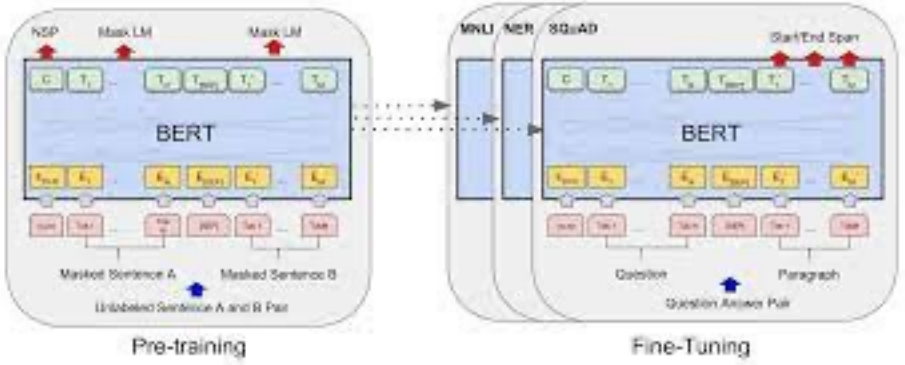
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

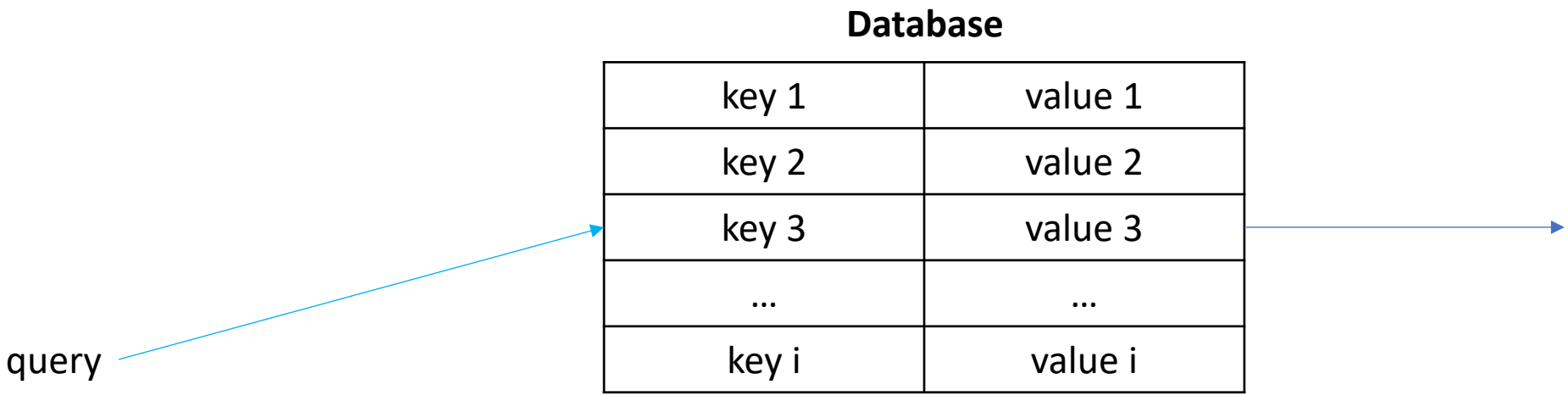
4 June, 2021

BERT



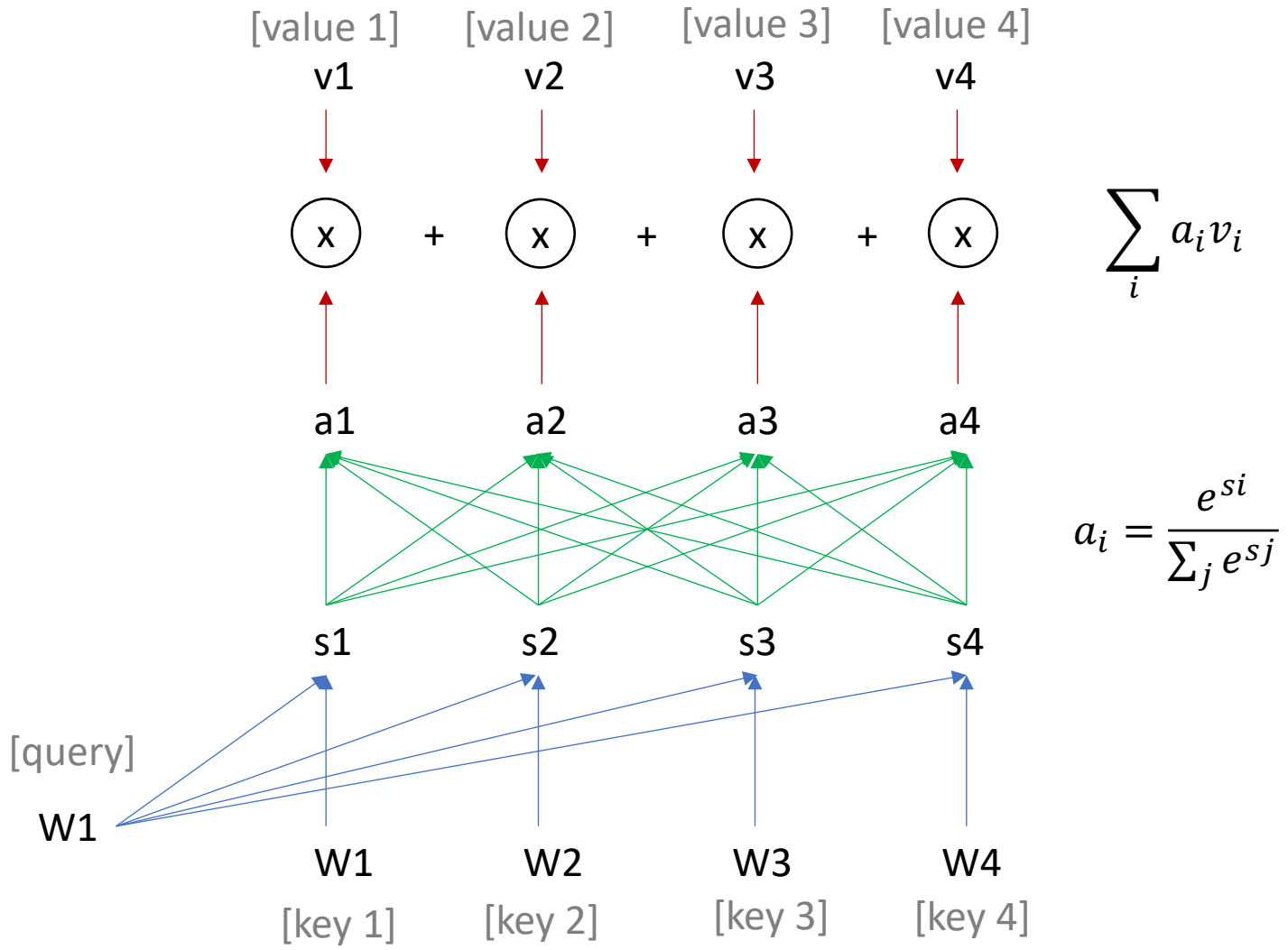
- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
  - Model architecture: Transformers
  - Tasks in training: Masked LM, nest sentence prediction
  - Data: BooksCorpus (800M words) + English Wikipedia (2,500M words)
- Pre-trained language model
  - A model to predict next word or recover the missing word in the sequence
  - Feature-based: use task-specific architectures that include the pre-trained representations as additional features
  - Fine-tuning: is trained on the downstream tasks by simply fine-tuning all pretrained parameters

