



Amirkabir University of Technology
(Tehran Polytechnic)

Natural Language Processing

Lecture 13: Contextualized Representation (Pre-trained Models)

Amirkabir University of Technology

Dr Momtazi

Outline

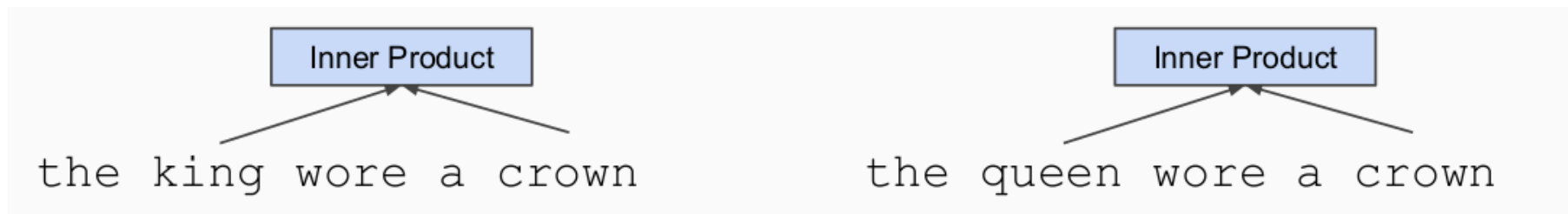
- **Introduction**
- ELMO
- BERT

Pre-training in NLP

- Word embeddings are the basis of deep learning for NLP

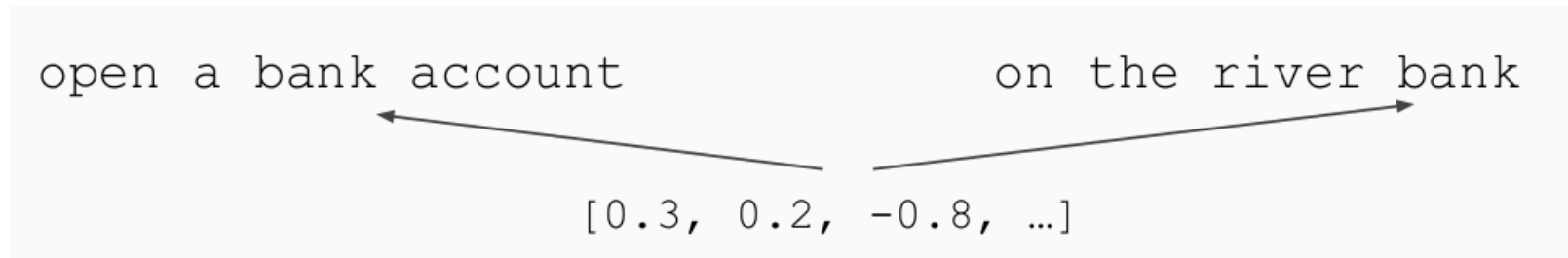


- Word embeddings (word2vec, GloVe) are often *pre-trained* on text corpus from co-occurrence statistics

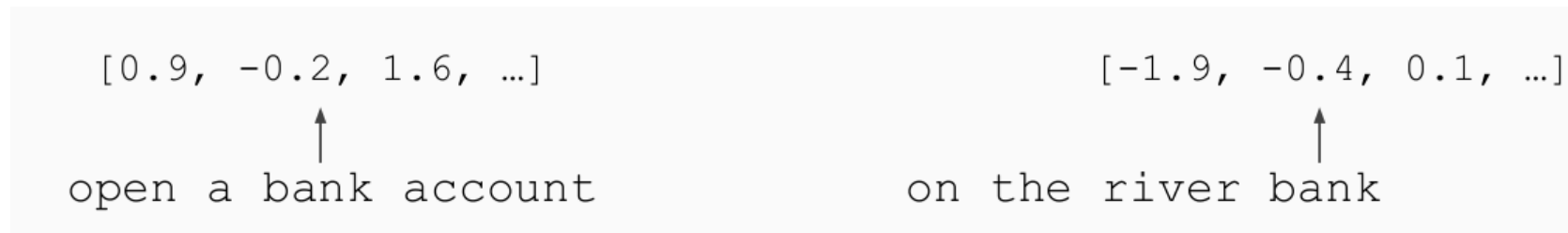


Contextual Representations

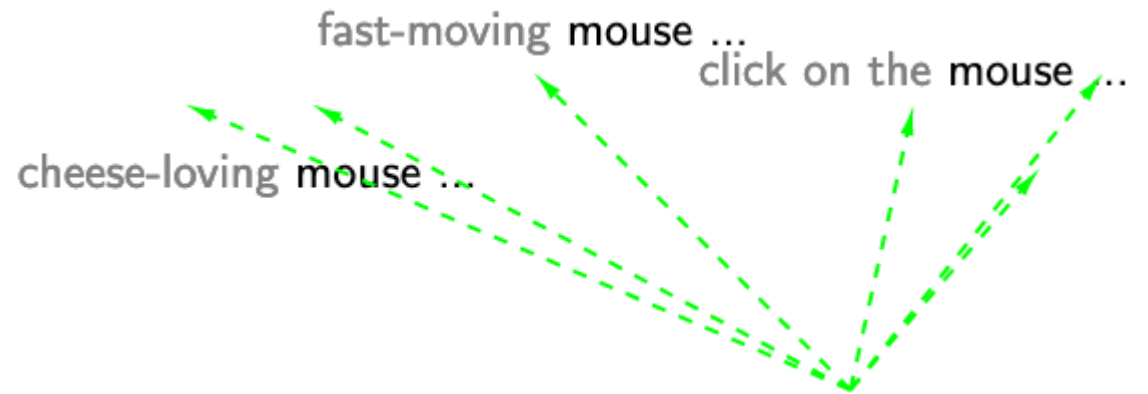
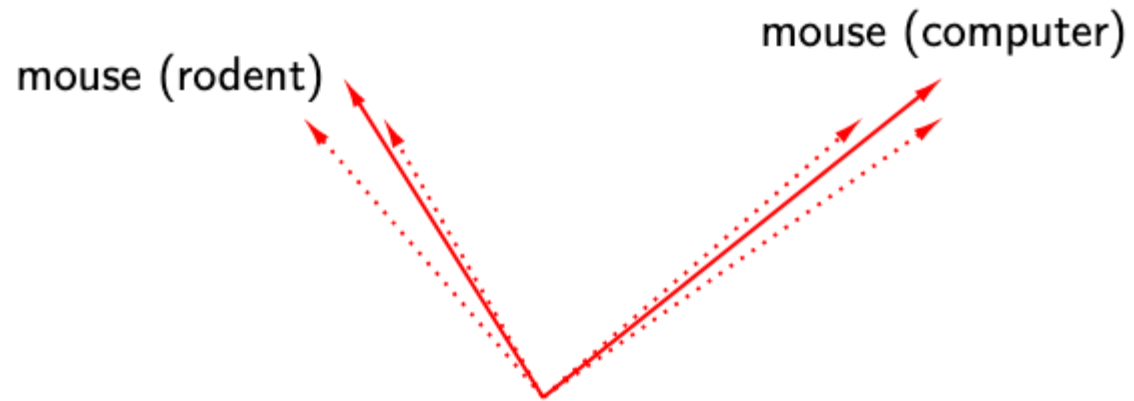
- **Problem:** Word embeddings are applied in a context free manner



