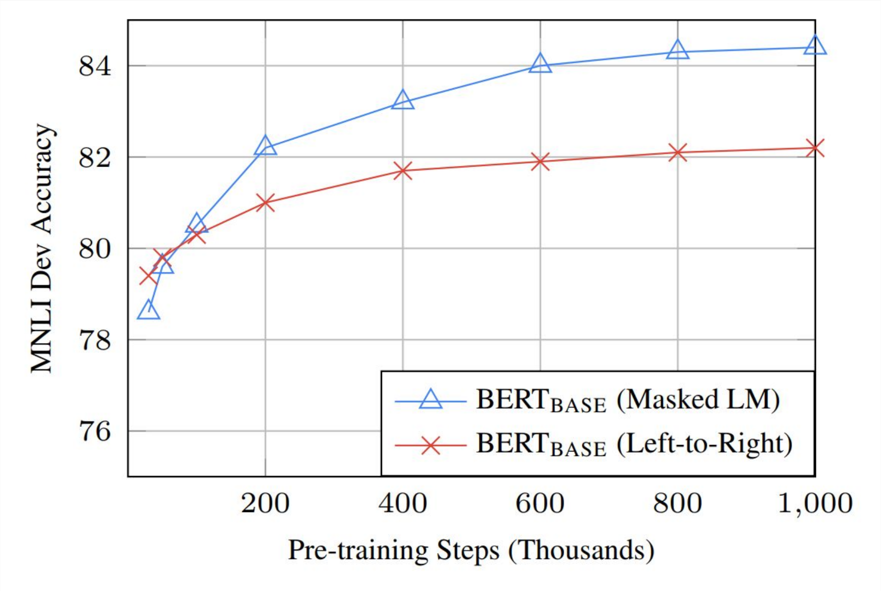
Premise: Hills and mountains are especially sanctified in Jainism.
Hypothesis: Jainism hates nature.
Label: Contradiction

CoLa

Sentence: The wagon rumbled down the road.
Label: Acceptable

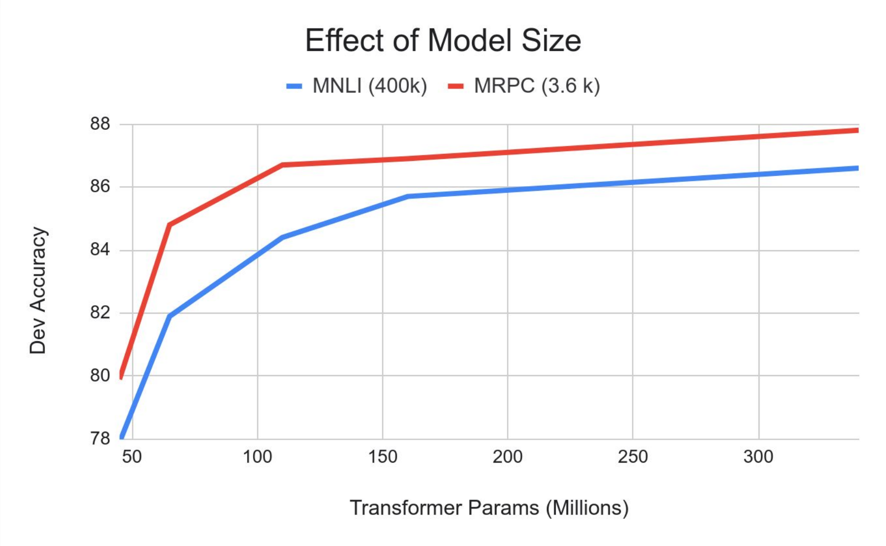
Sentence: The car honked down the road.
Label: Unacceptable

Effect of Directionality and Training Time



- Masked LM takes slightly longer to converge because we only predict 15% instead of 100%
- But absolute results are much better almost immediately

Effect of Model Size



- Big models help *a lot*
- Going from 110M -> 340M params helps even on datasets with 3,600 labeled examples
- Improvements have *not* asymptoted

Attention

↓

Self-Attention

↓

Transformers

↓

BERT

Thank you!