BERT (Part 1)

CS 685, Spring 2021

Advanced Topics in Natural Language Processing

http://brenocon.com/cs685

https://people.cs.umass.edu/~brenocon/cs685_s21/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

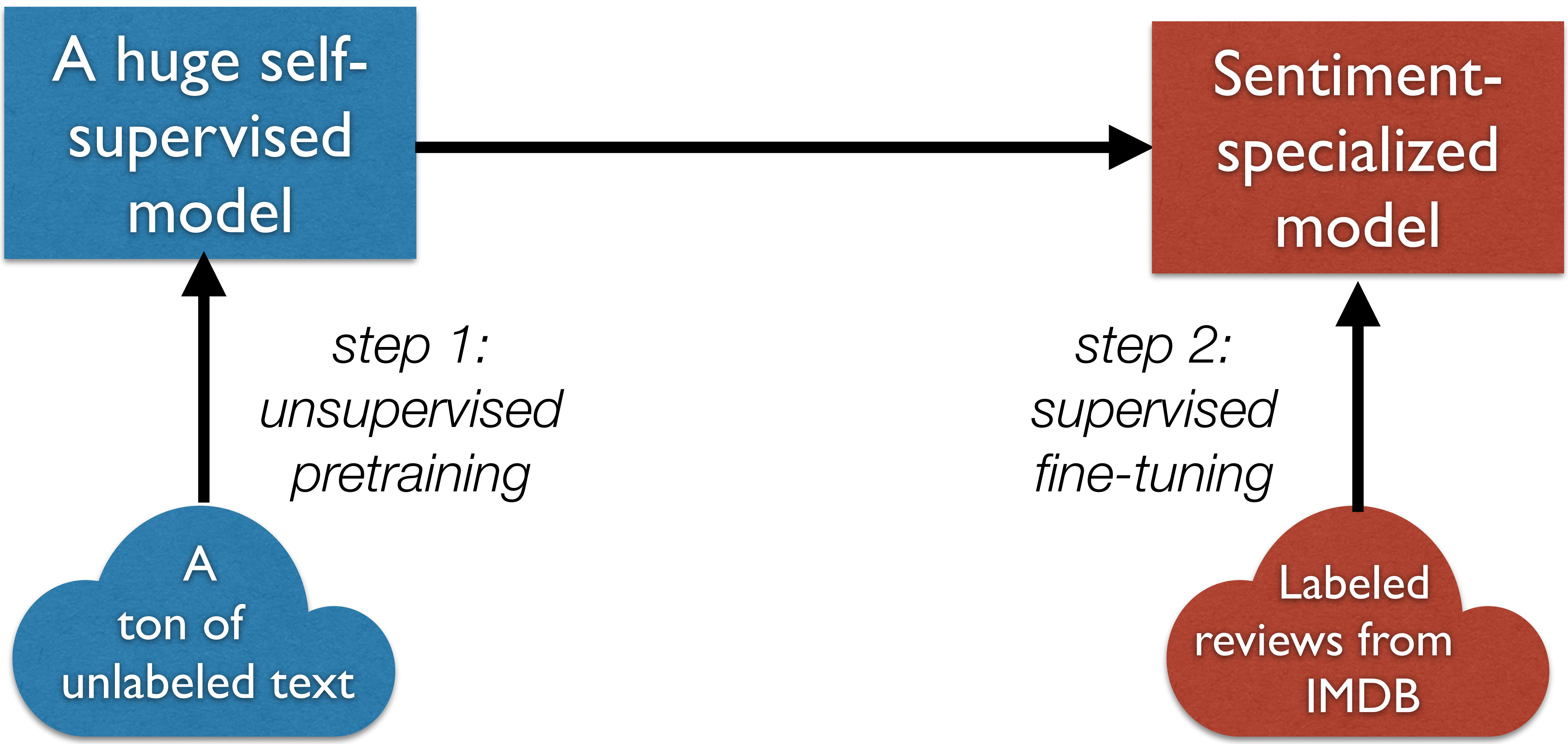
[Nearly all slides from Jacob Devlin]

- My OH is tomorrow, 9:30-10:30 (now on webpage)
- For Wed: reading review #3: pick a BERTology paper

- Fri Mar 26: Lit review due
- Fri Apr 2: Project proposals due
  - will talk about on Wed

What is transfer learning?

- **In our context:** take a network trained on a task for which it is easy to generate labels, and adapt it to a different task for which it is harder.
    - **In computer vision:** train a CNN on ImageNet, transfer its representations to every other CV task
    - **In NLP:** train a really big language model on billions of words, transfer to every NLP task!


A huge self-supervised model
Sentiment-specialized model
step 1: unsupervised pretraining
step 2: supervised fine-tuning
A ton of unlabeled text
Labeled reviews from IMDB

FROM

TO

- Context in 2018: ELMo demonstrated
  - unsupervised transfer
  - context-dependent, token-level feature extraction
  - with LSTM-based bidirectional LM
- Then there was BERT. Same thing but
  - fine-tuning, not just feature extraction
    [Follow-up work: but does this matter?]
  - with Transformer-based masked “LM”
  - and lots of layers
    [Follow-up work: what do they learn?]
  - *[More follow-ups: do we need so many attention heads?]*
  - *[More follow-ups: how much training variation is there?]*
- Today: BERT, or a close variant, is the best!
  But we don’t know why.
  - Will we still be using BERT in X years?