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



Static vs Contextualized Representation

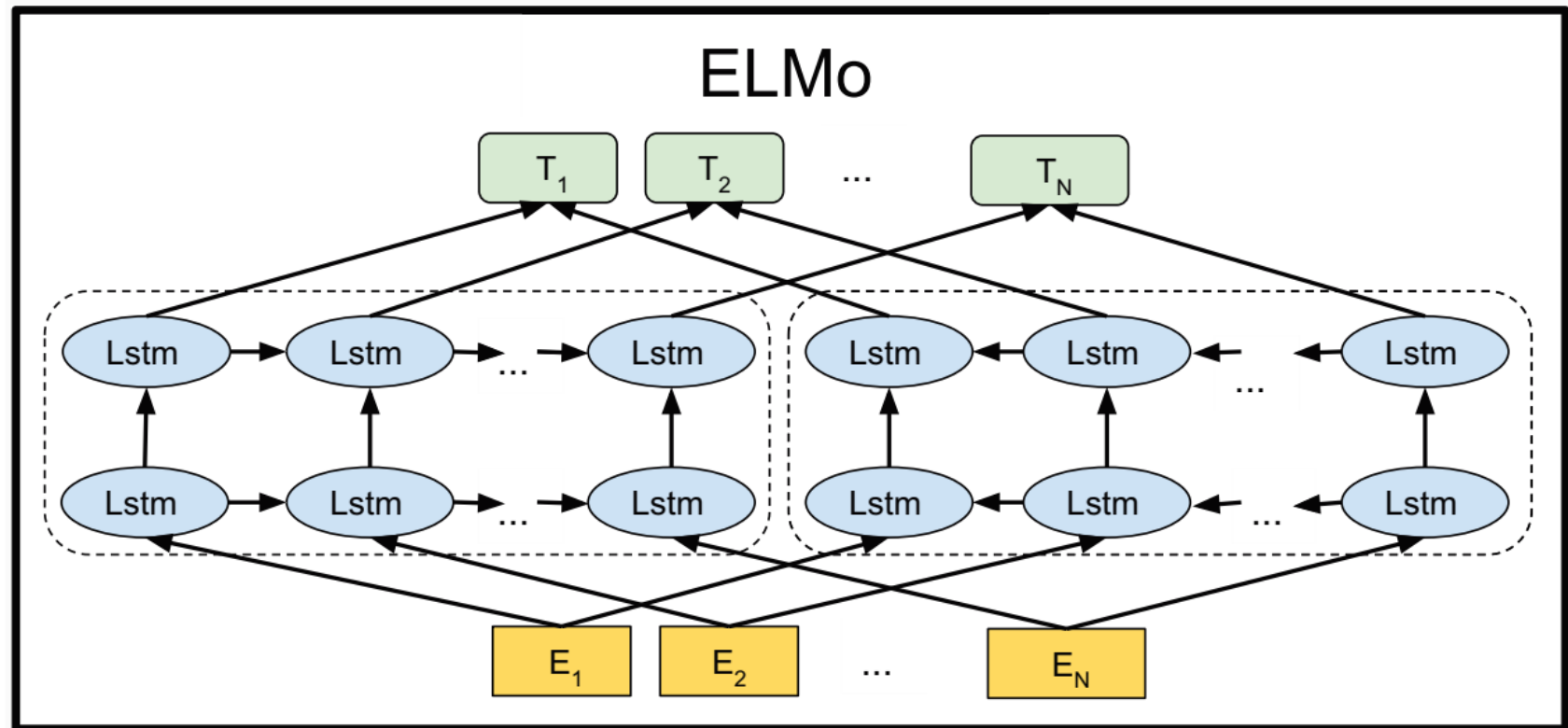


Outline

- Introduction
- **ELMO**
- BERT

ELMO

- Bidirectional language model
- Train Separate Left-to-Right and Right-to-Left LMs



ELMO

- Bidirectional training

- Forward language model

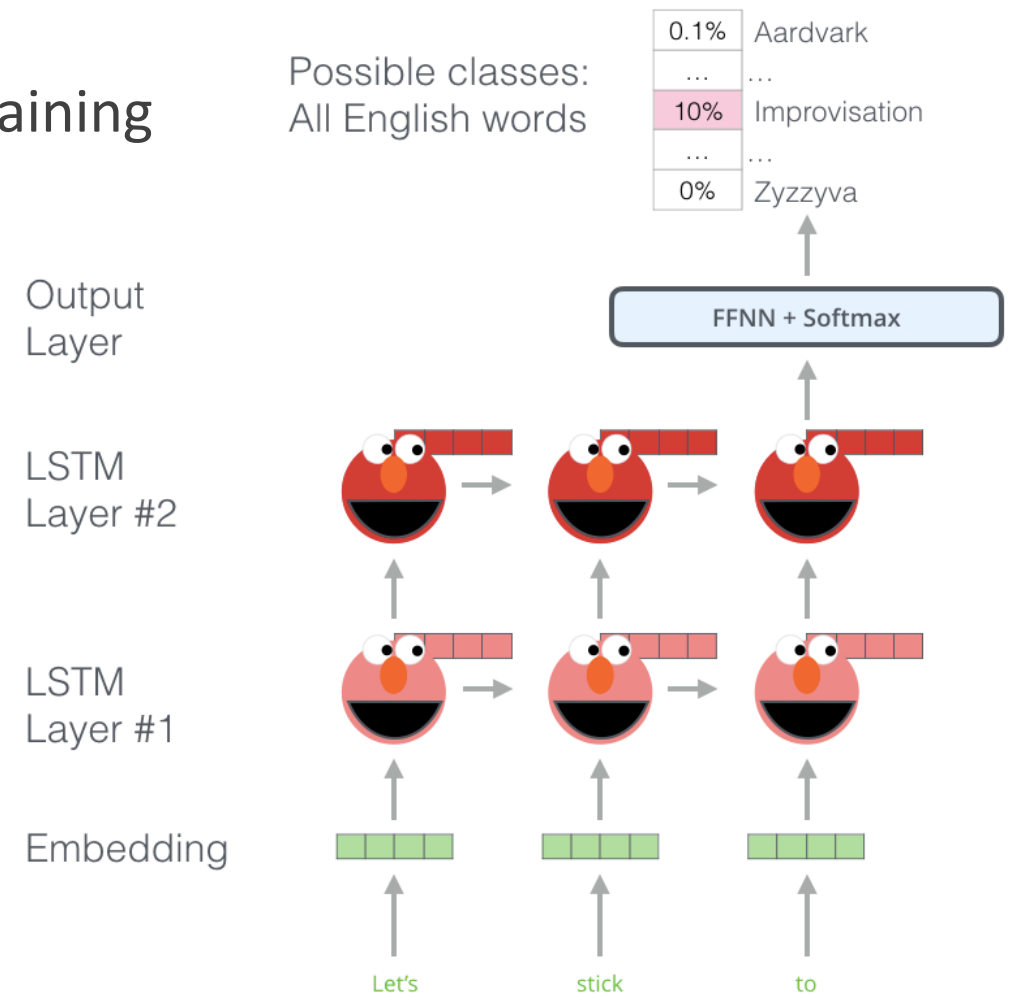
$$p(t_1, t_2; , \dots, t_N) = \prod_{k=1}^N p(t_k | t_1, t_2; , \dots, t_{k-1})$$

- Backward language model

$$p(t_1, t_2; , \dots, t_N) = \prod_{k=1}^N p(t_k | t_{k+1}, t_{k+2}; , \dots, t_N)$$

ELMO

- Predicting the next word in forward training
- Predicting the previous word in backward training



ELMO

- Extracting embedding

1- Concatenate hidden layers



2- Multiply each vector by a weight based on the task

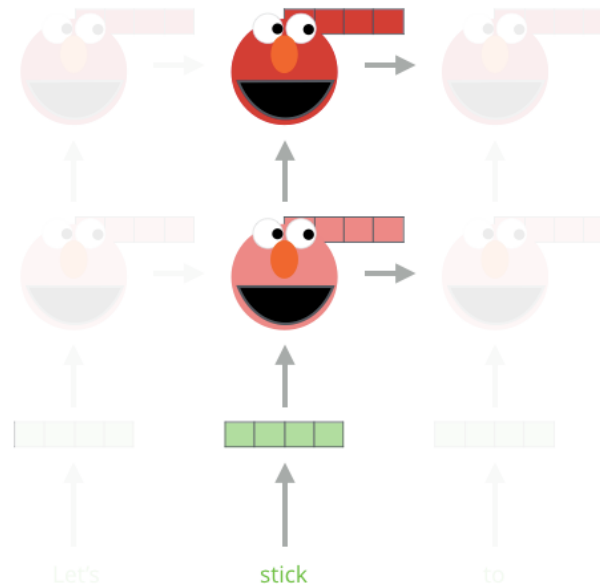


3- Sum the (now weighted) vectors

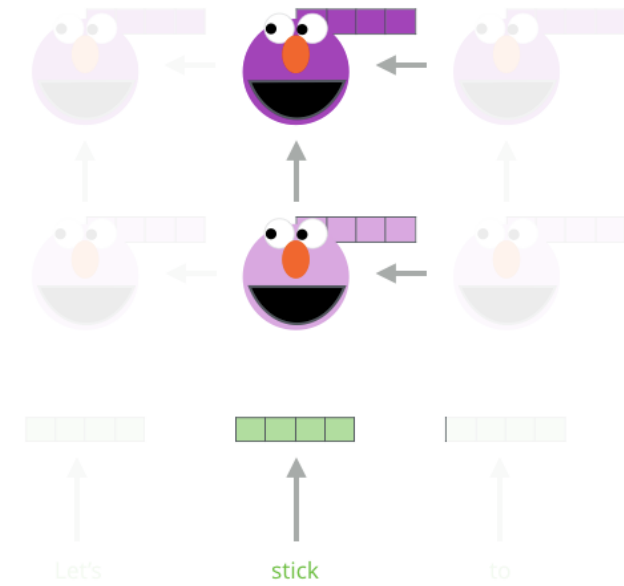


ELMo embedding of "stick" for this task in this context

Forward Language Model



Backward Language Model



ELMO

- Extracting embedding

ELMo is a task specific representation. A down-stream task learns weighting parameters

