BERT: Contextual Representations for Better Text Understanding

CS 293S UCSB Tao Yang 2022

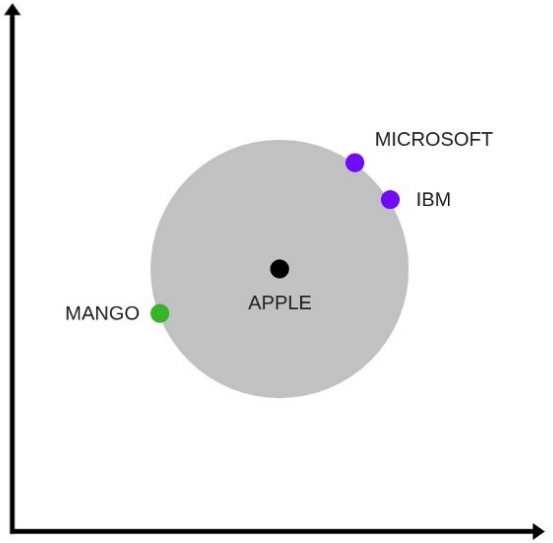
Outline

- Why BERT
- BERT Architecture
  - Self-attention computation
  - How to pre-train
- BERT Applications
  - Classification, question answering, ad-hoc search

Word Embeddings and Contextual Representations

Word embeddings are the basis of deep learning for text understanding and NLP

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



Problem: Word embeddings are applied in a context free manner

The taste of this apple is good → `[0.9, -.2, .6, ...]`

This apple phone looks good

Solution: Train *contextual* representations on text corpus

`[0.1, -.2, .6, ...]` `[0.3, .1, .3, ...]`

BERT: Bidirectional Encoder Representations for Transformers

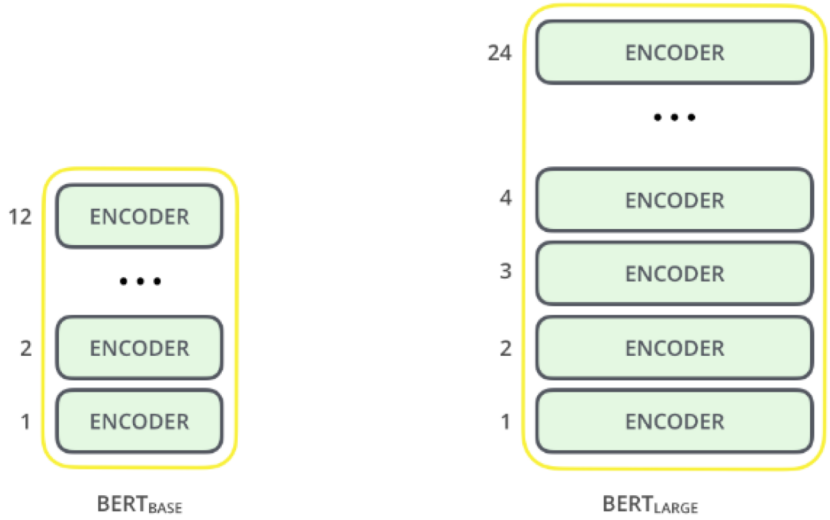
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova Google, Proc of NAACL 2019

BERT architecture is a multi-layer bidirectional Transformer encoder.

- BERT base – 12 layers (transformer blocks), 768-hidden, 12 attention heads, and 110 million parameters.
- BERT Large – 24 layers, 1024-hidden, 16 attention heads and, 340 million parameters.