Problem with Previous Methods

- **Problem**: Language models only use left context *or* right context, but language understanding is bidirectional.
- Why are LMs unidirectional?

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.

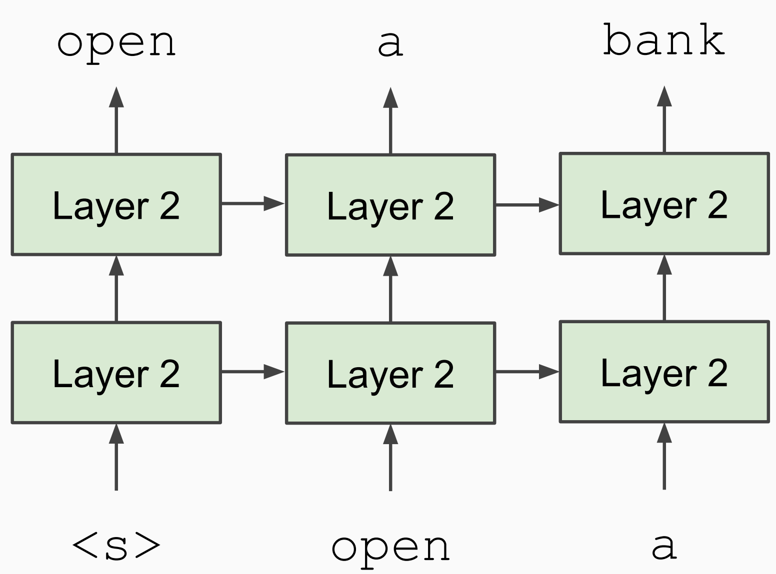
Why not?

Problem with Previous Methods

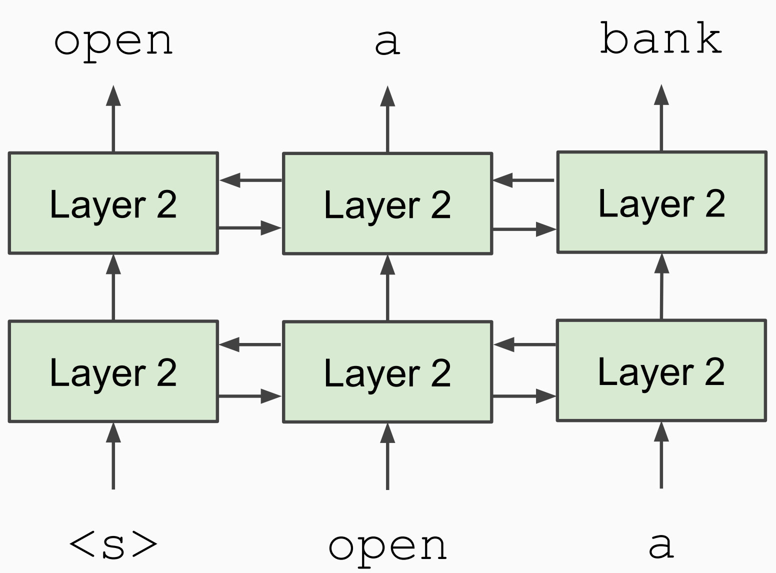
- **Problem**: Language models only use left context *or* right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
- <u>Reason 1</u>: Directionality is needed to generate a well-formed probability distribution.
    - We don't care about this.
- <u>Reason 2</u>: Words 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
  - We always use k = 15%

```
                     store                gallon
                       ↑                     ↑
the man went to the [MASK] to buy a [MASK] of milk
```

What are the pros and cons of increasing *k*?