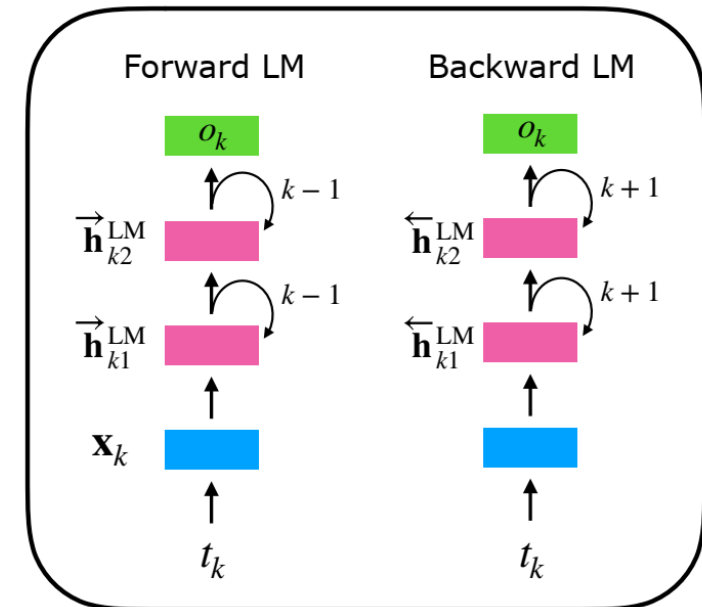
$$\text{ELMo}_k^{\text{task}} = \gamma^{\text{task}} \times \sum \left\{ \begin{array}{l} s_2^{\text{task}} \times \mathbf{h}_{k2}^{\text{LM}} \\ s_1^{\text{task}} \times \mathbf{h}_{k1}^{\text{LM}} \\ s_0^{\text{task}} \times \mathbf{h}_{k0}^{\text{LM}} \end{array} \right. \quad \left([\mathbf{x}_k; \mathbf{x}_k] \right)$$

Concatenate hidden layers

$[\vec{\mathbf{h}}_{kj}^{\text{LM}}; \overleftarrow{\mathbf{h}}_{kj}^{\text{LM}}]$

Unlike usual word embeddings, ELMo is assigned to every *token* instead of a *type*

biLMs

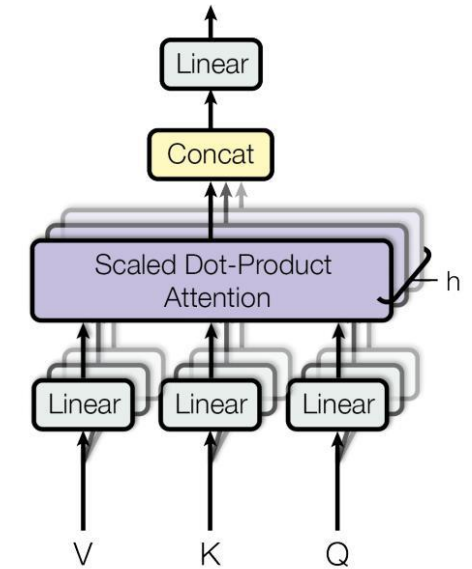
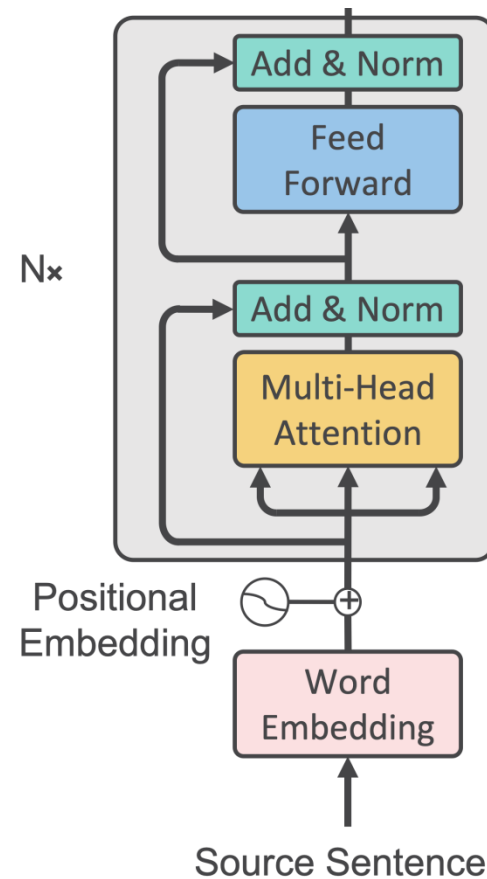


Outline

- Introduction
- ELMO
- **BERT**

Model Architecture

- Multi-headed self attention
 - Models context
- Feed-forward layers
 - Computes non-linear hierarchical features
- Layer norm and residuals
 - Makes training deep networks healthy
- Positional embeddings
 - Allows model to learn relative positioning



Model Architecture

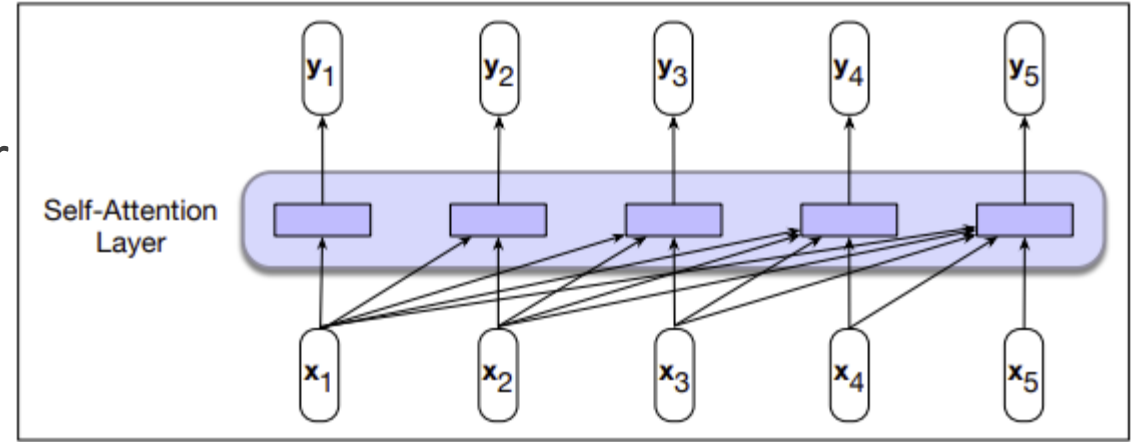
- Empirical advantages of Transformer vs. LSTM:
 - Self-attention == no locality bias
 - Long-distance context has “equal opportunity”
 - Single multiplication per layer == efficiency on TPU
 - Effective batch size is number of *words*, not *sequences*

Problem with Previous Methods

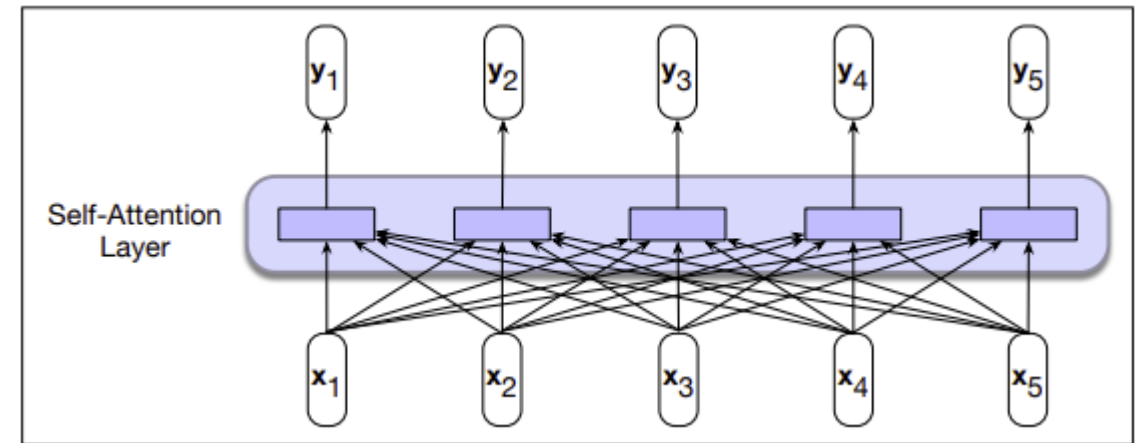
- **Problem:** Language models only use left context *or* right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
- Reason 1: Directionality is needed to generate a well-formed probability distribution.
 - We don't care about this.
- Reason 2: Words can “see themselves”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

