BERT (and other encoder models)

THE UNIVERSITY OF UTAH

Slides credit: Chris Tanner, Jacob Devlin, Daniel Kashabi, Jay Alammar, and others

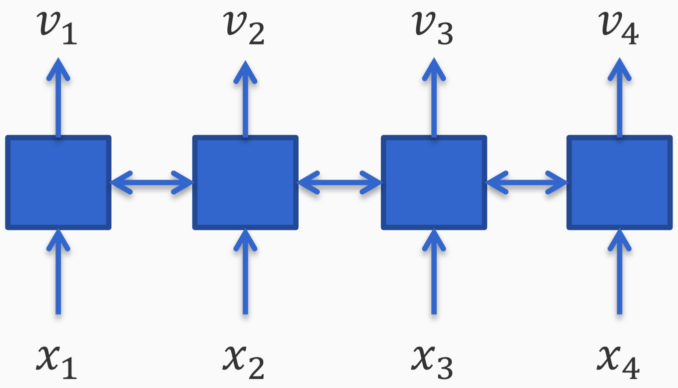
Outline

- Transformers: Recap
- What is BERT?
- Pre-training and fine-tuning
- The impact of BERT
- What does BERT “know”?

Outline

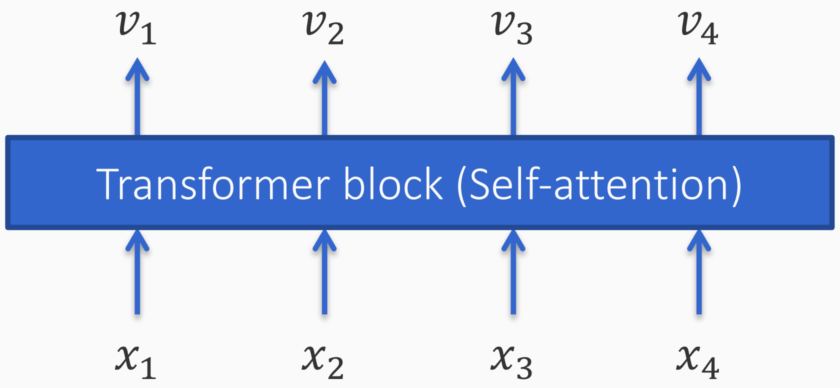
- Transformers: Recap
- What is BERT?
- Pre-training and fine-tuning
- The impact of BERT
- What does BERT “know”?

Recurrent networks → Self-attention



(bi)RNNs encode sequences

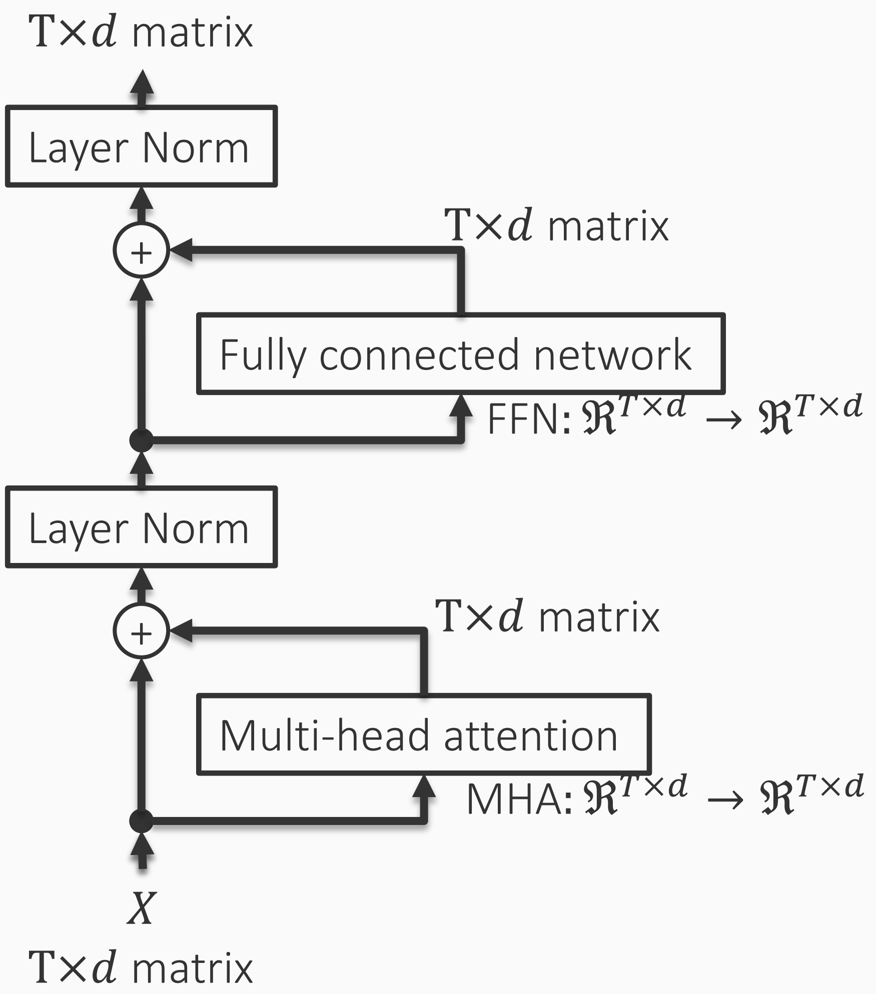
ELMo: Use stacks of RNNs to train contextualized word embeddings