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`

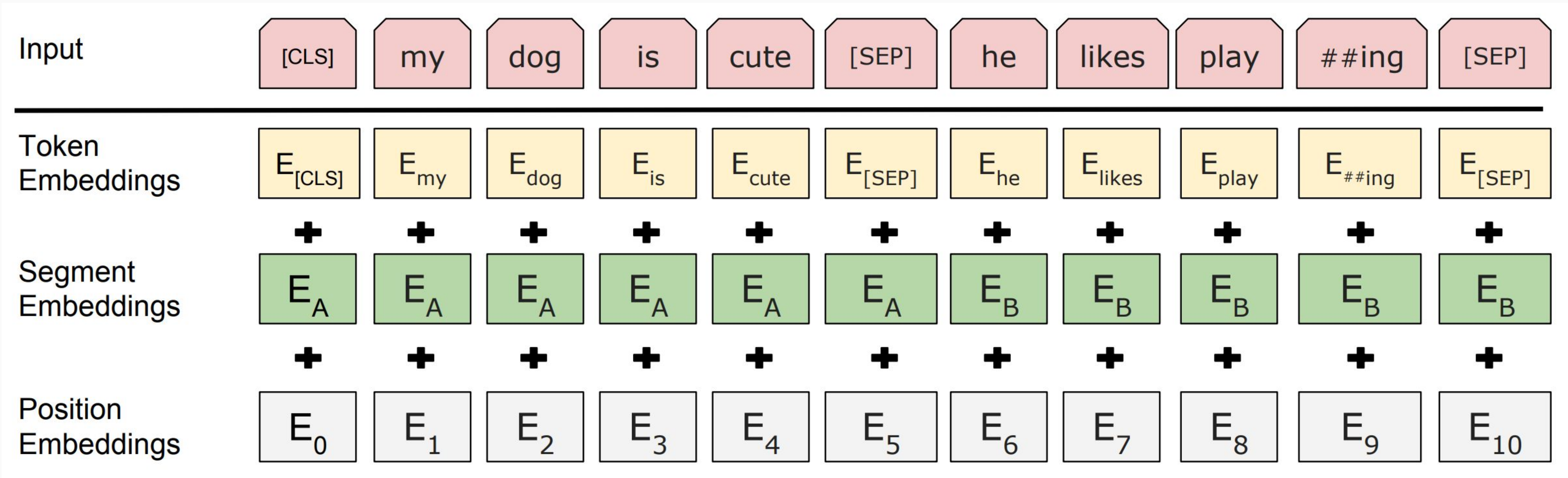
Next Sentence Prediction

- 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`

NOTE: Follow-up work, e.g. RoBERTa (Liu et al. 2019) has cast doubt on whether this part is necessary.

Input Representation

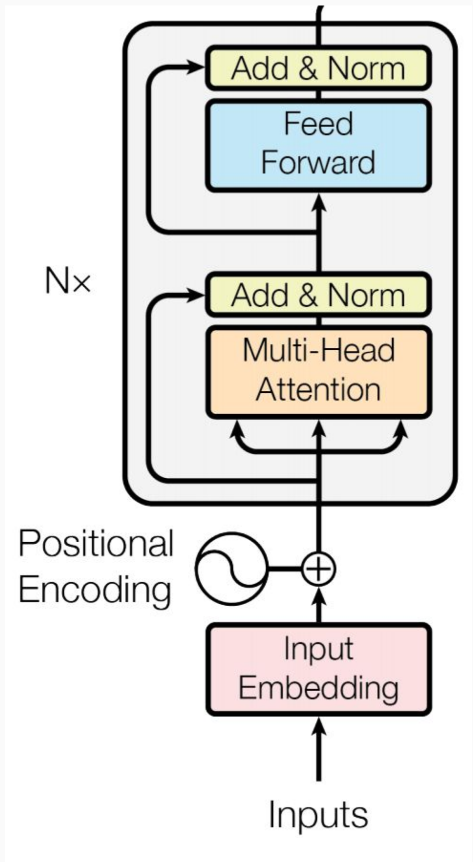


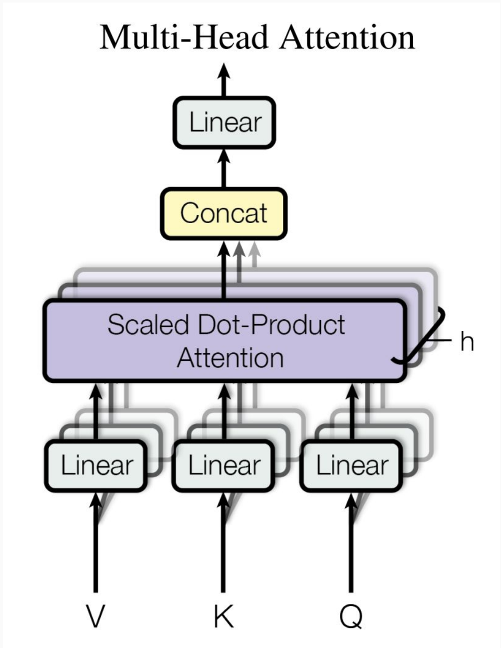
- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings
- Single sequence is much more efficient.

Model Architecture

Transformer encoder

- 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



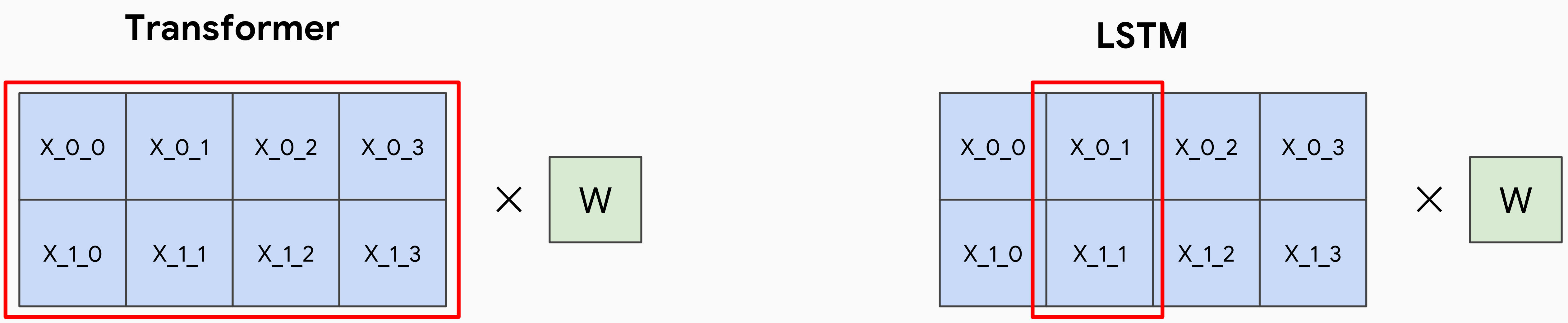


My favorite notation so far (added to webpage from last week):
https://namedtensor.github.io/#sec:transformer