Bidirectional Transformer

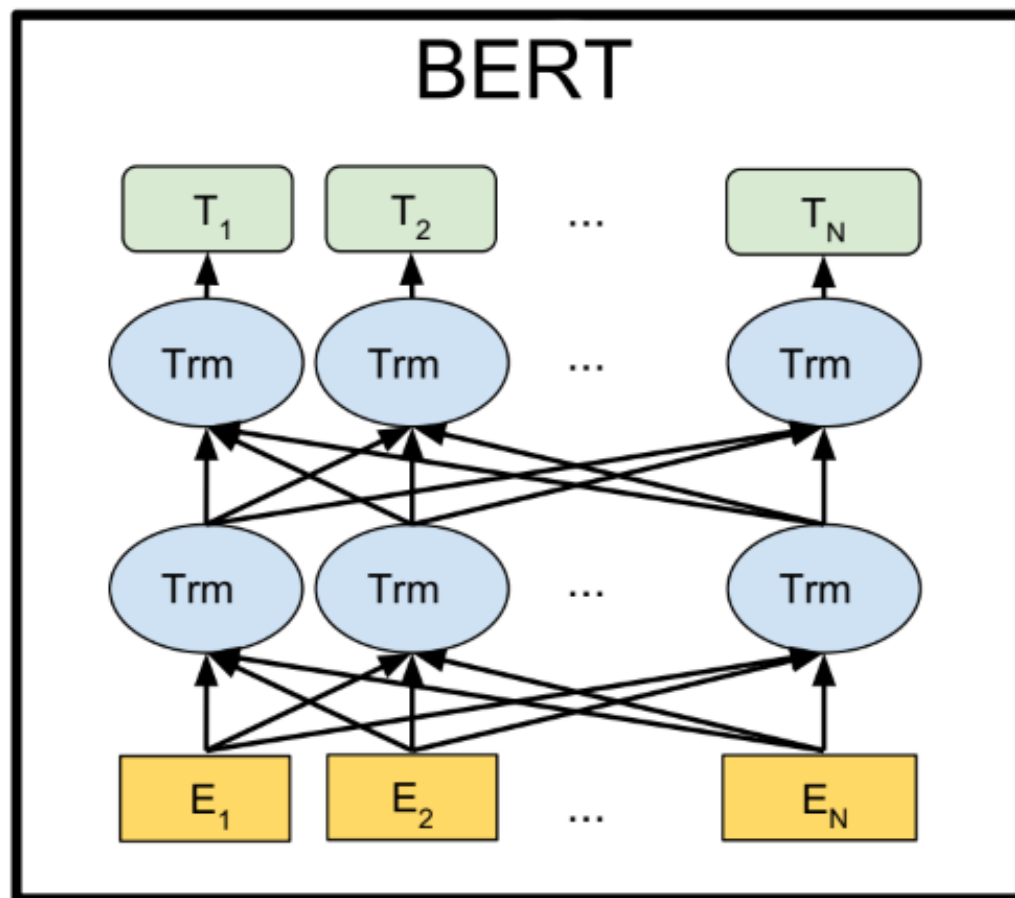
- A causal backward looking transformer



- A bidirectional transformer



BERT

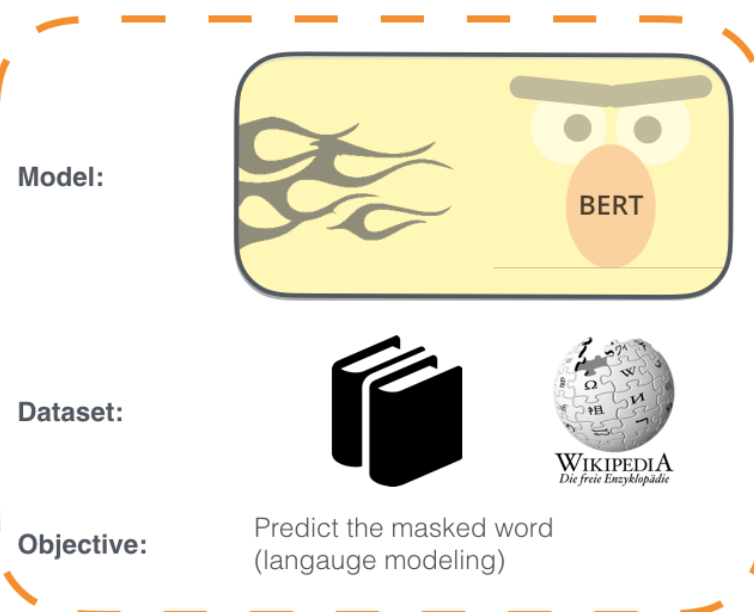


BERT

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

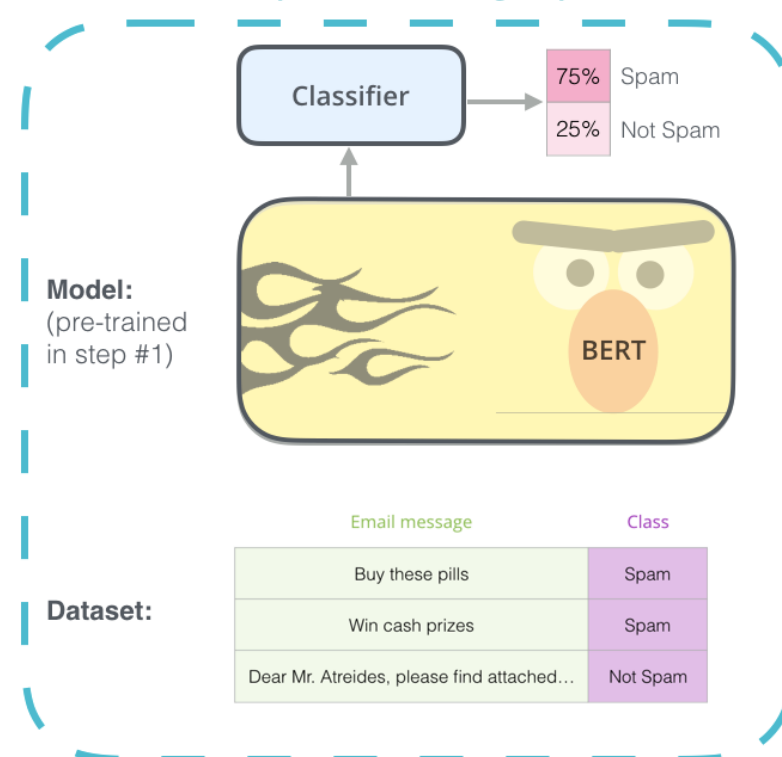
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

Semi-supervised Learning Step



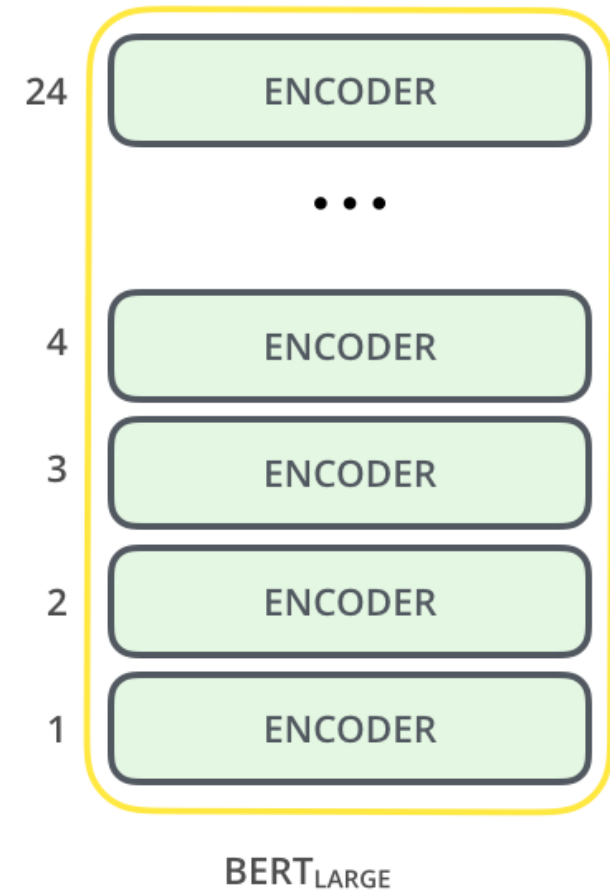
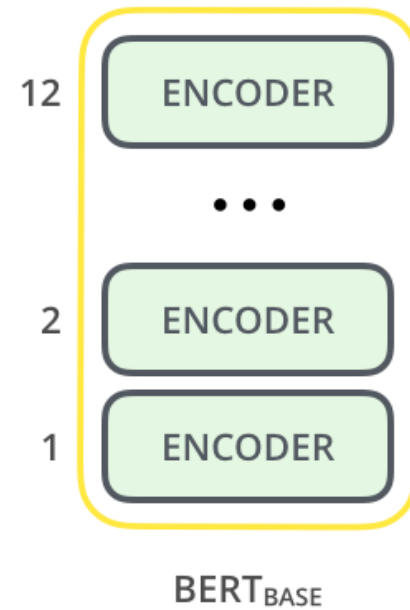
2 - Supervised training on a specific task with a labeled dataset.