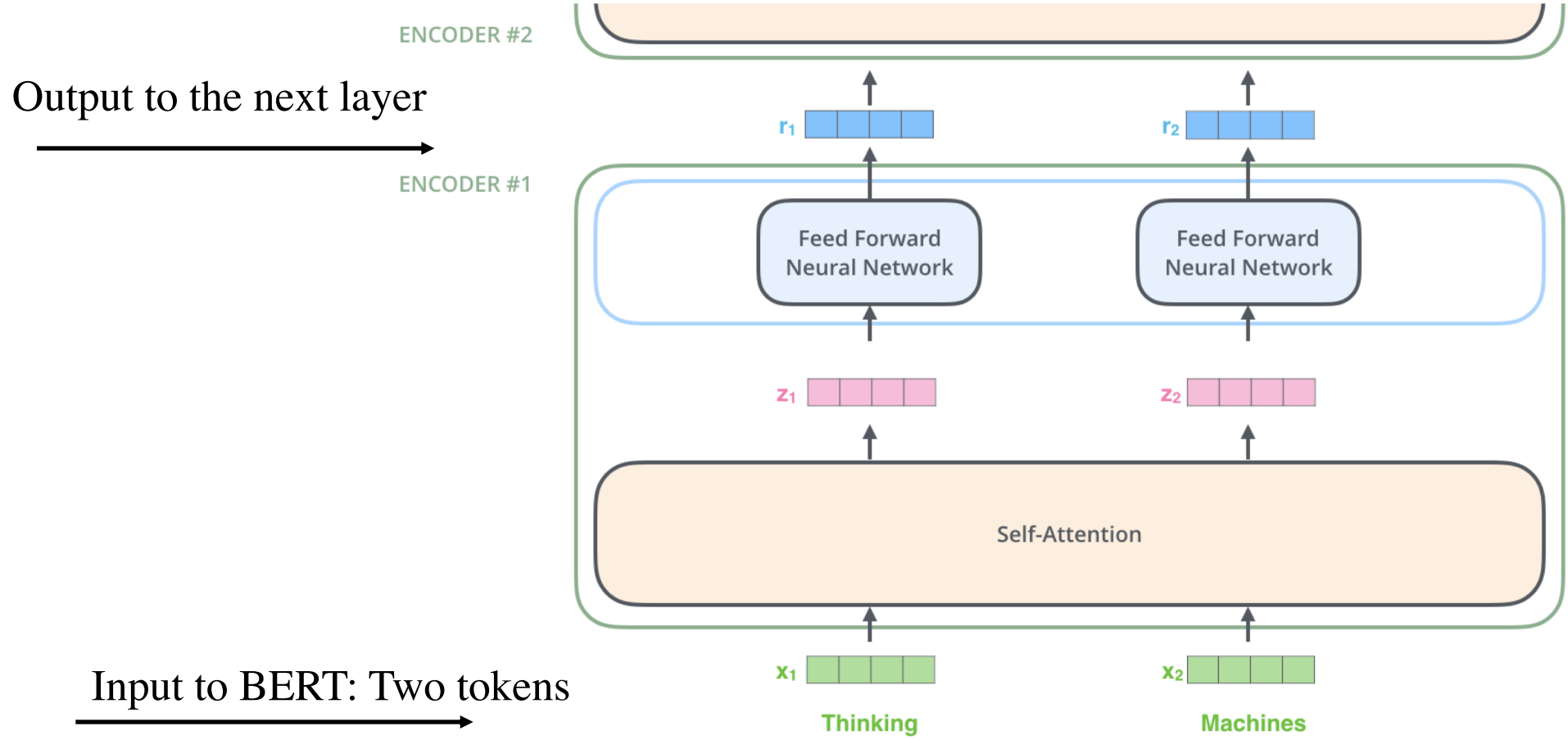
BERT computation is expensive

Takes many days to train



The transformer layers of BERT

- Also called encoder



https://jalammar.github.io/illustrated-transformer/

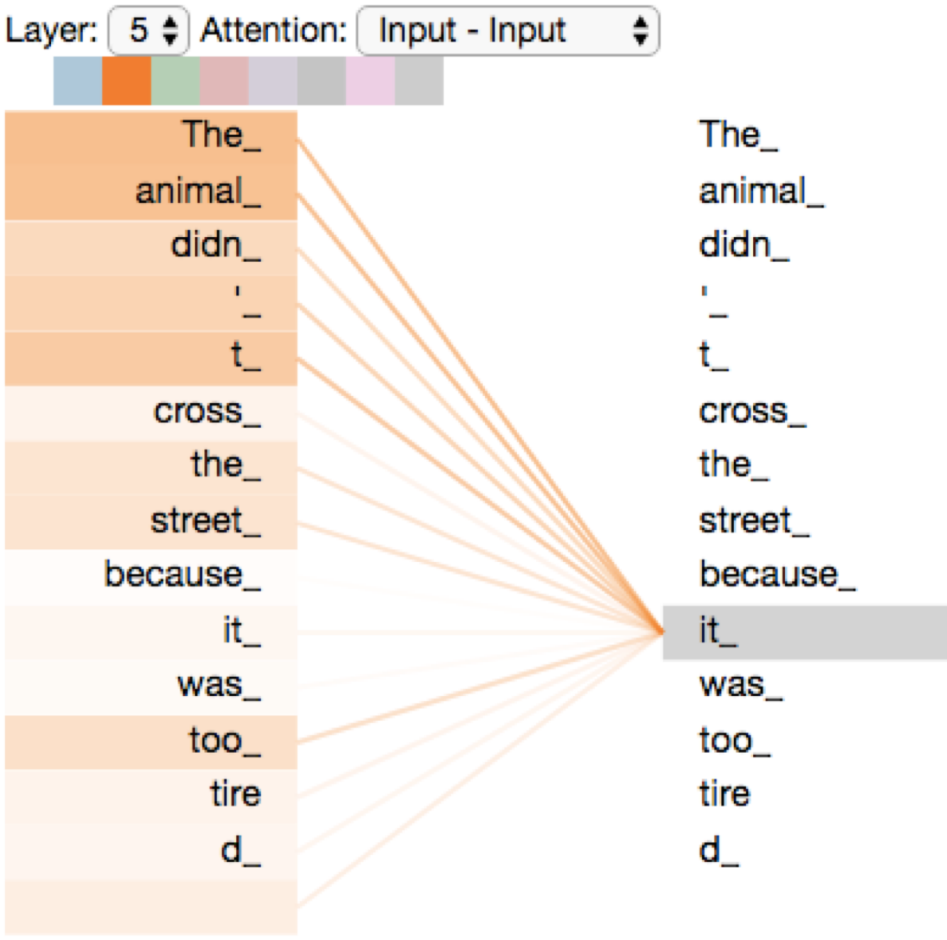
More about Attention

"The Attention is All You Need" paper, A. Vaswani et al. NIPS 2017

Self-attention is the method the Transformer uses to detect how other words relevant to the current word being processed

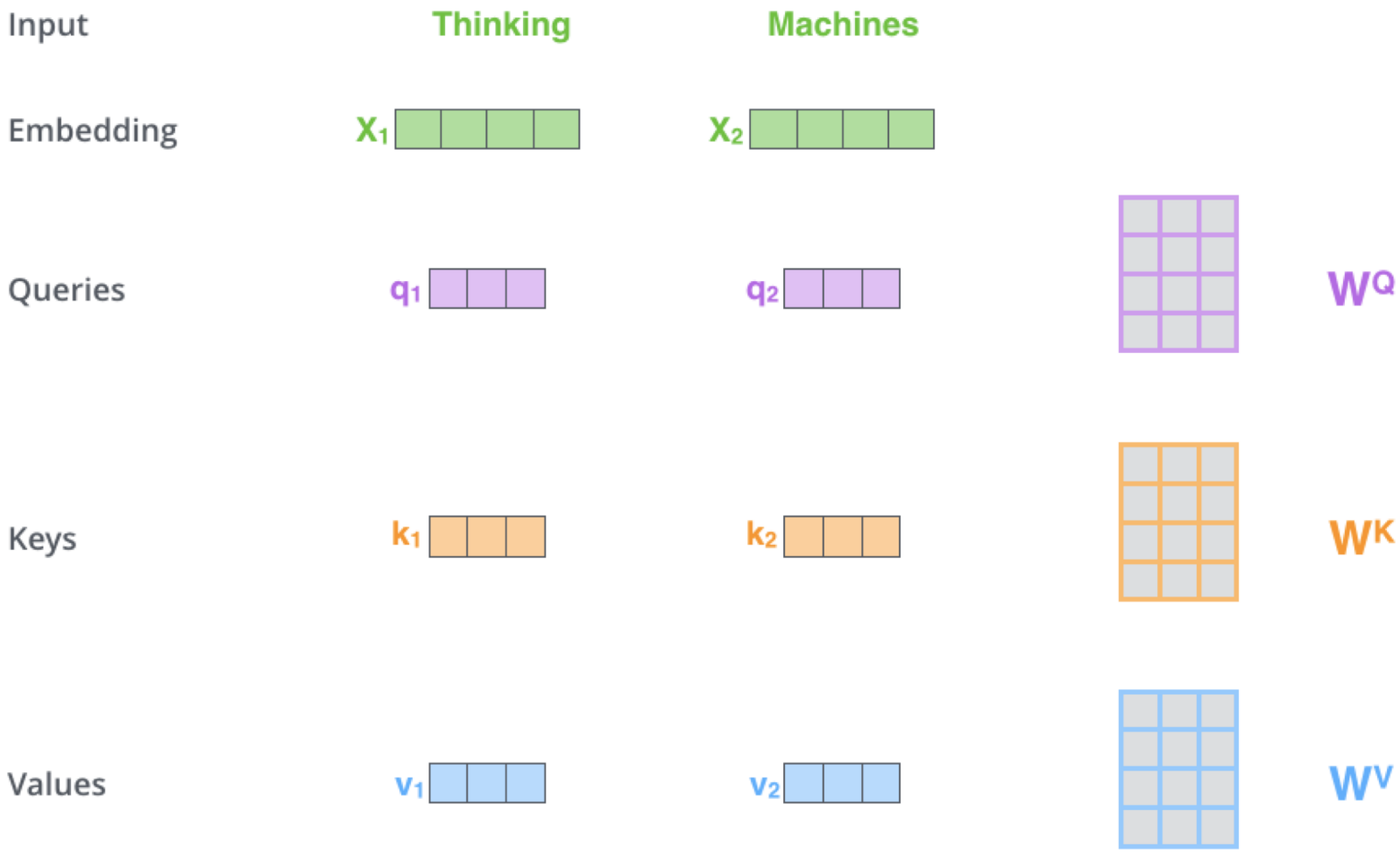
Example: "The animal didn't cross the street because it was too tired"

What does "it" in this sentence refer to? self-attention allows it to associate "it" with "animal".



Transformer Computation for Self Attention

So for each input token, we create a Query vector, a Key vector, and a Value vector.



Transformer Computation for Self-Attention: Compute Query, Key, and Value Matrices

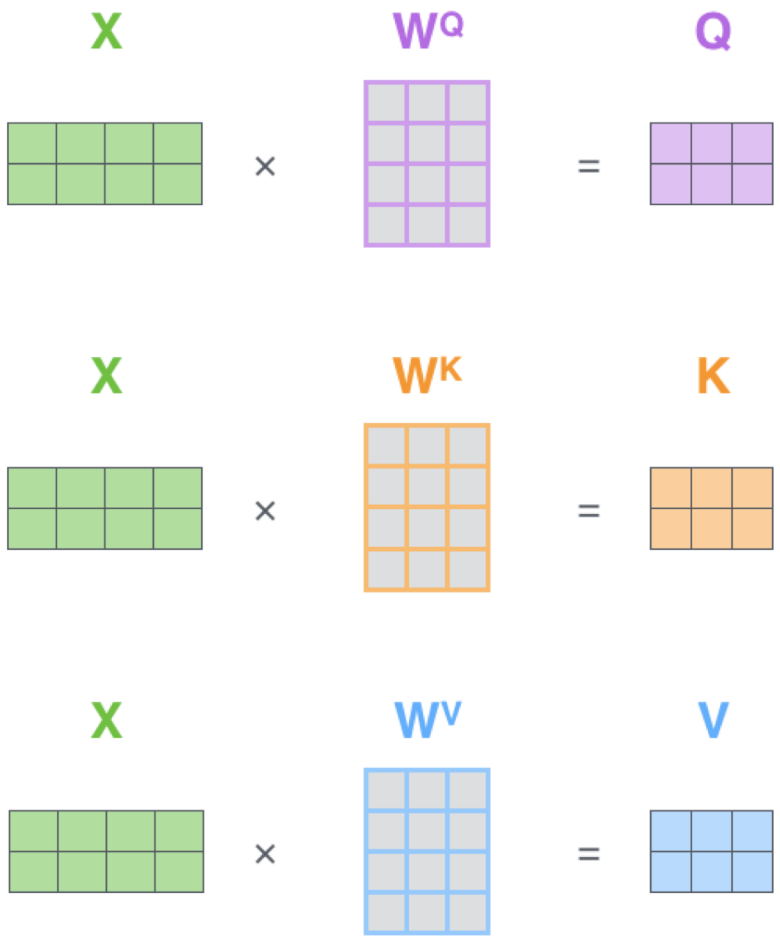
For 2 tokens x_1, x_2 , they form two rows of matrix X.