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*

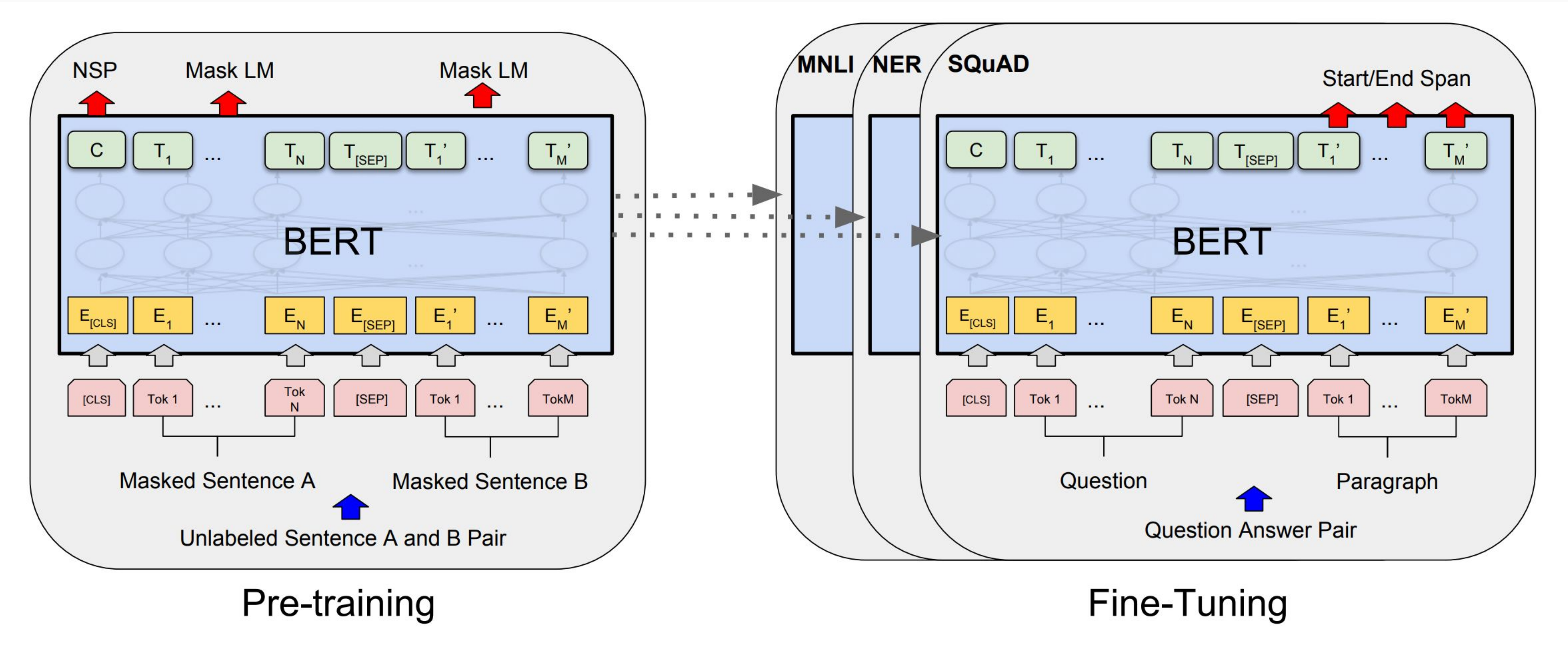


TPU hardware = Google’s variant of GPU

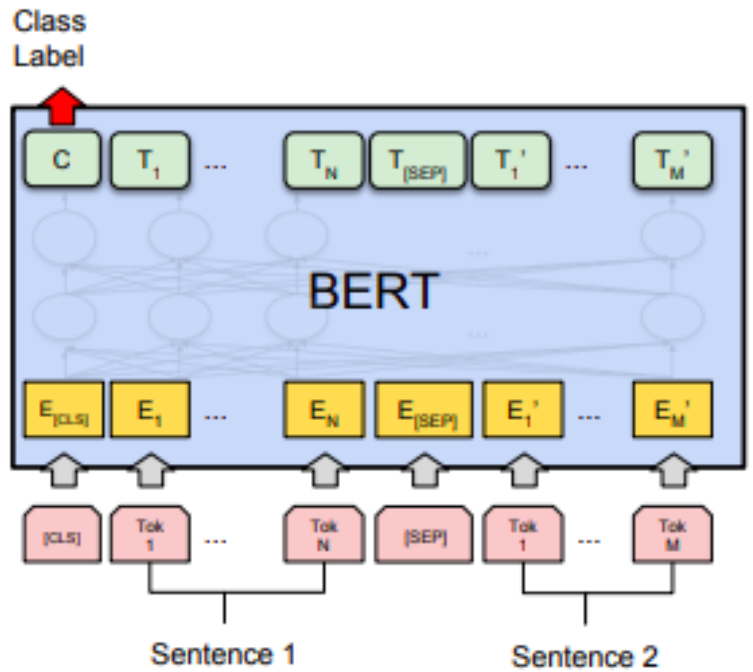
Model Details

- 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