Supervised Learning Step



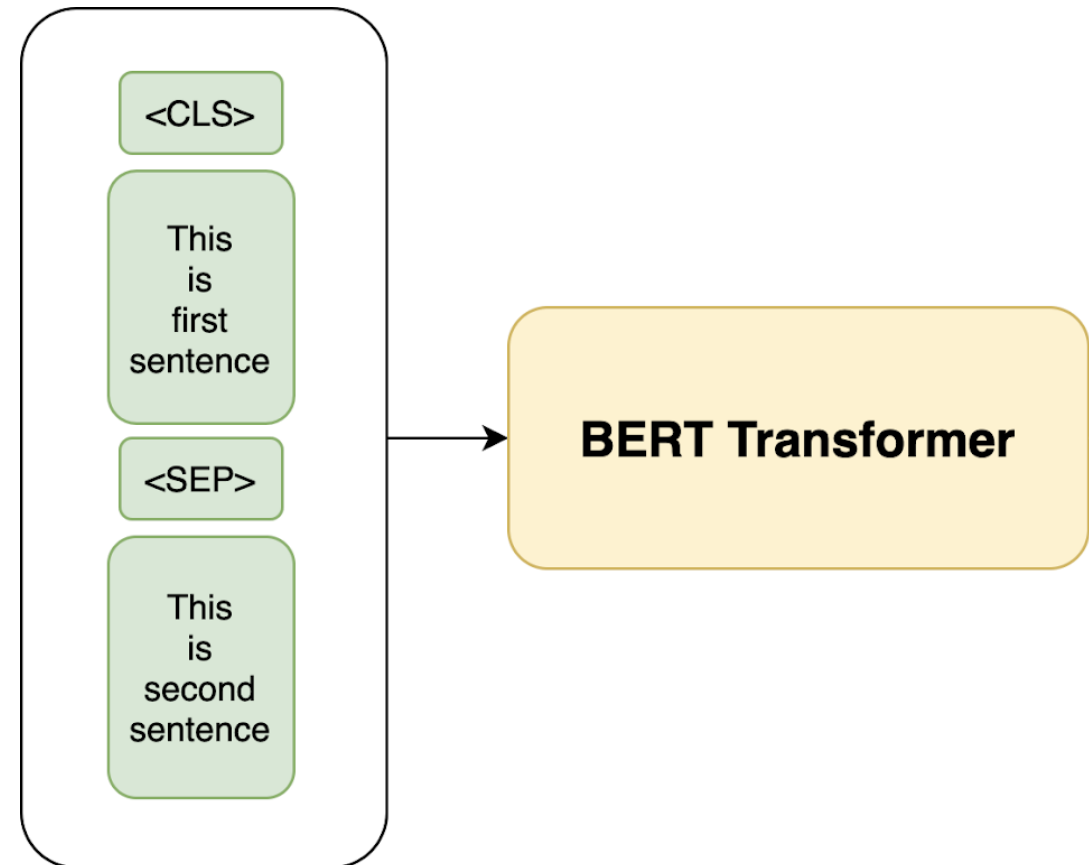
BERT

- BERT Base
- BERT Large

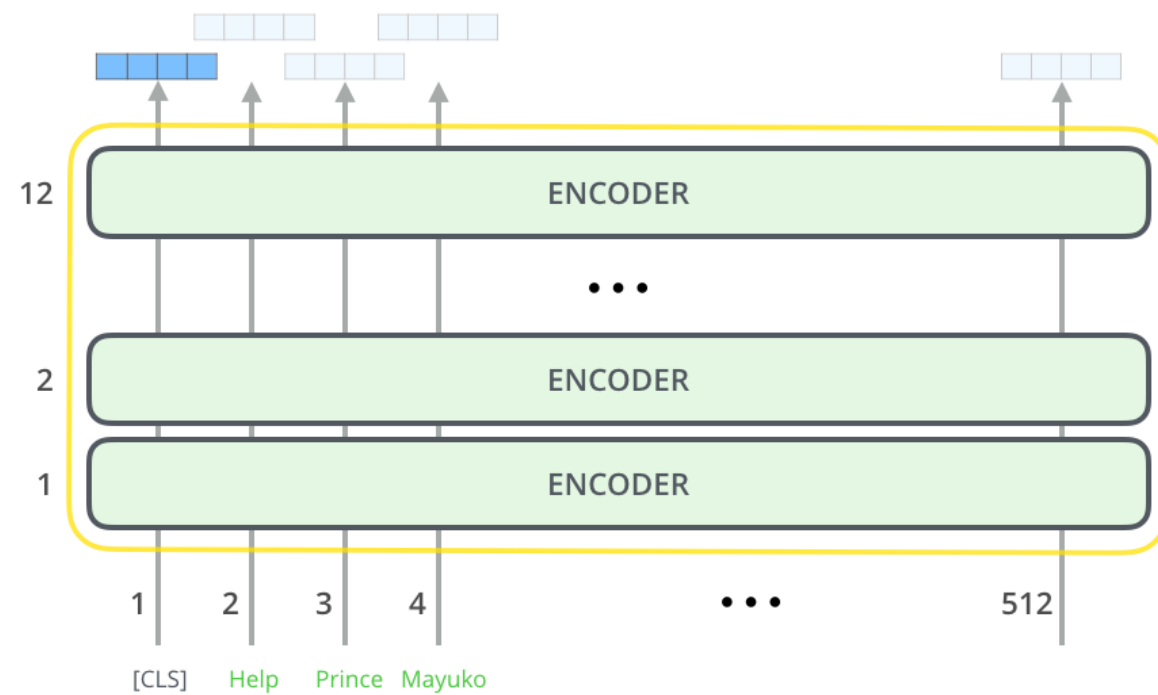
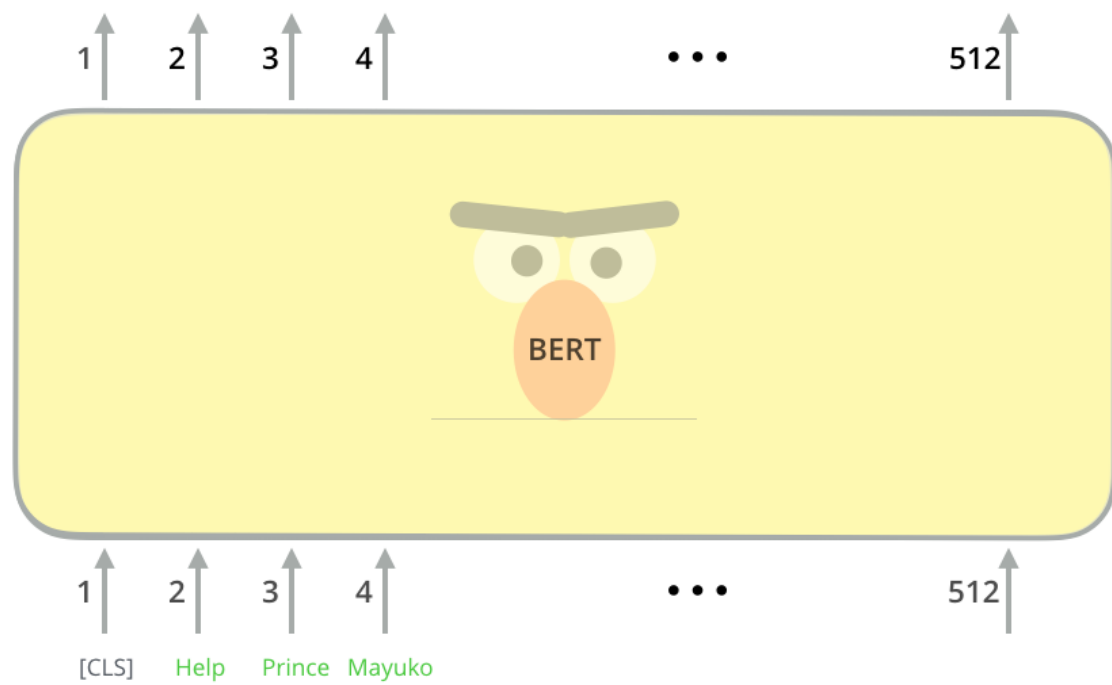


BERT

- Training type
 - Masked language model
 - Next sentence prediction
- Input
 - First token [CLS]
 - Delimiter token [SEP]
 - Masked token [MASK]



BERT



BERT

- Masked language model
- **Solution:** Mask out $k\%$ of the input words, and then predict the masked words
 - We always use $k = 15\%$