Multiply with 3 weight matrices, we get query, key, value matrices Q, K, V

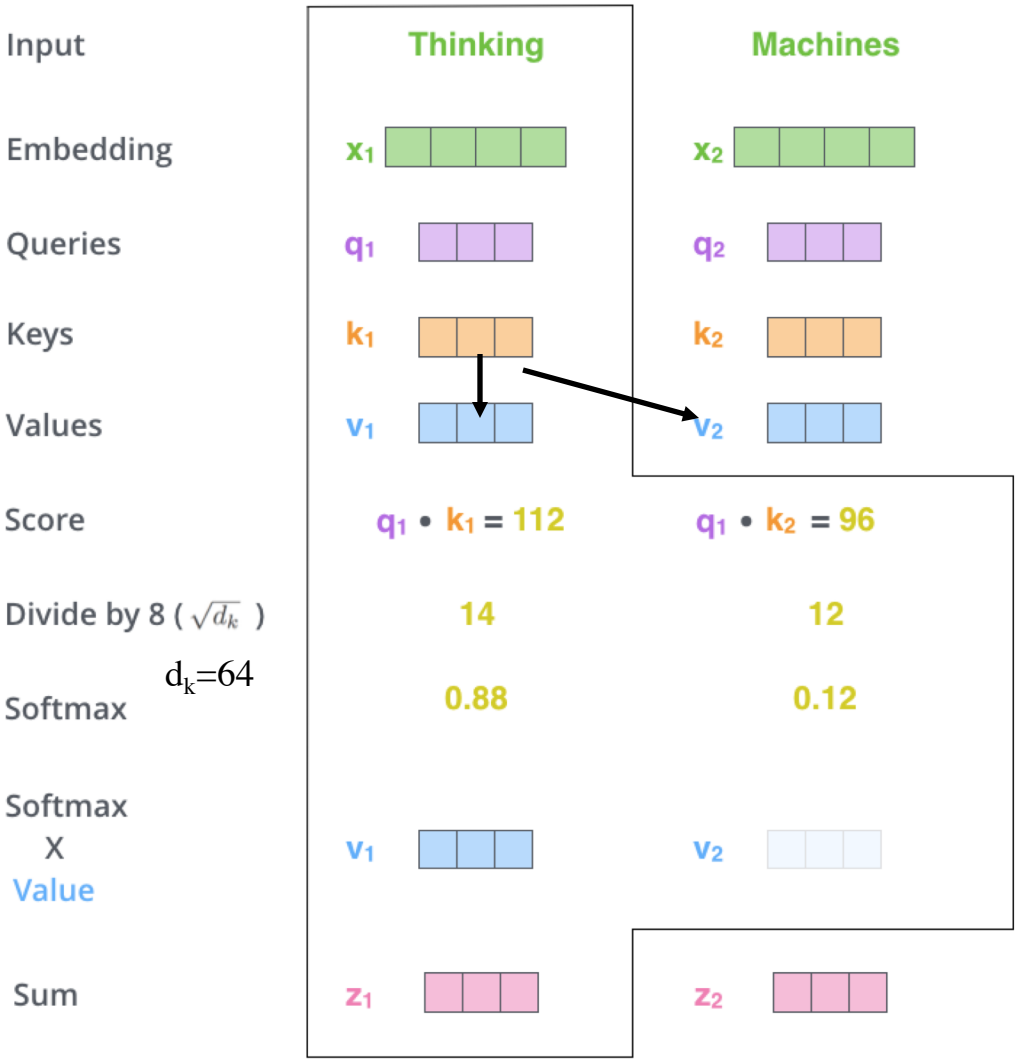
Q contains q_1, q_2

K contains k_1, k_2

V contains v_1, v_2



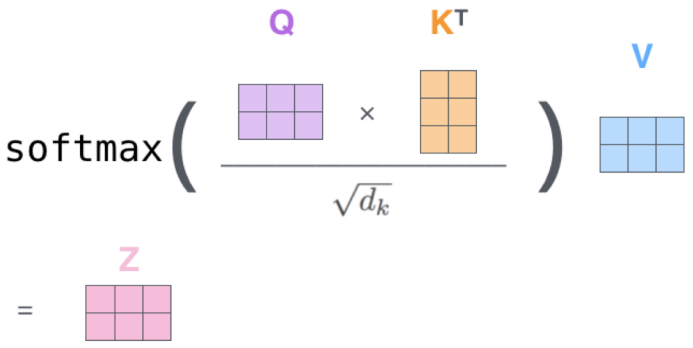
Example: Output of Self Attention



For 2 tokens x_1, x_2 of dimension d_k,

output z_1, z_2

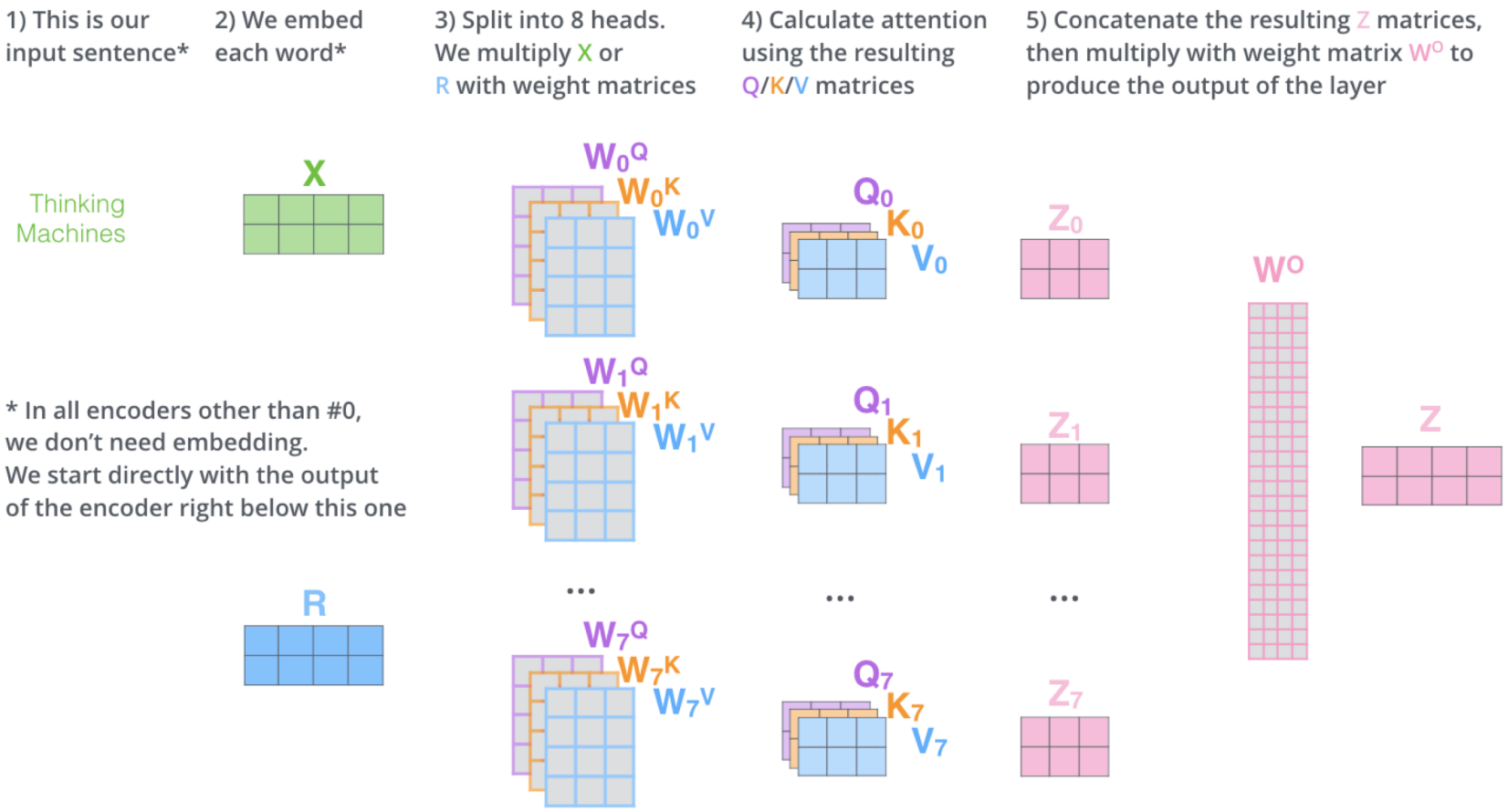
z_1 represents interaction of q_1 with k_1 , and interaction of q_1 with k_2