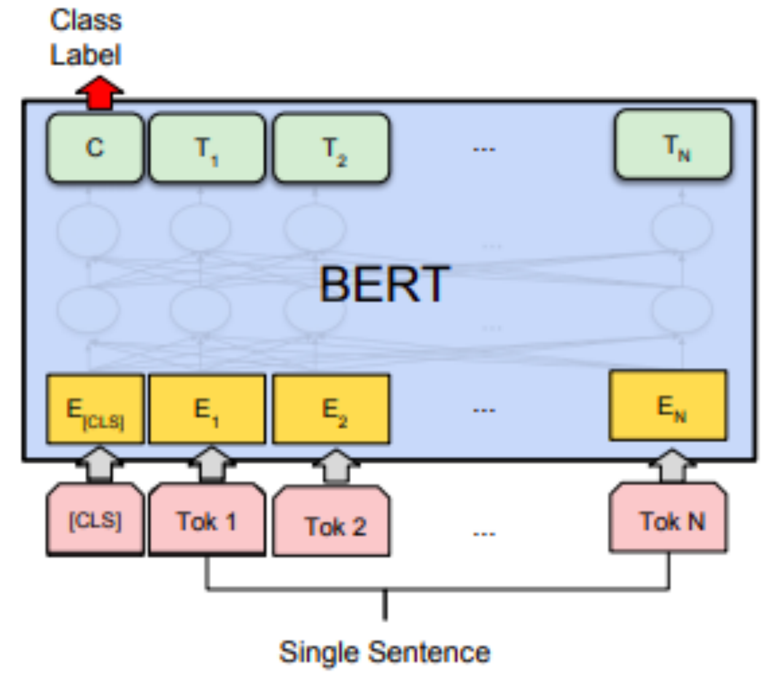
Fine-Tuning Procedure



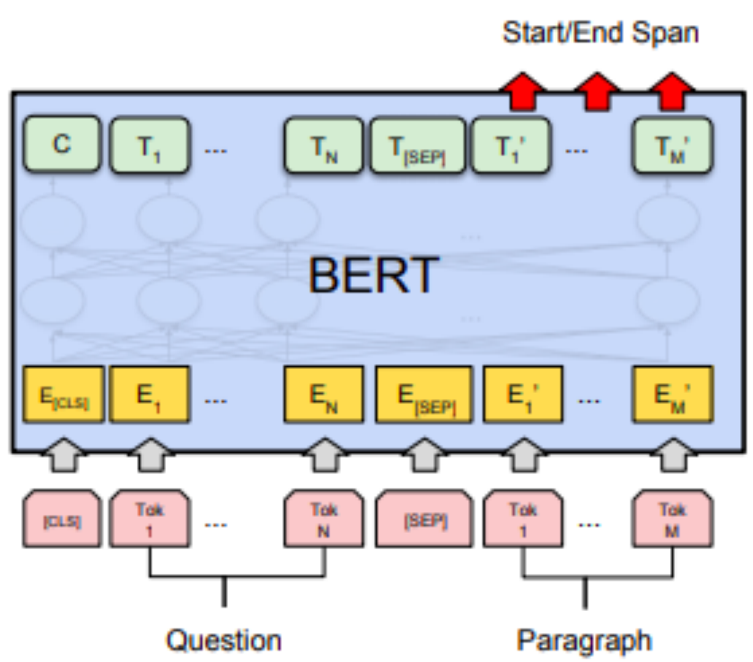
Fine-Tuning Procedure



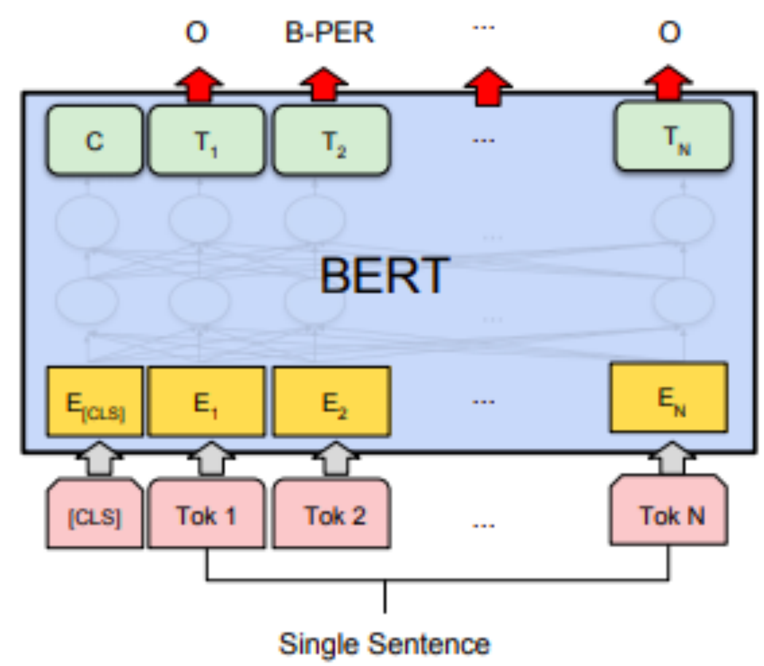
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