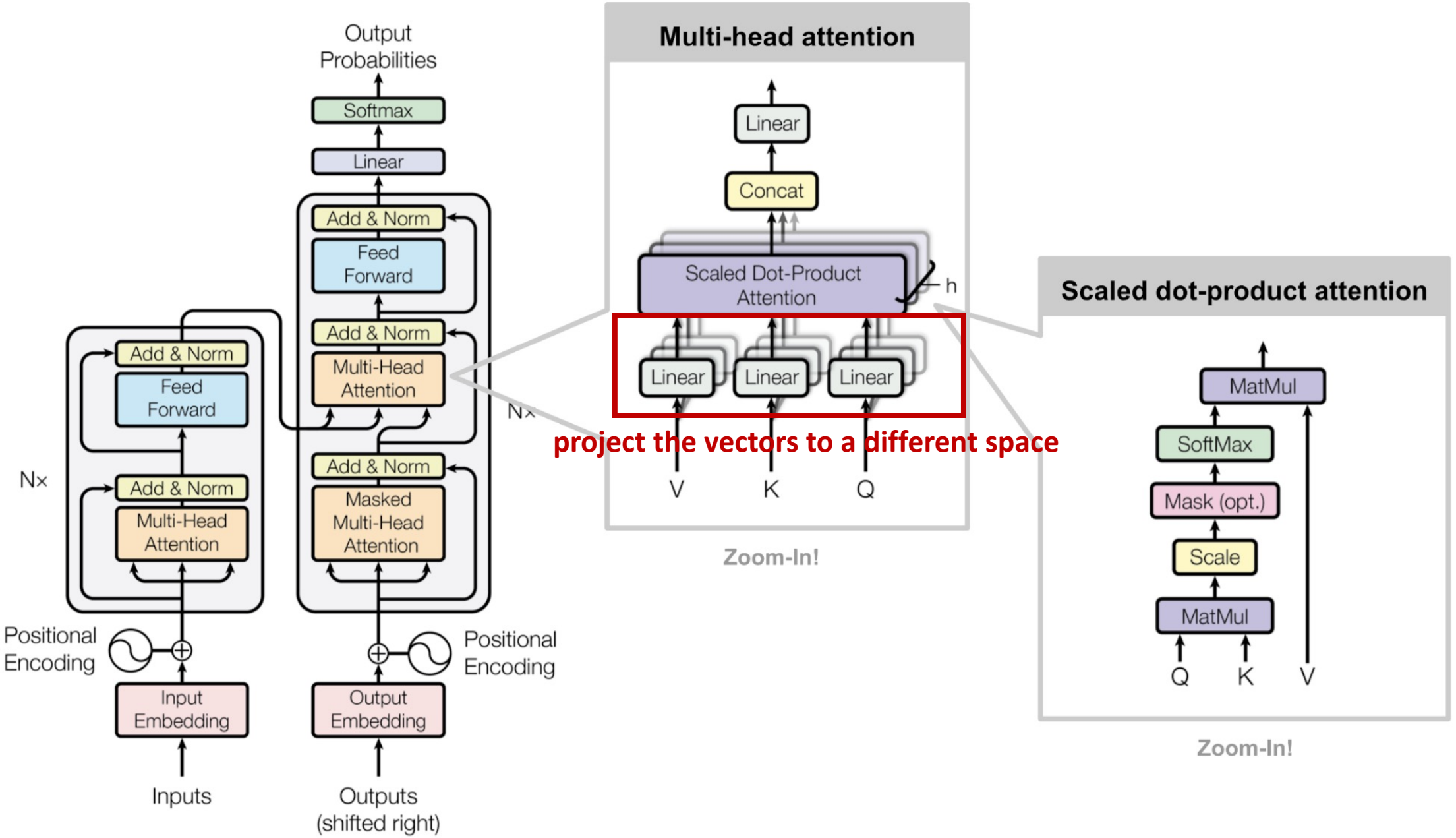
Transformers: Attention is all you need



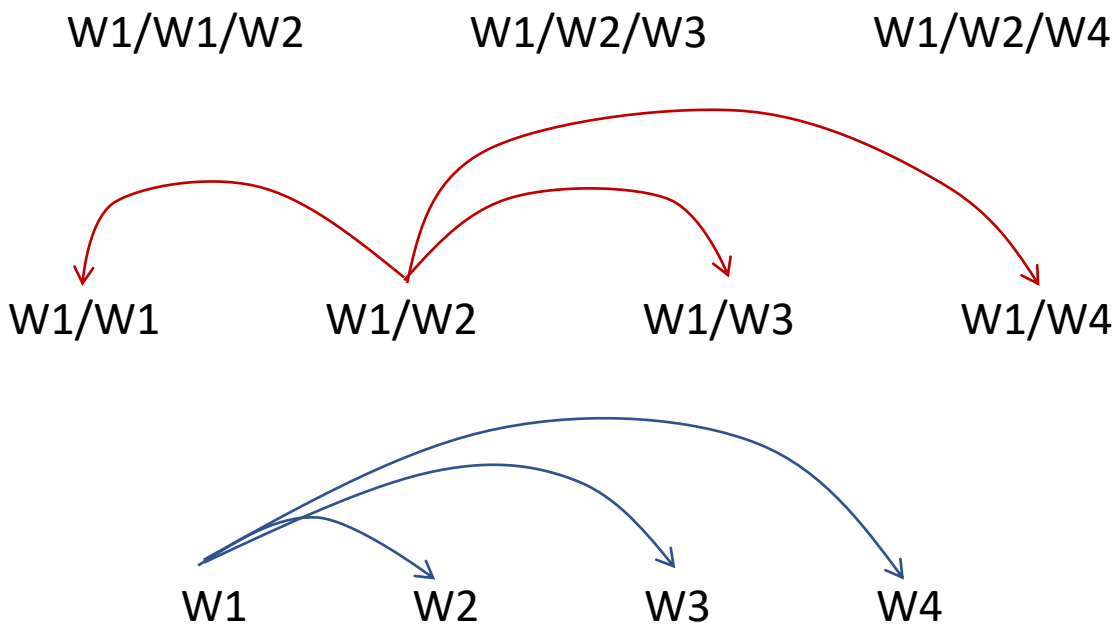
$$\text{attention}(q, \boldsymbol{k}, \boldsymbol{v}) = \sum_i \text{similarity}(q, k_i) \times v_i$$