Multi-headed Attention: Assume 8 Heads

Repeat the same attention computation for different heads

Finally concatenate them together multiplied by weight matrix

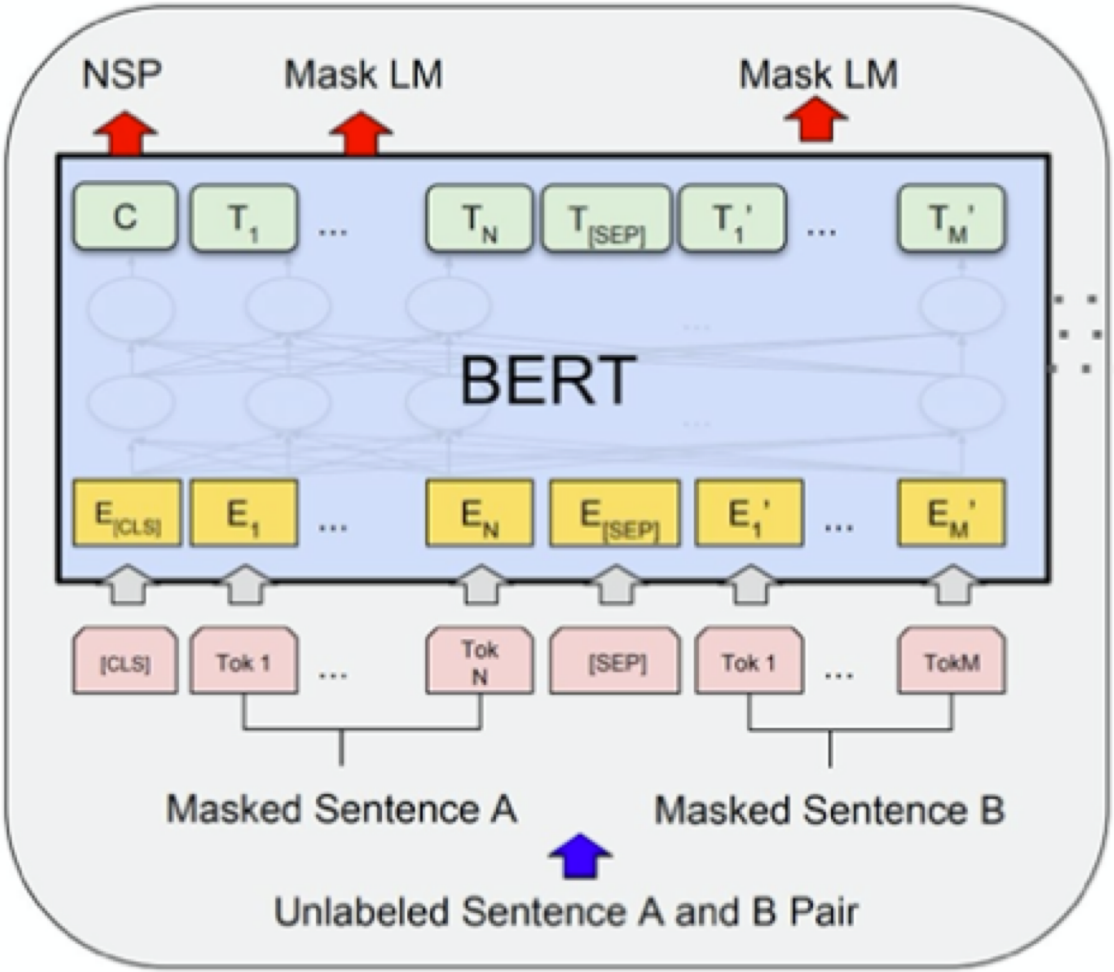


Outline

- Why BERT
- BERT Architecture
  - Self-attention computation
  - How to pre-train 
- BERT Applications
  - Classification, question answering, ad-hoc search

BERT Architecture and How to Pre-Train

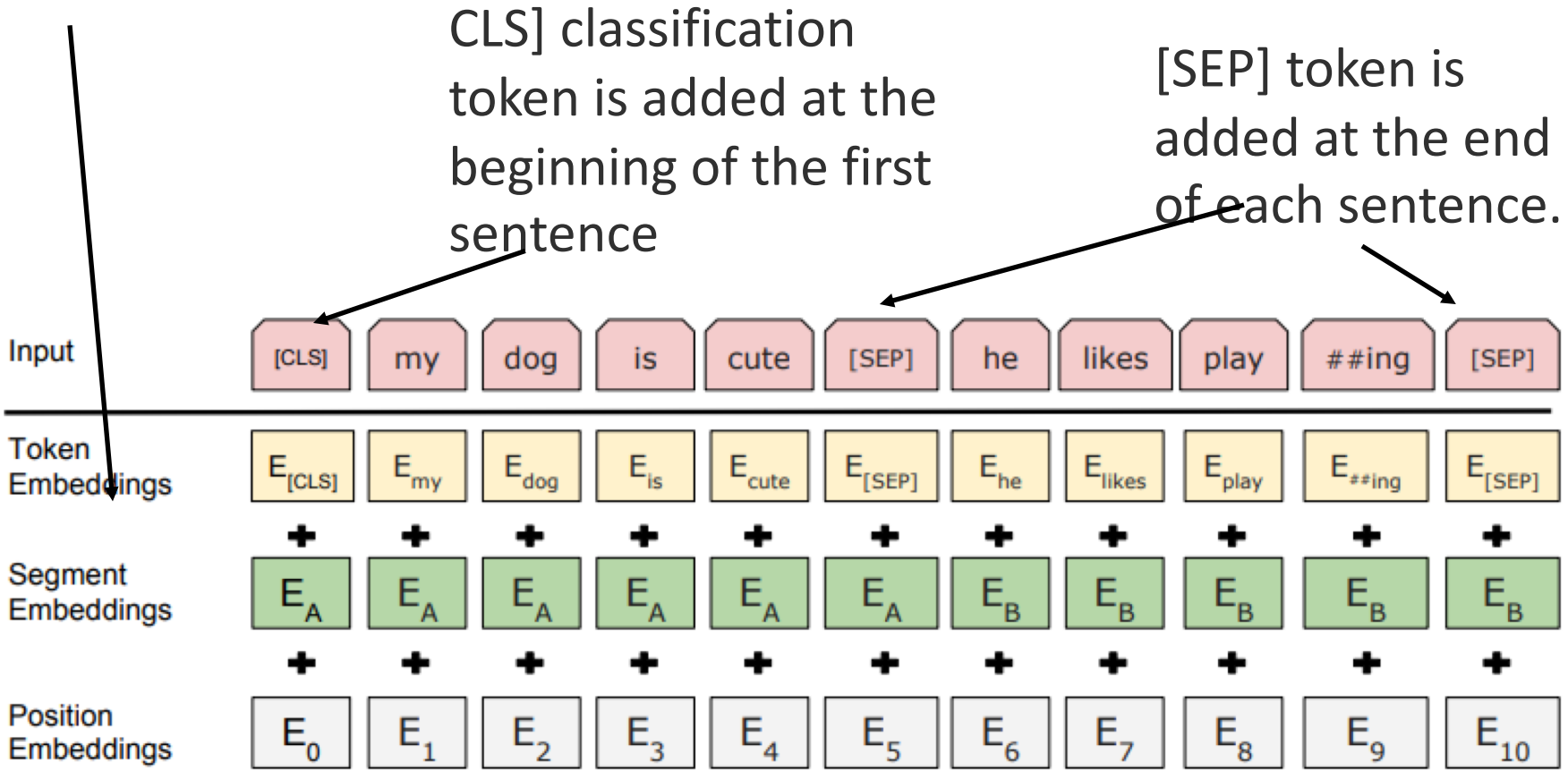
- Scan books (800M words) and Wikipedia (2,500M words)
- Unsupervised training
  - **Masked language model**: Predicted a masked word in a sentence
    - "my dog is ~~cute~~"
  - **Next sentence prediction (NSP)**: Sentence A appears Sentence B
    - my dog is cute, he likes playing