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



(c) Question Answering Tasks:
SQuAD v1.1



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

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

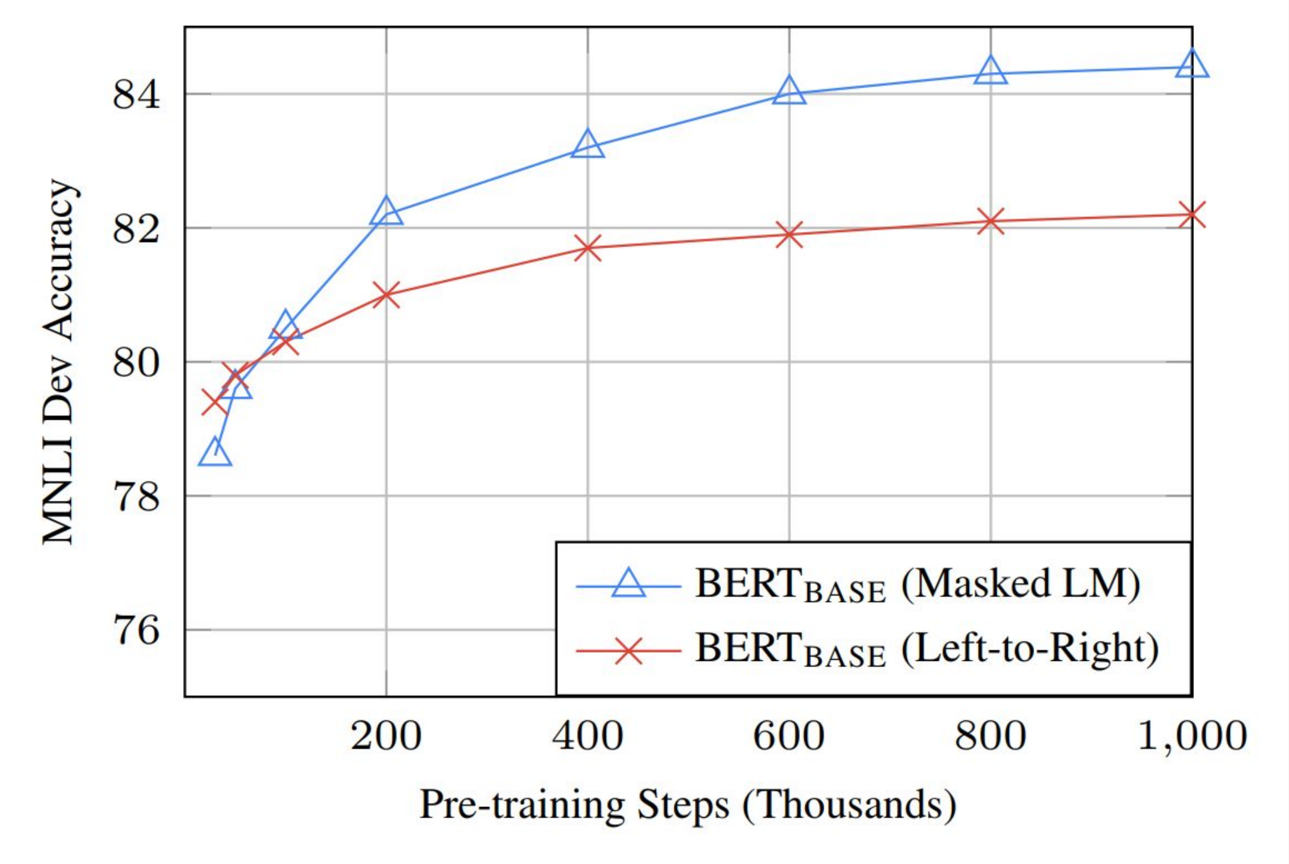
CoLa

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

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

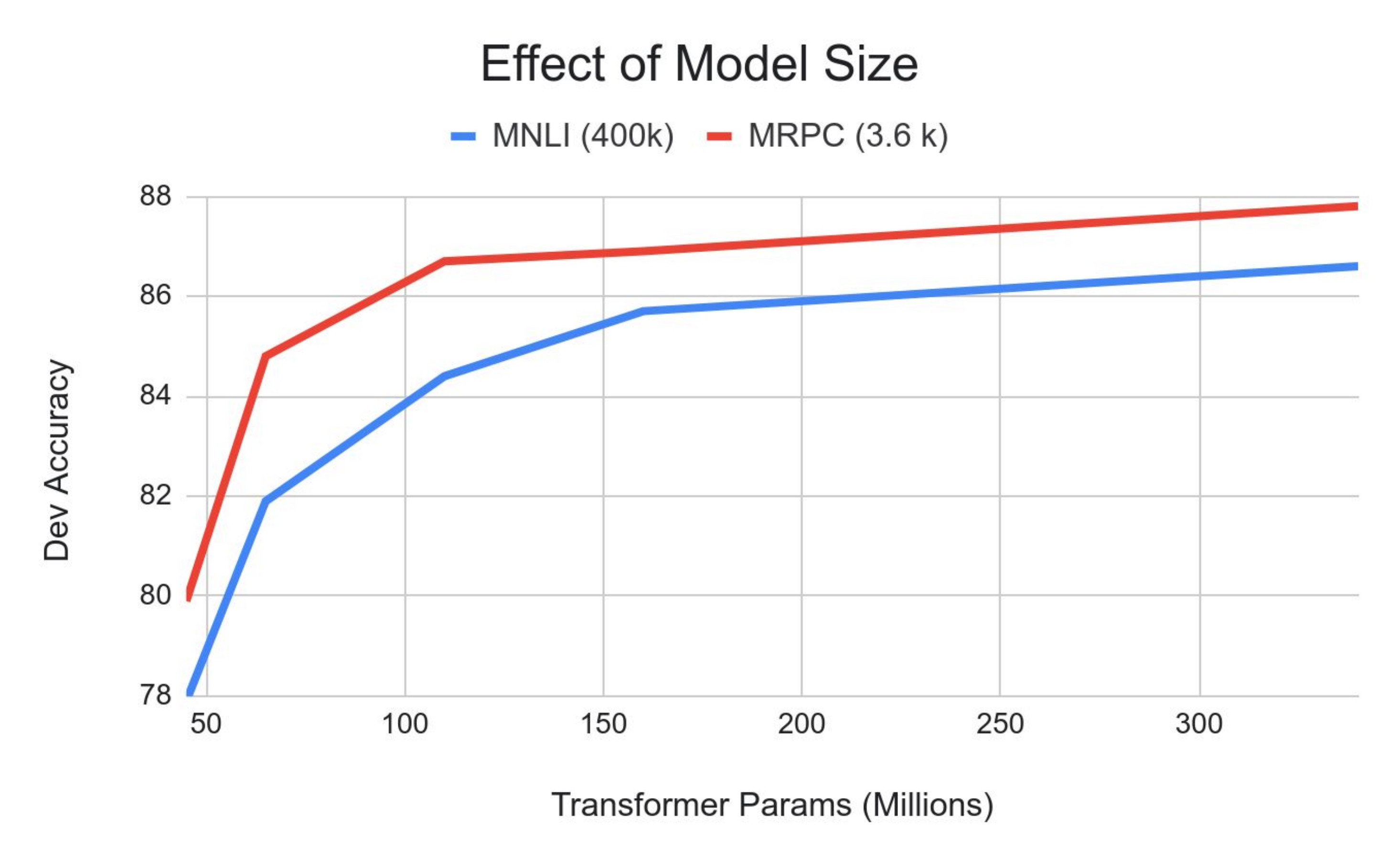
Presentation/summary tip:
Always show examples of he task!