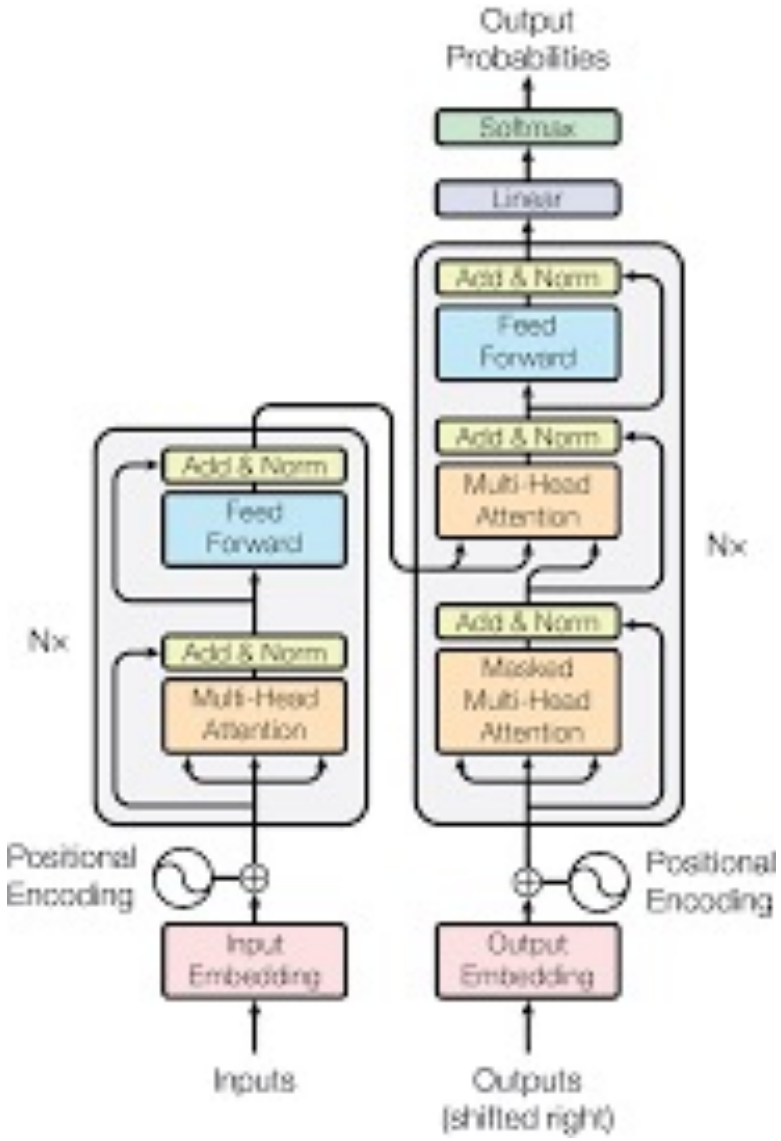
Attention mechanism in Transformers





https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html


W1/W1/W2
W1/W2/W3
W1/W2/W4
W1/W1
W1/W2
W1/W3
W1/W4
W1
W2
W3
W4


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

RNNs vs. Transformers

Challenges with RNN	Transformer networks
Long range dependencies	Facilitate long range dependencies
Gradient vanishing and explosion	No gradient vanishing and explosion
Large number of training steps	Fewer training steps
Recurrence prevents parallel computation	No recurrence that facilitate parallel computation

https://cs.uwaterloo.ca/~ppoupart/teaching/cs480-fall20/slides/cs480-lecture19.pdf

Pre-training BERT

Task #1: Masked LM

- Cloze task
- Bidirectional
  - 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
  - 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
  - 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

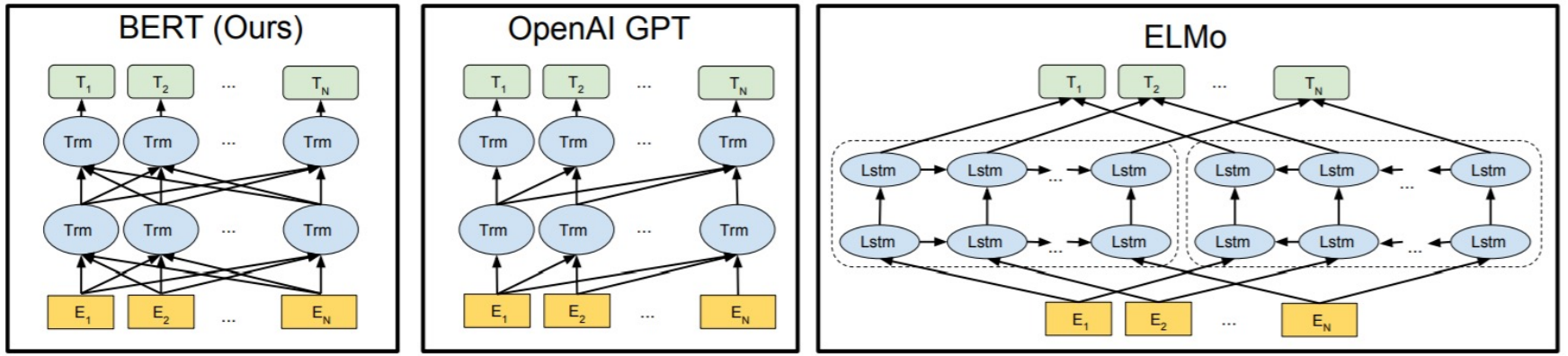
Task #2: Next Sentence Prediction

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

Comparison with other LMs

- GPT, GPT-2
  - Decoder transformer predicting next word based on previous words: $P(x_t|x_1 \dots x_{t-1})$
- BERT
  - Transformer predicting a missing word based on surrounding words: $P(x_t|x_1 \dots x_{t-1}, x_{t+1} \dots x_T)$



Performance

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	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

But was beaten by XLNet later…