Transformers also encode sequences, and they are easily parallelizable

Can we use them for training word embeddings?

Recall: A transformer

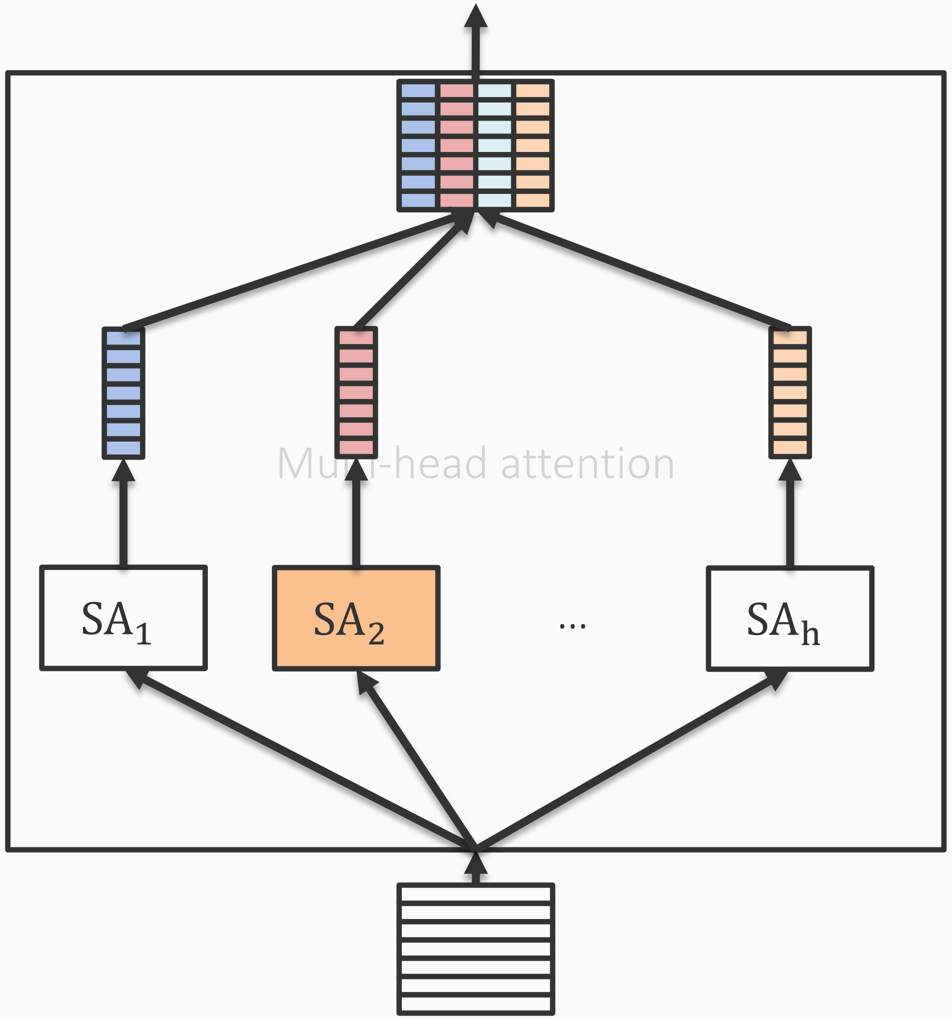



```python
def transformer_layer(X):
    X1 = layer_norm1(X + multi_head_attention(X))
    X2 = layer_norm2(X1 + fully_connected(X1))
    return X2
```

$$X_1 = \text{LayerNorm}\big(X + MHA(X)\big)$$
$$\text{Result} = \text{LayerNorm}\big(X_1 + FFN(X_1)\big)$$