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

Features vs. fine-tuning

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

Effect of Masking Strategy

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI Fine-tune	NER Fine-tune	NER Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

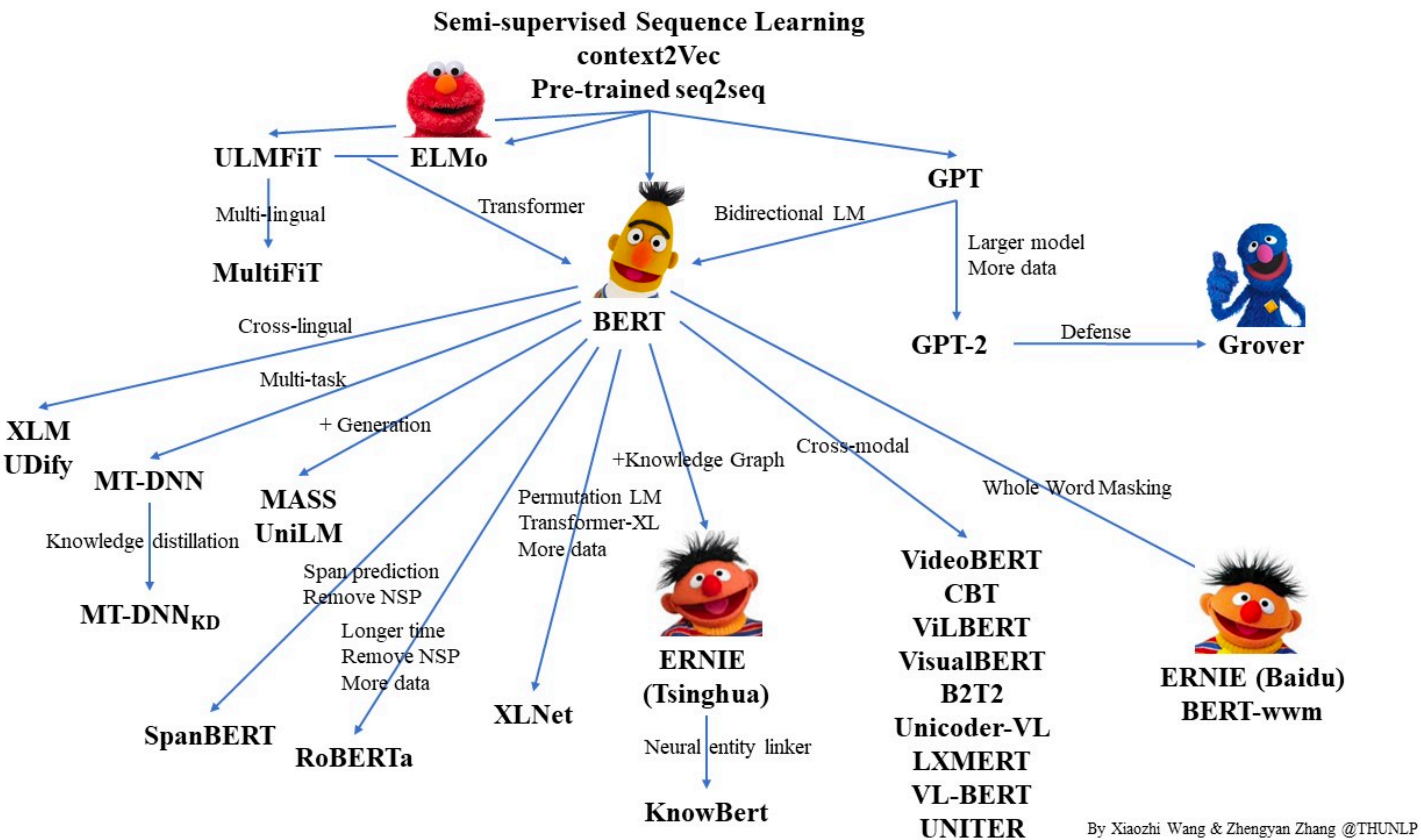
- Masking 100% of the time hurts on feature-based approach
- Using random word 100% of time hurts slightly