Pre-training

How to pre-train BERT

- Input token combines 3 embeddings.



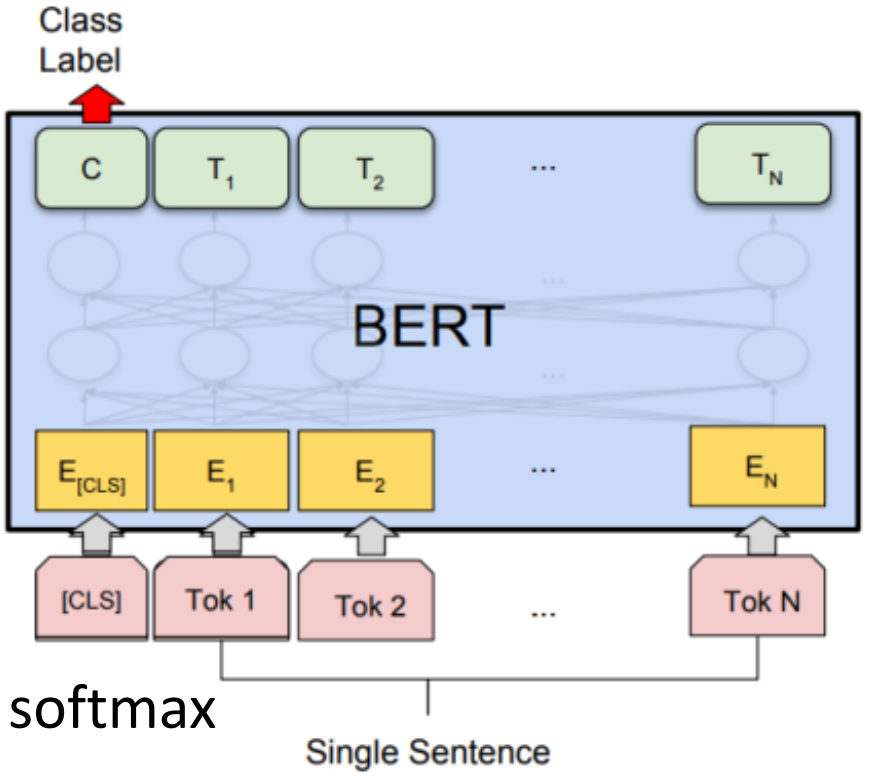
Embedding dimension of each token: 768 for BERT base model. #input tokens≤512

How to Use BERT for Applications

- Start with a pre-trained BERT model
- Add some computation layer on top of the core BERT model
  - **Fine-tune** with your own dataset by converting application input into the specific format used for BERT model

Example for sequence classification: Does a sentence contain personal attack?

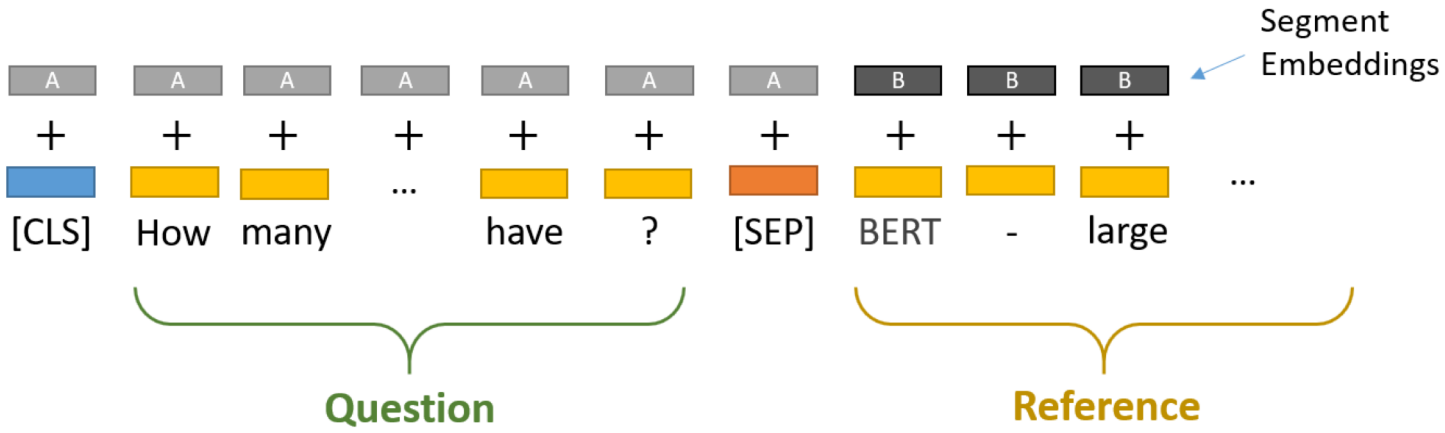
- Final hidden state **[CLS]** is the fixed-dimensional pooled representation of the input sequence, i.e. a sentence embedding capturing trained semantics



- Computation on [CLS} vector yields a softmax probability for the targeted class

BERT for Question Answering

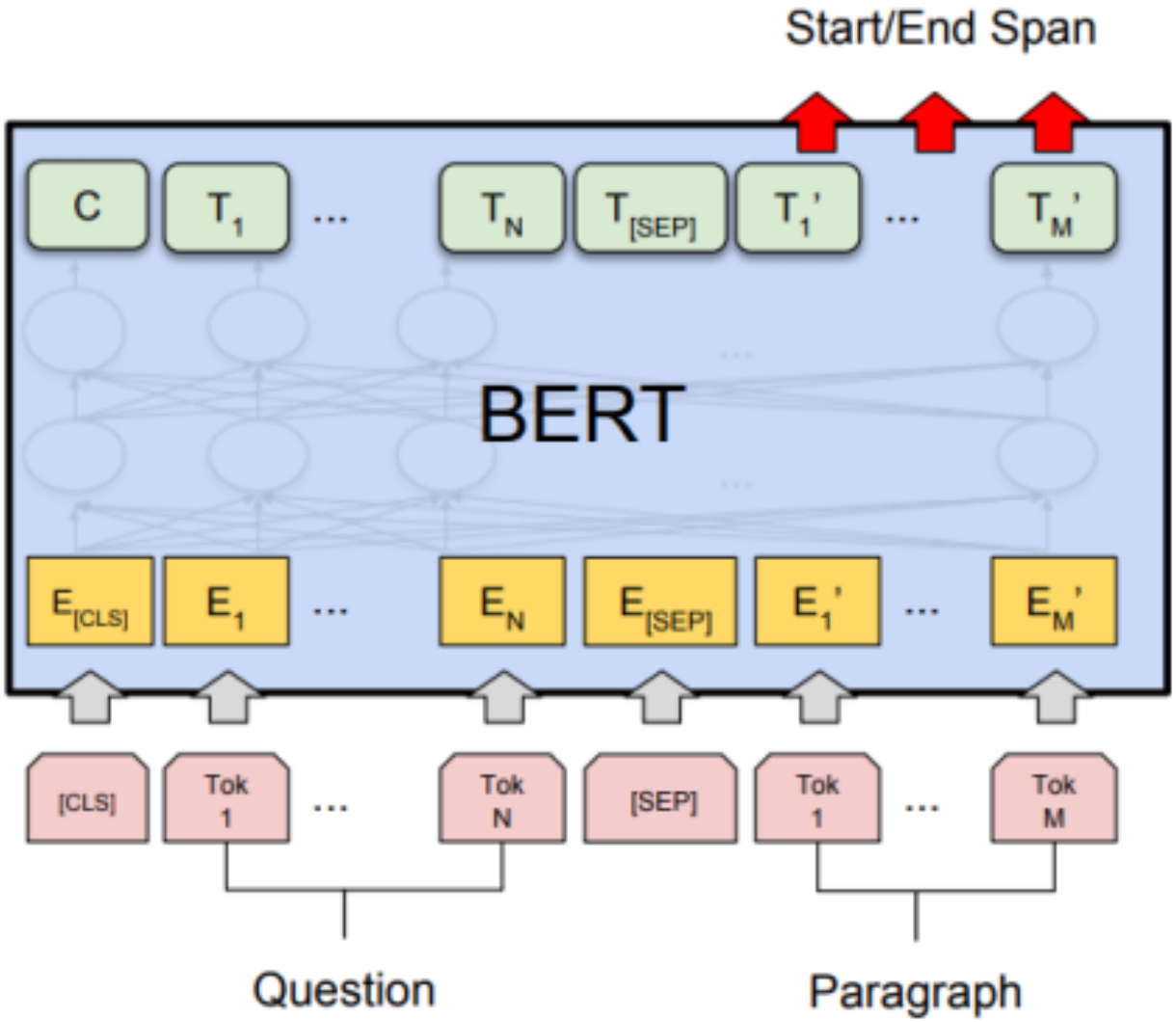
Given a question and a context paragraph, the model predicts a start and an end token from a reference paragraph that most likely answers the question.