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

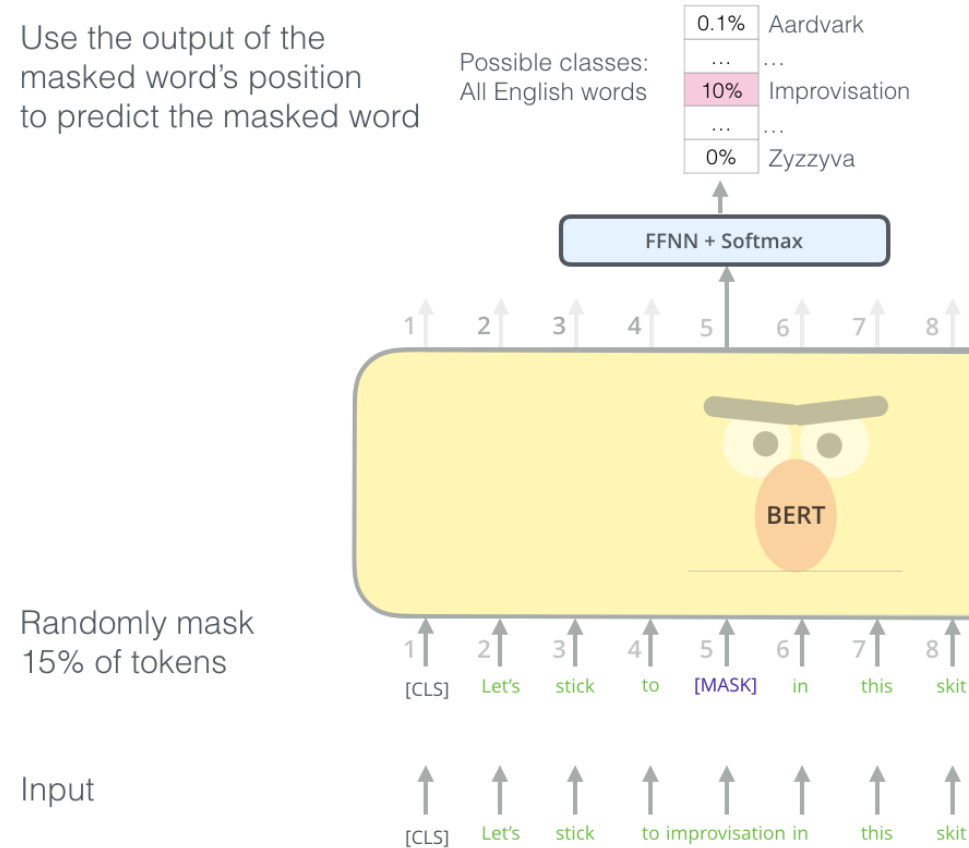
store gallon

↑ ↑

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

BERT

- Masked language model



BERT

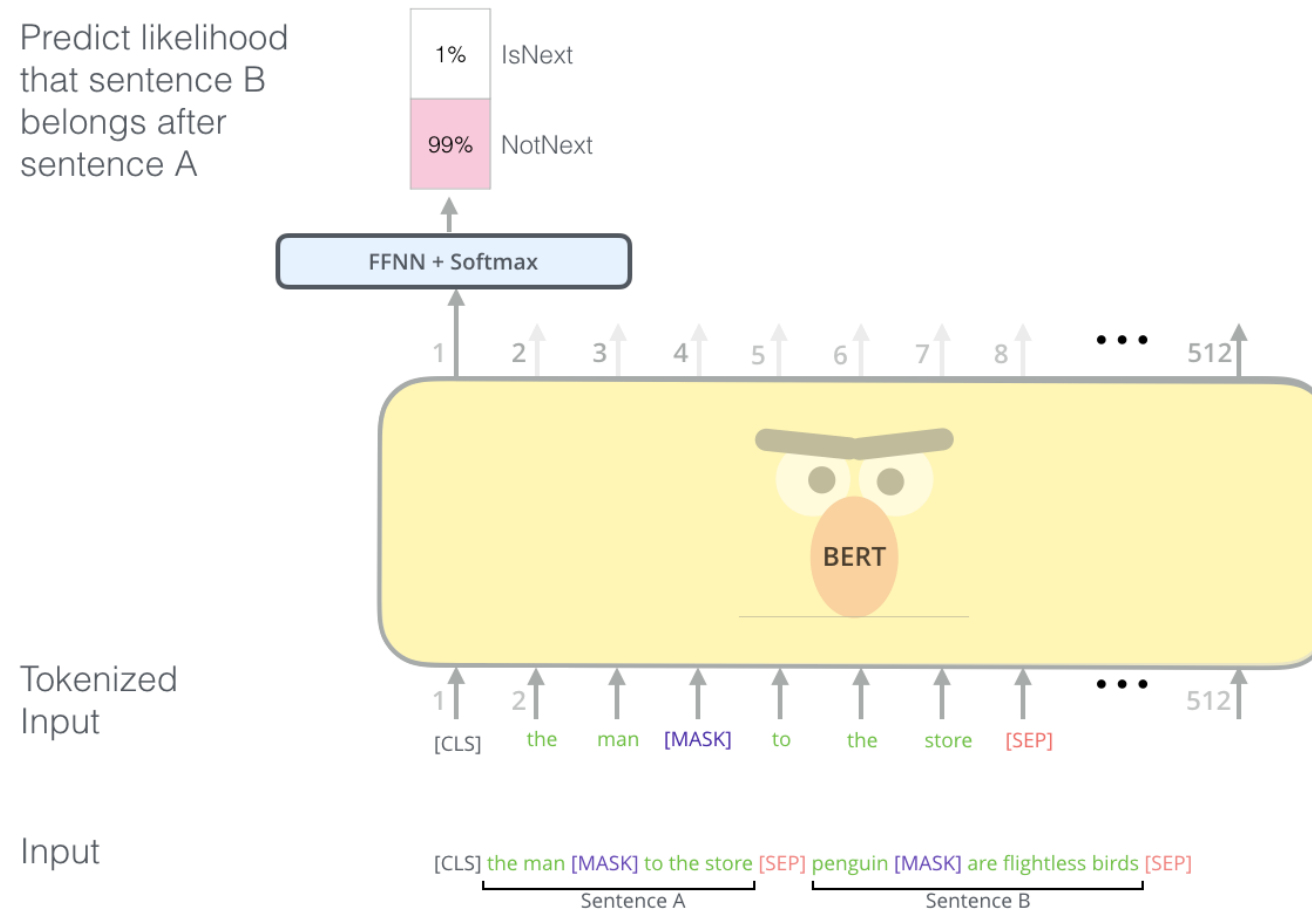
- Next sentence prediction
- To learn *relationships* between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

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

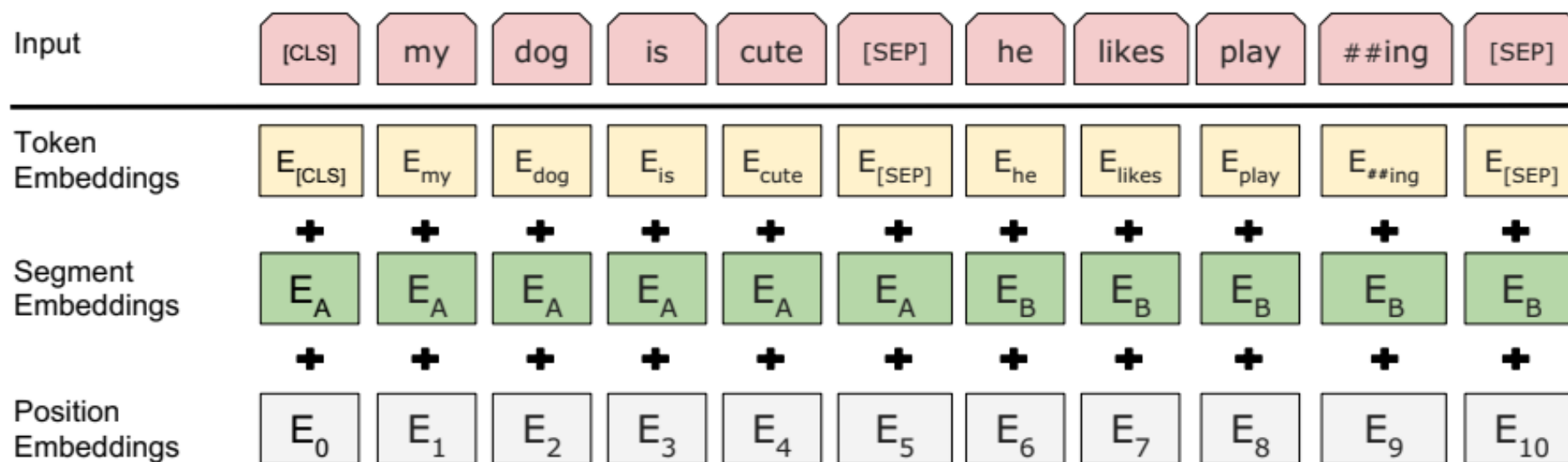
Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