Multilingual BERT

- Trained single model on 104 languages from Wikipedia. Shared 110k WordPiece vocabulary.

System	English	Chinese	Spanish
XNLI Baseline - Translate Train	73.7	67.0	68.8
XNLI Baseline - Translate Test	73.7	68.4	70.7
BERT - Translate Train	81.9	76.6	77.8
BERT - Translate Test	81.9	70.1	74.9
BERT - Zero Shot	81.9	63.8	74.3

- XNLI is MultiNLI translated into multiple languages.
- Always evaluate on human-translated Test.
- Translate Train: MT English Train into Foreign, then fine-tune.
- Translate Test: MT Foreign Test into English, use English model.
- Zero Shot: Use Foreign test on English model.


Semi-supervised Sequence Learning
context2Vec
Pre-trained seq2seq
ULMFiT
ELMo
GPT
Transformer
Bidirectional LM
Multi-lingual
MultiFiT
Larger model
More data
BERT
GPT-2
Defense
Grover
Cross-lingual
Multi-task
XLM
UDify
+ Generation
MT-DNN
MASS
UniLM
Knowledge distillation
MT-DNN$_{KD}$
Span prediction
Remove NSP
Longer time
Remove NSP
More data
SpanBERT
RoBERTa
Permutation LM
Transformer-XL
More data
XLNet
+Knowledge Graph
ERNIE
(Tsinghua)
Neural entity linker
KnowBert
Cross-modal
VideoBERT
CBT
ViLBERT
VisualBERT
B2T2
Unicoder-VL
LXMERT
VL-BERT
UNITER
Whole Word Masking
ERNIE (Baidu)
BERT-wwm
By Xiaozhi Wang & Zhengyan Zhang @THUNLP

Common Questions

- Is *deep* bidirectionality really necessary? What about ELMo-style shallow bidirectionality on bigger model?
- Advantage: Slightly faster training time
- Disadvantages:
    - Will need to add non-pre-trained bidirectional model on top
    - Right-to-left SQuAD model doesn't see question
    - Need to train two models
    - Off-by-one: LTR predicts next word, RTL predicts previous word
    - Not trivial to add arbitrary pre-training tasks.

Common Questions

- The model must be learning more than “contextual embeddings”
- Alternate interpretation: Predicting missing words (or next words) requires learning many types of language understanding features.
    - syntax, semantics, pragmatics, coreference, etc.
- Implication: Pre-trained model is much bigger than it needs to be to solve specific task
- Task-specific model distillation words very well

Common Questions

- Is modeling “solved” in NLP? I.e., is there a reason to come up with novel model architectures?
    - But that’s the most fun part of NLP research :(
- Maybe yes, for now, on some tasks, like SQuAD-style QA.
    - At least using the same deep learning “lego blocks”
- Examples of NLP models that are not “solved”:
    - Models that minimize total training cost vs. accuracy on modern hardware
    - Models that are very parameter efficient (e.g., for mobile deployment)
    - Models that represent knowledge/context in latent space
    - Models that represent structured data (e.g., knowledge graph)
    - Models that jointly represent vision and language

Conclusions

- Empirical results from BERT are great, but biggest impact on the field is:
- With pre-training, bigger == better, without clear limits (so far).
- Unclear if adding things on top of BERT really helps by very much.
  - Good for people and companies building NLP systems.
  - Not necessary a “good thing” for researchers, but important.

- Context in 2018: ELMo demonstrated
    - unsupervised transfer
    - context-dependent, token-level feature extraction
    - with LSTM-based bidirectional LM
- Then there was BERT. Same thing but
    - fine-tuning, not just feature extraction
      [Follow-up work: but does this matter?]
    - with Transformer-based masked "LM"
    - and lots of layers
      [Follow-up work: what do they learn?]
    - *[More follow-ups: do we need so many attention heads?]*
    - *[More follow-ups: how much training variation is there?]*
- Today: BERT, or a close variant, is the best!
  But we don't know why.
    - Will we still be using BERT in X years?