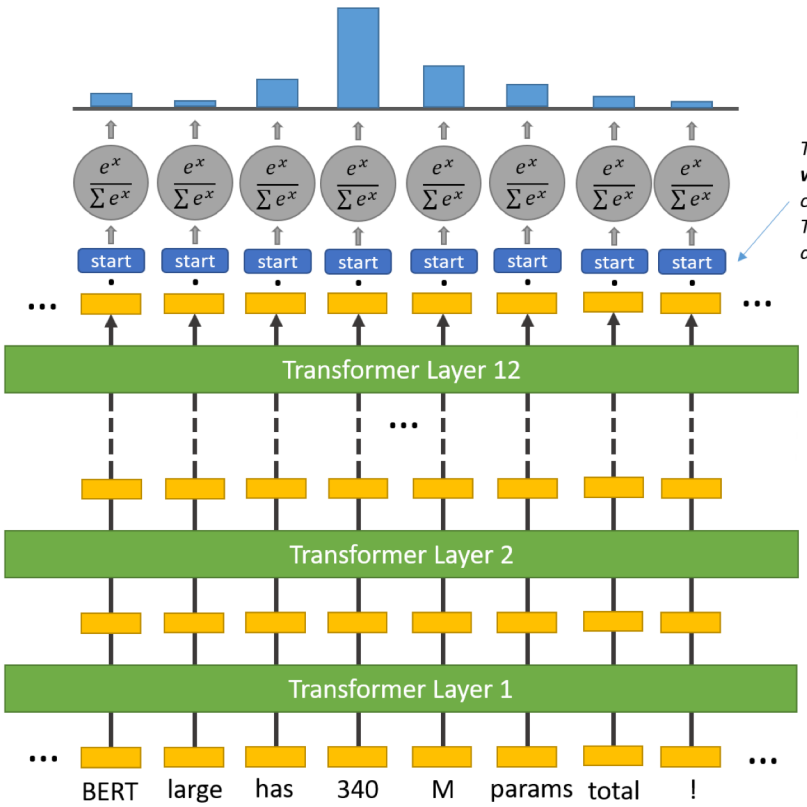
Question: How many parameters does BERT-large have?

Reference Text: BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

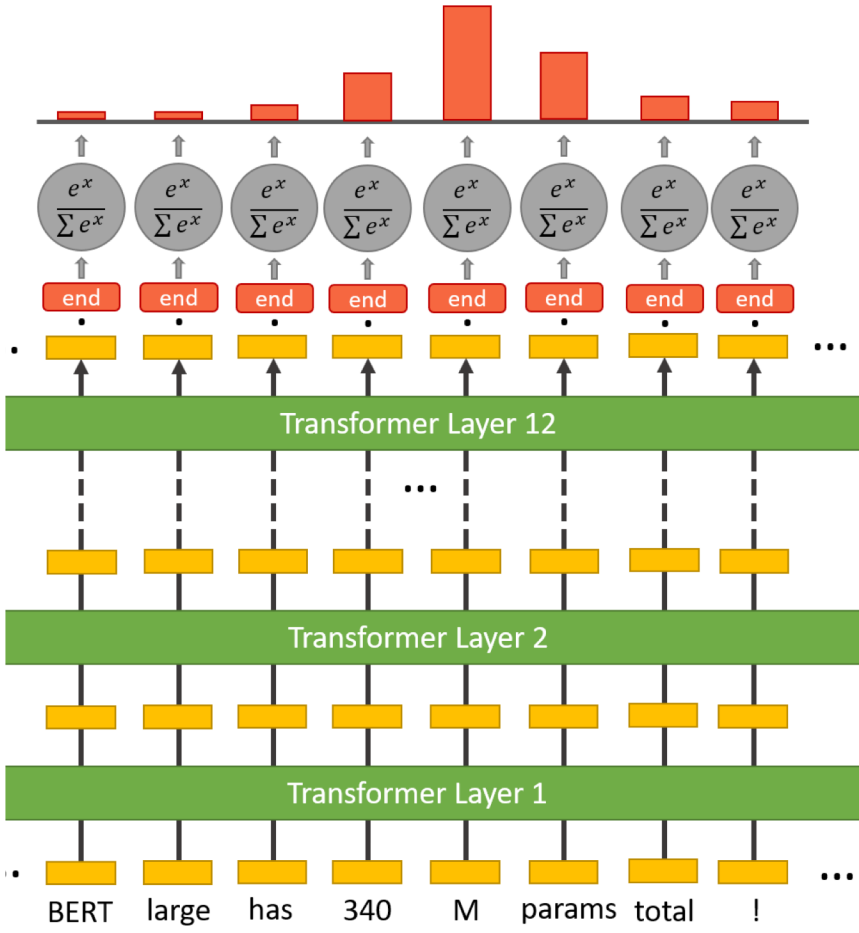
BERT for Question Answering



BERT for Question Answering: Find the ~~beginning and end of an answer~~



*This length 768 vector is the **weights** for the start token classifier.*
*The **same weights** are applied to **every position**.*



https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/

BERT Performance for QA

- Standford Question Answering Dataset (SQuAD) is a collection of 100k crowdsourced question/answer pairs
- Given a question and a paragraph from Wikipedia containing the answer, predict the answer text span in the paragraph.

System	Dev		Test	
	EM	F1	EM	F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

SQuAD 1,1 results from Google 2019 BERT paper by Jacob Devlin et al.