BERT

- Next sentence prediction

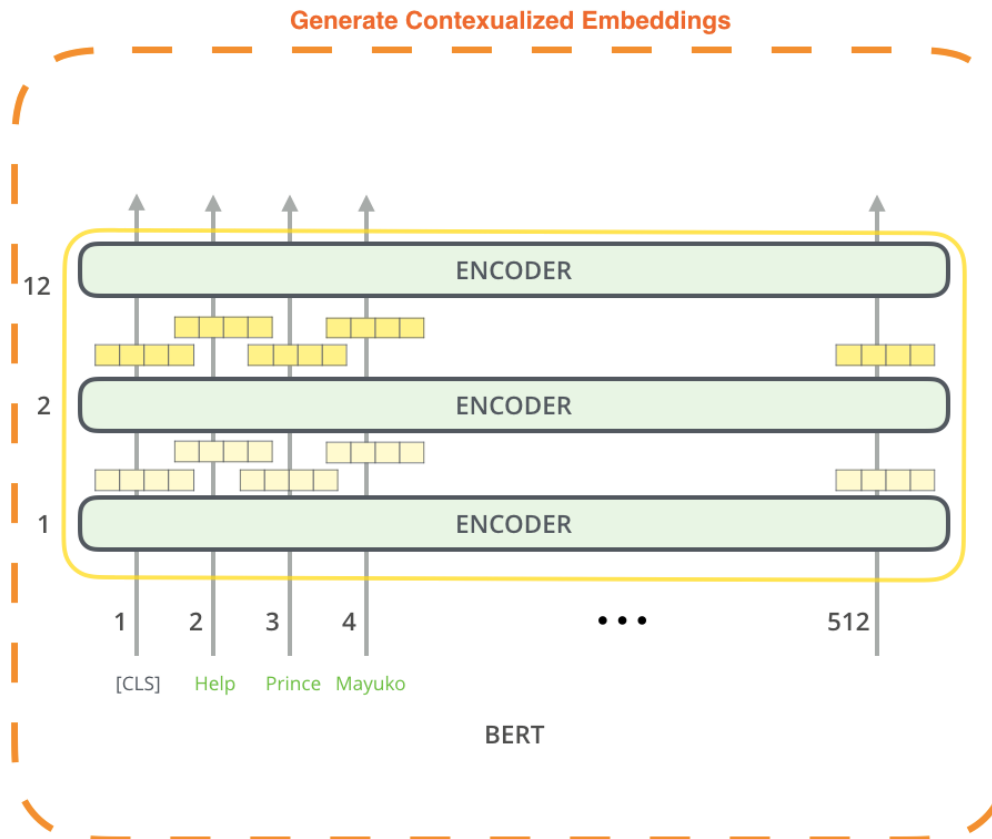


BERT

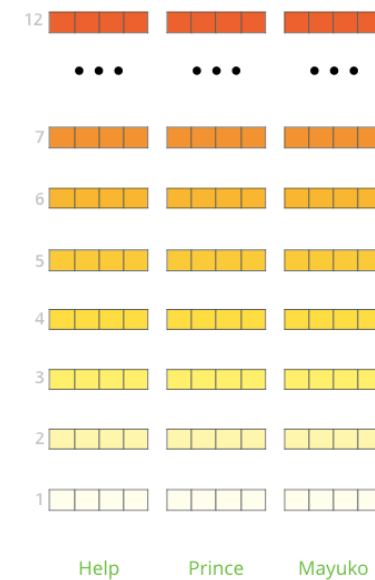


BERT

- Extracting embedding



The output of each encoder layer along each token's path can be used as a feature representing that token.



















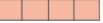


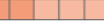



But which one should we use?

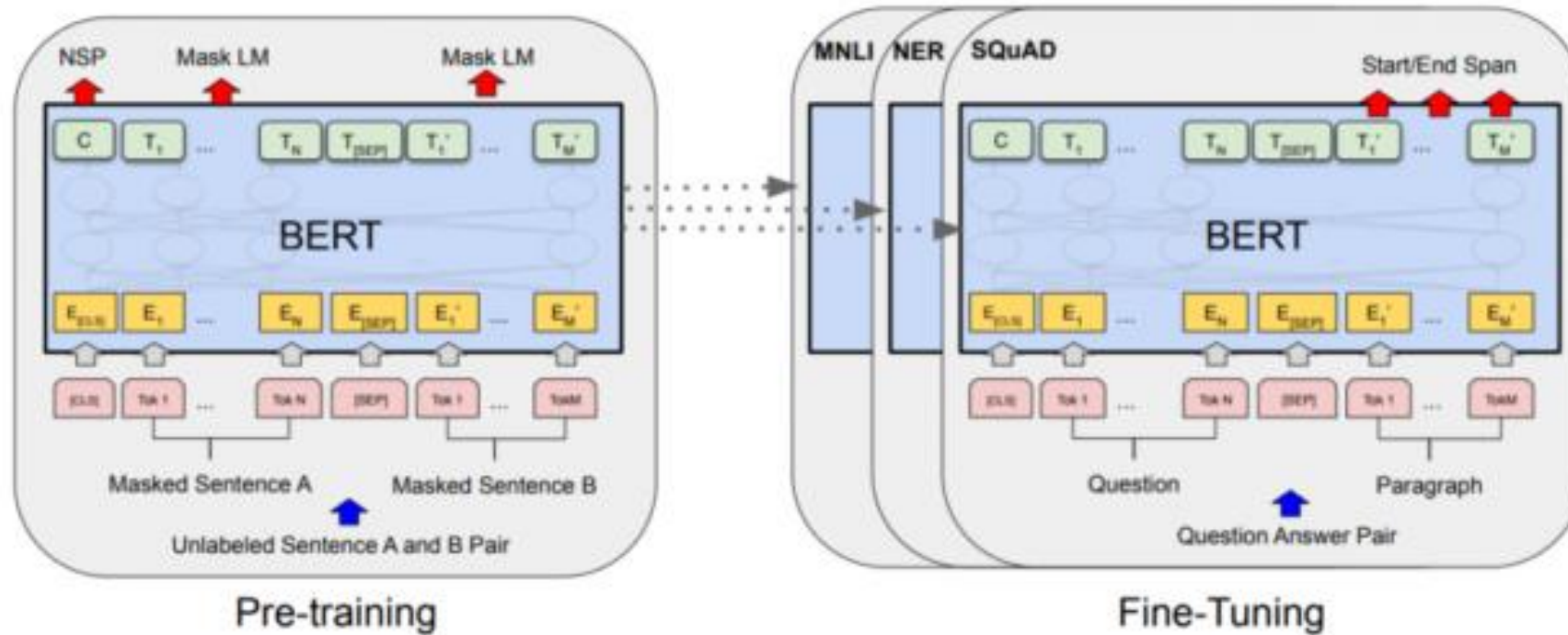
BERT

- Extracting embedding

What is the best contextualized embedding for “Help” in that context?
For named-entity recognition task CoNLL-2003 NER

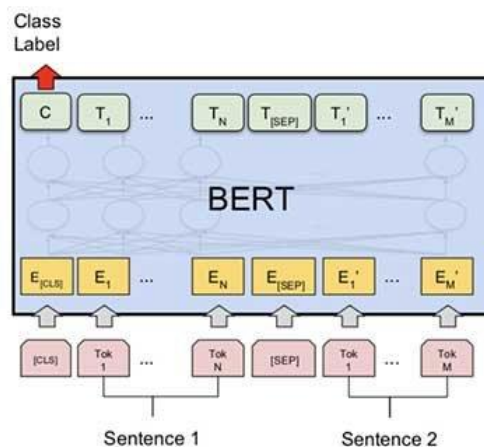
		Dev F1 Score
12 	First Layer Embedding 	91.0
• • •		
7 	Last Hidden Layer 12 	94.9
6 		
5 	Sum All 12 Layers	95.5
4 	12  + • • • + 2  + 1  =	
3 	Second-to-Last Hidden Layer 11 	95.6
2 		
1 	Sum Last Four Hidden	95.9
	12  + 11  + 10  + 9  =	
Help	Concat Last Four Hidden 9  10  11  12 	96.1