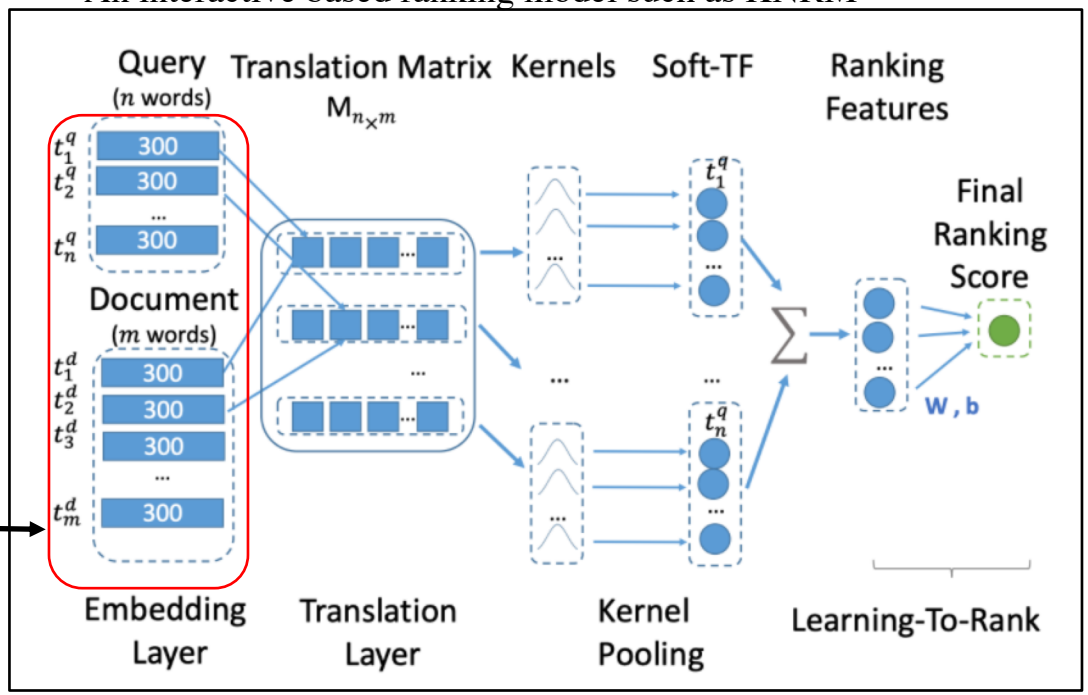
CEDR (MacAvaney et al., SIGIR'19) -
Integrate BERT contextual embedding with interaction based Neural Rankers

Key techniques:

- End-to-end training
- Use the BERT's intermediate outputs to represent words in query and document tokens

BERT_uncased_base

An interactive based ranking model such as KNRM

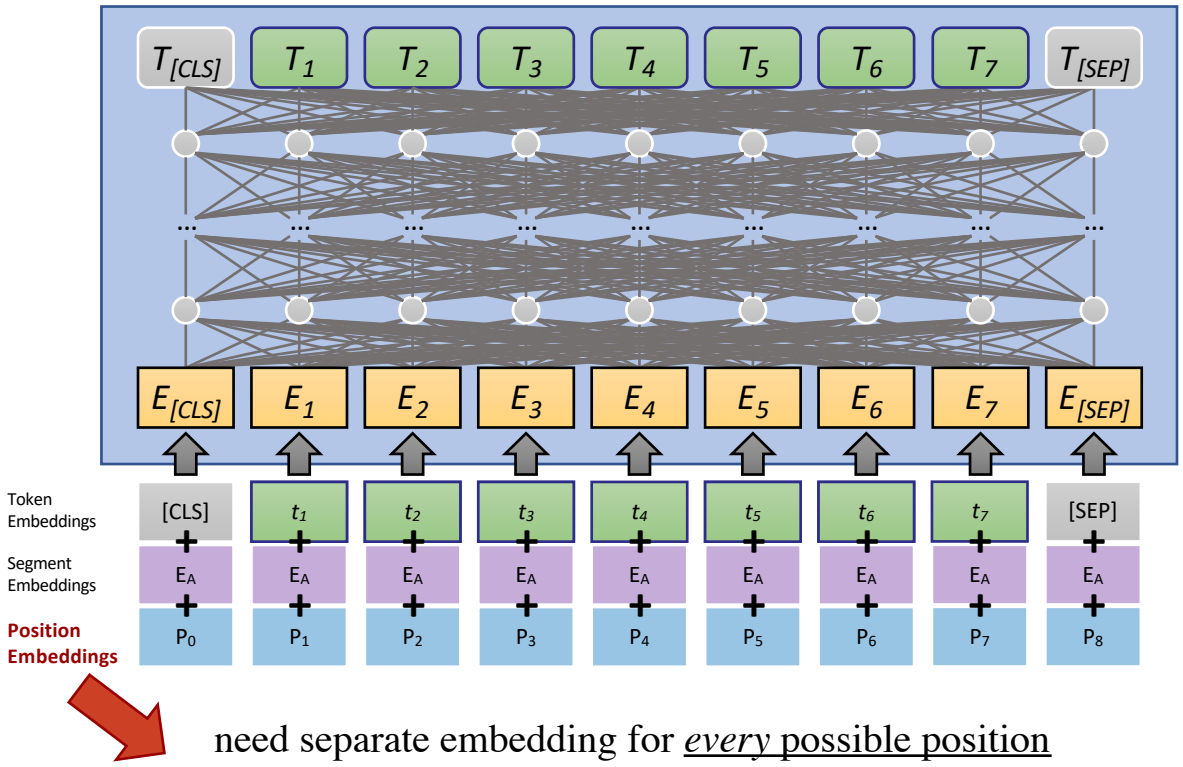


Inject BERT embeddings to KNRM and Conv-KNRM

Integrate BERT with KNRM and ConvKNRM for Document Ranking

		Robust04		WebTrack 2012–14	
Ranker	Input Representation	P@20	nDCG@20	nDCG@20	ERR@2
BM25	n/a	0.3123	0.4140	0.1970	0.147
SDM [13]	n/a	0.3749	0.4353	-	
TREC-Best	n/a	**0.4386**	**0.5030**	0.2855	**0.253**
ConvKNRM	GloVe	0.3349	0.3806	[B] 0.2547	[B] 0.183
Vanilla BERT	BERT (fine-tuned)	[BC] 0.4042	[BC] 0.4541	**[BC] 0.2895**	[BC] 0.221
PACRR	GloVe	0.3535	[C] 0.4043	0.2101	0.160
PACRR	ELMo	[C] 0.3554	[C] 0.4101	[BG] 0.2324	[BG] 0.188
PACRR	BERT	[C] 0.3650	[C] 0.4200	0.2225	0.181
PACRR	BERT (fine-tuned)	[BCVG] 0.4492	[BCVG] 0.5135	[BCG] 0.3080	[BCG] 0.233
CEDR-PACRR	BERT (fine-tuned)	**[BCVG] 0.4559**	**[BCVG] 0.5150**	**[BCVGN] 0.3373**	**[BCVGN] 0.265**
KNRM	GloVe	0.3408	0.3871	[B] 0.2448	0.175
KNRM	ELMo	[C] 0.3517	[CG] 0.4089	0.2227	0.168
KNRM	BERT	[BCG] 0.3817	[CG] 0.4318	[B] 0.2525	[B] 0.194
KNRM	BERT (fine-tuned)	[BCG] 0.4221	[BCVG] 0.4858	[BCVG] 0.3287	[BCVG] 0.255
CEDR-KNRM	BERT (fine-tuned)	**[BCVGN] 0.4667**	**[BCVGN] 0.5381**	**[BCVG] 0.3469**	**[BCVG] 0.277**
DRMM	GloVe	0.2892	0.3040	0.2215	0.160
DRMM	ELMo	0.2867	0.3137	[B] 0.2271	0.176
DRMM	BERT	0.2878	0.3194	[BG] 0.2459	[BG] 0.197
DRMM	BERT (fine-tuned)	[CG] 0.3641	[CG] 0.4135	[BG] 0.2598	[B] 0.185
CEDR-DRMM	BERT (fine-tuned)	**[BCVGN] 0.4587**	**[BCVGN] 0.5259**	**[BCVGN] 0.3497**	**[BCVGN] 0.262**