Fine-Tuning Procedure

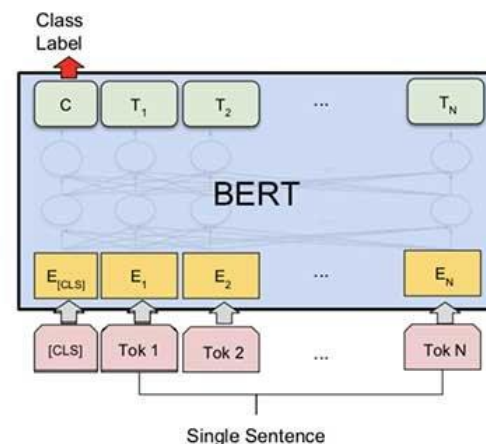


Fine-Tuning Procedure

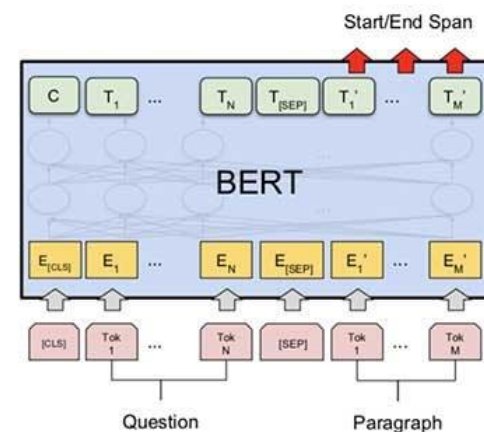
- Sentence pair classification
- Single sentence classification
- Question answering
- Single sentence sequence labeling



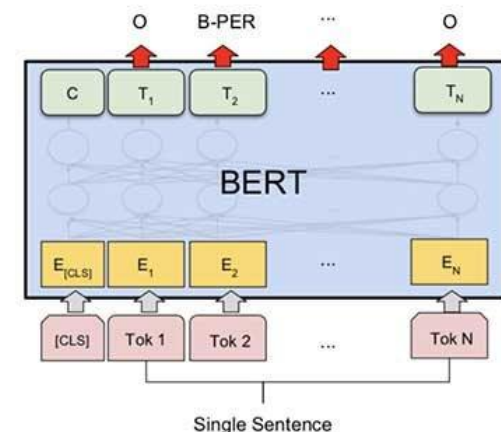
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



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



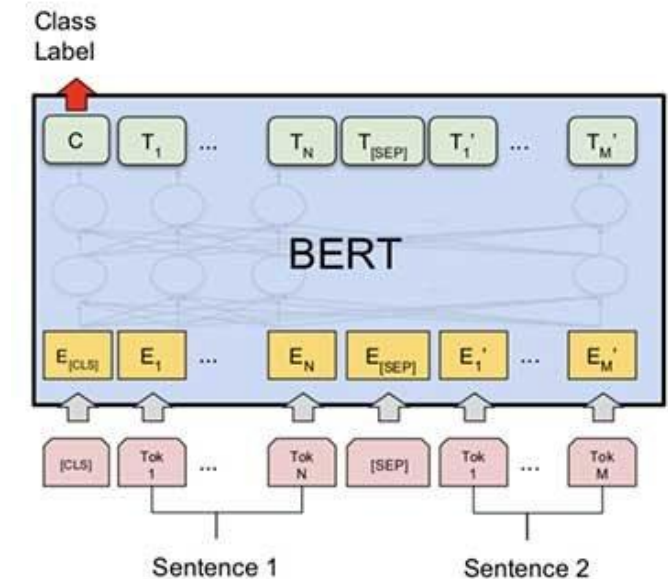
(c) Question Answering Tasks:
SQuAD v1.1



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

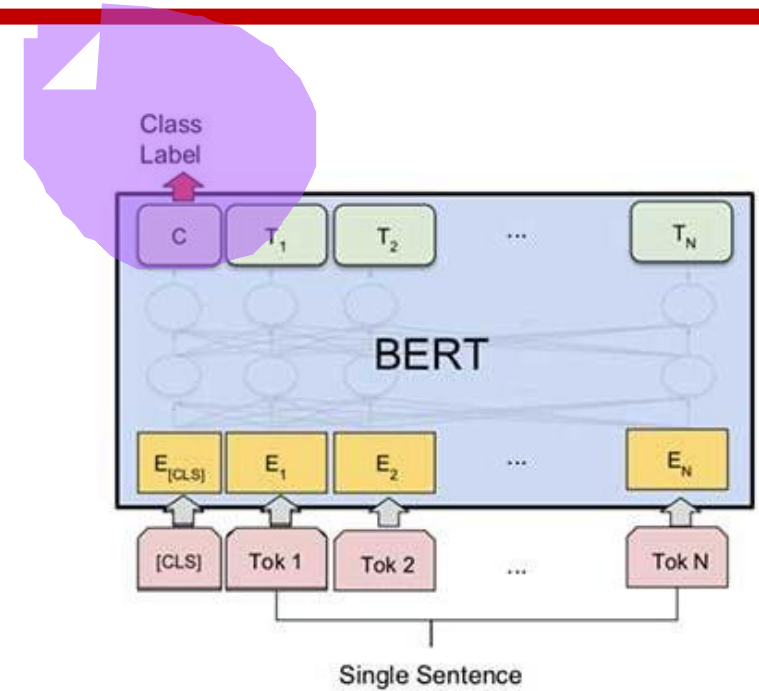
Fine-Tuning Procedure

- Sentence pair classification tasks
 - Paraphrase identification
 - Answer retrieval
 - Textual entailment
- Datasets:
 - MNLI
 - QQP
 - QNLI
 - STS-B
 - MRPC
 - RTE
 - SWAG



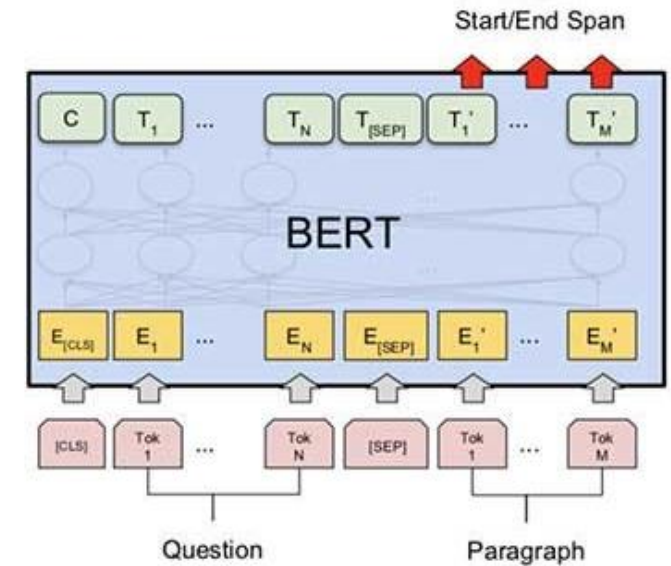
Fine-Tuning Procedure

- Single sentence classification tasks:
 - Spam detection
 - Sentiment analysis
 - News categorization
- Datasets:
 - SST-2
 - CoLA



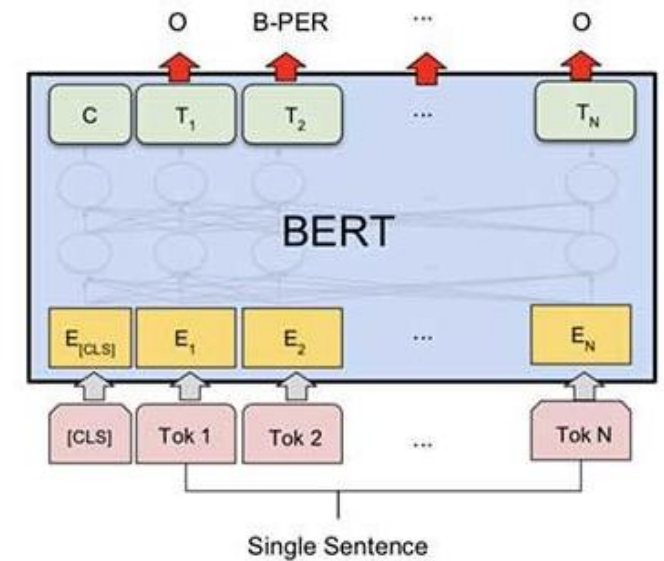
Fine-Tuning Procedure

- Question answering tasks
(finding answer span)
- Dataset:
 - SQuAD



Fine-Tuning Procedure

- Single sentence sequence labeling tasks:
 - NER
 - POS tagging
 - Slot filing
- Datasets:
 - CoNLL-2003 NER



Motivations for Improving BERT

- Accuracy
- Large
- Slow
- Hard to train

	BERT
Size (millions)	Base: 110 Large: 340
Training Time	Base: 8 x V100 x 12 days* Large: 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)
Performance	Outperforms state-of-the-art in Oct 2018
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)

Model Improvement Methods

- Larger with more data
- Quantization
 - Quantization-aware training
- Distillation
- Pruning
- Specialization

Further Reading

- Speech and Language Processing (3rd ed. draft)
 - Chapter 11