BERT's Limitations

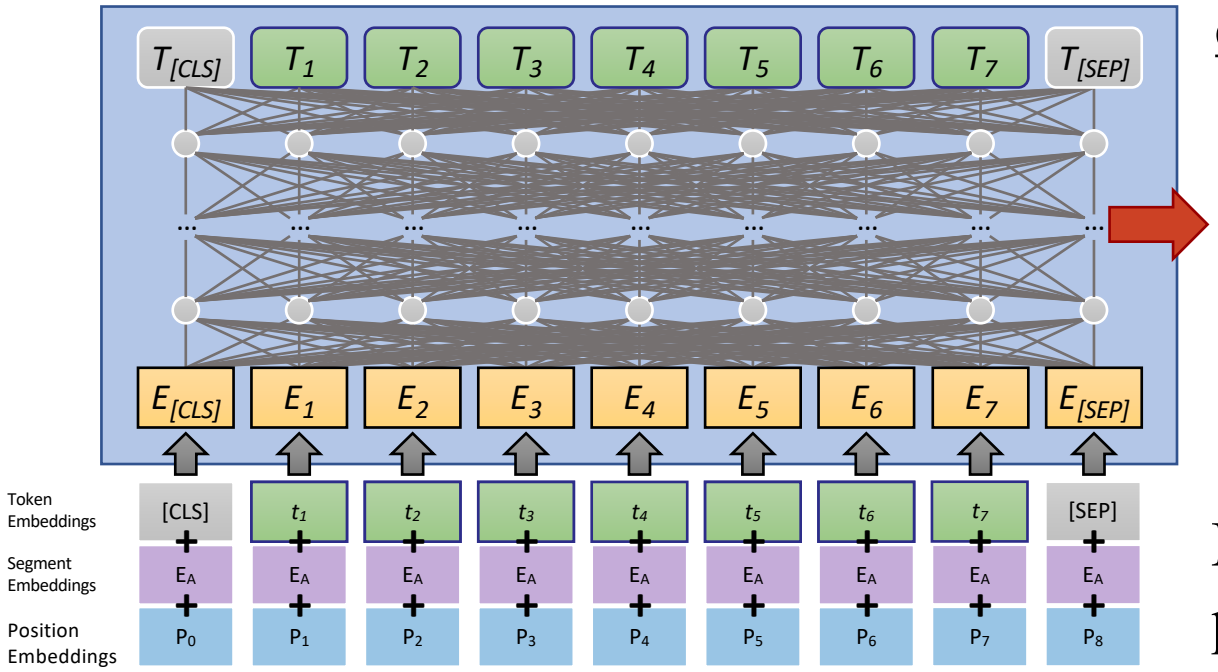


need separate embedding for *every* possible position
➔ restricted to indices 0-511

Cannot input entire documents

- what do we input?
- & how do we label it?

BERT's Limitations



Computationally expensive layers

➔ e.g., 110+ *million* learned weights

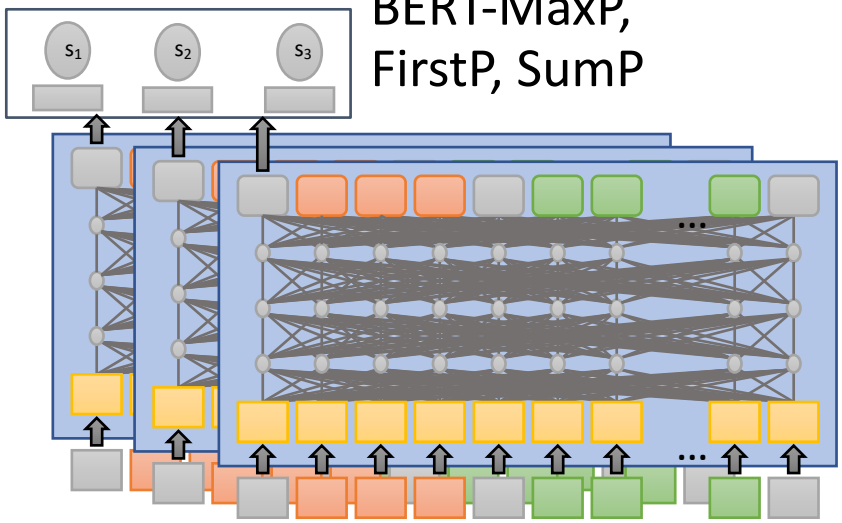
(later: *Beyond BERT & Dense Representations*)

Multi-stage ranking pipeline

- Identify candidate documents
- Rerank

How to Rank Long Documents?

1. **Score aggregation**
   a. Over **passage** scores. (Dai and Callan, SIGIR'19).



BERT-MaxP, FirstP, SumP

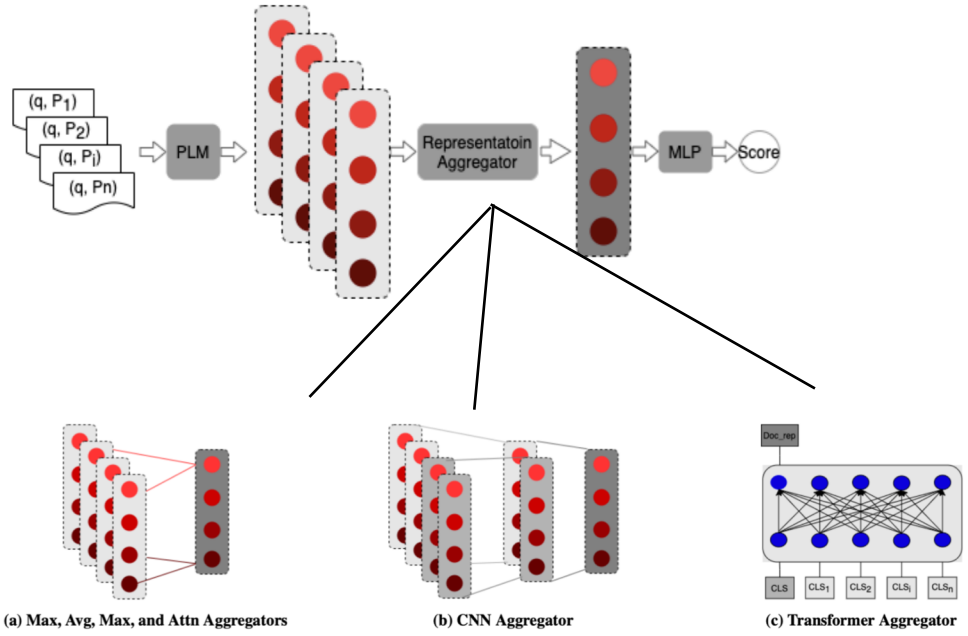
Model		Robust04 nDCG@20 Title	Robust04 nDCG@20 Description
(1)	BOW	0.417	0.409
(2)	SDM	0.427	0.427
(3)	LTR	0.427	0.441
(4a)	~~BERT–FirstP~~	~~0.444†~~	~~0.491†~~
(4b)	BERT–MaxP	**0.469**†	**0.529**†
(4c)	BERT–SumP	0.467†	0.524†

1. Over **passage** scores. *Dai, Callan. Deeper Text Understanding for IR with Contextual Neural Language Modeling. SIGIR 2019.*
2. Over **sentence** scores. *Yilmaz, Yang, Zhang, Lin. Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval. EMNLP '19.*

Long Documents - Representation Aggregation

PARADE (Li et al., 2020)

CNN and -Transformer perform well, (significant improvement over other aggregation)



	Robust04 Title		
	MAP	P@20	nDCG@20
BM25	0.2531†	0.3631†	0.4240†
BM25+RM3	0.3033†	0.3974†	0.4514†
Birch	0.3763	0.4749†	0.5454†
ELECTRA-MaxP	0.3183†	0.4337†	0.4959†
T5-3B (from [57])	-	-	-
ELECTRA-KNRM	0.3673†	0.4755†	0.5470†
CEDR-KNRM (Max)	0.3701†	0.4769†	0.5475†
PARADE-Avg	0.3352†	0.4464†	0.5124†
PARADE-Sum	0.3526†	0.4711†	0.5385†
PARADE-Max	0.3711†	0.4723†	0.5442†
PARADE-Attn	0.3462†	0.4576†	0.5266†
PARADE-CNN	**0.3807**	0.4821†	0.5625
PARADE-Transformer	0.3803	**0.4920**	**0.5659**

Summary

- Empirical results from BERT are very impressive for various NLP tasks
- With contextual pretraining in a big dataset, bigger == better
  - Good results on pre-training is >1,000x to 100,000 more expensive than supervised training.
- **Challenges**
  - Extremely expensive to run BERT
  - Active research for ad-hoc search queries