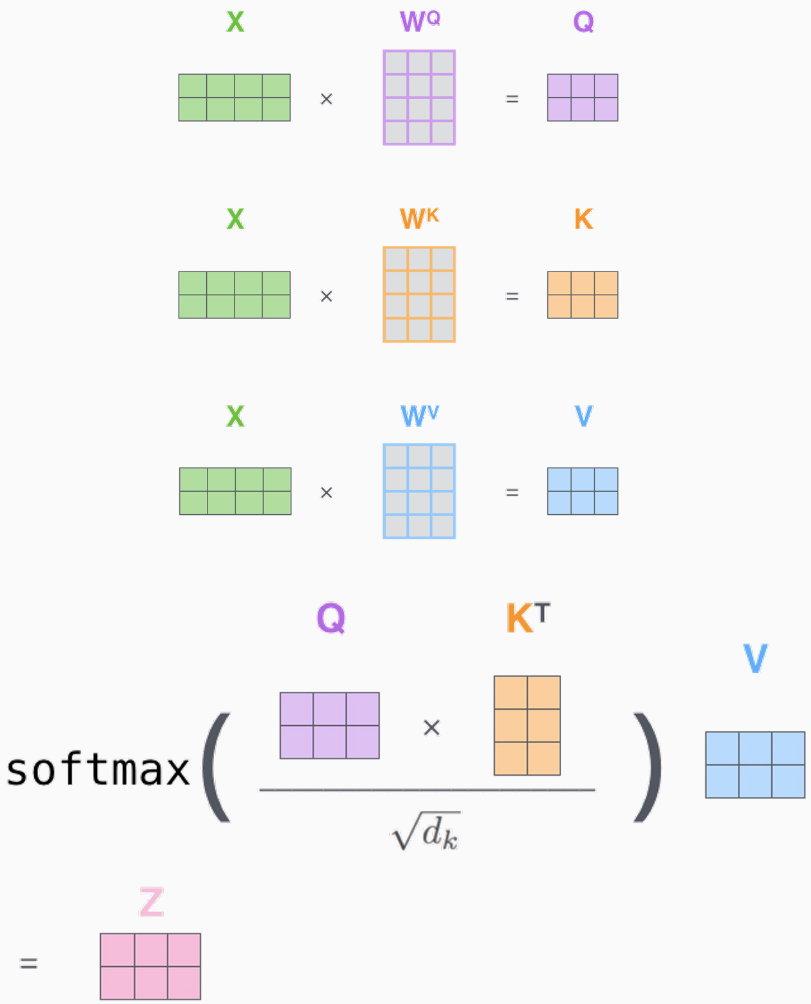
Recall: Multi-head attention

T×d output matrix



T×d input matrix

Recall: Self-attention

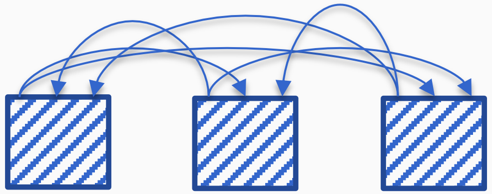


Given input $\mathbf{x}$:

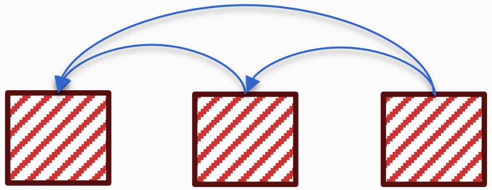
$$
\begin{aligned}
Q &= \mathbf{W}^q \mathbf{x} \\
K &= \mathbf{W}^k \mathbf{x} \\
V &= \mathbf{W}^v \mathbf{x}
\end{aligned}
$$

$$\mathrm{Attention}(\mathbf{x}) = \mathrm{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V$$

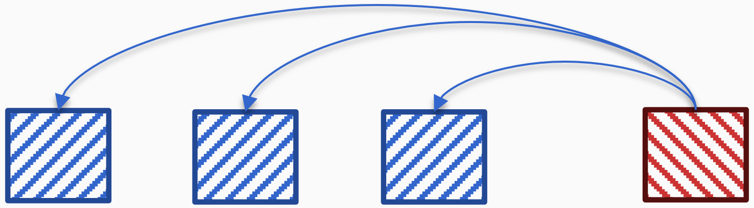
Three different kinds of attention



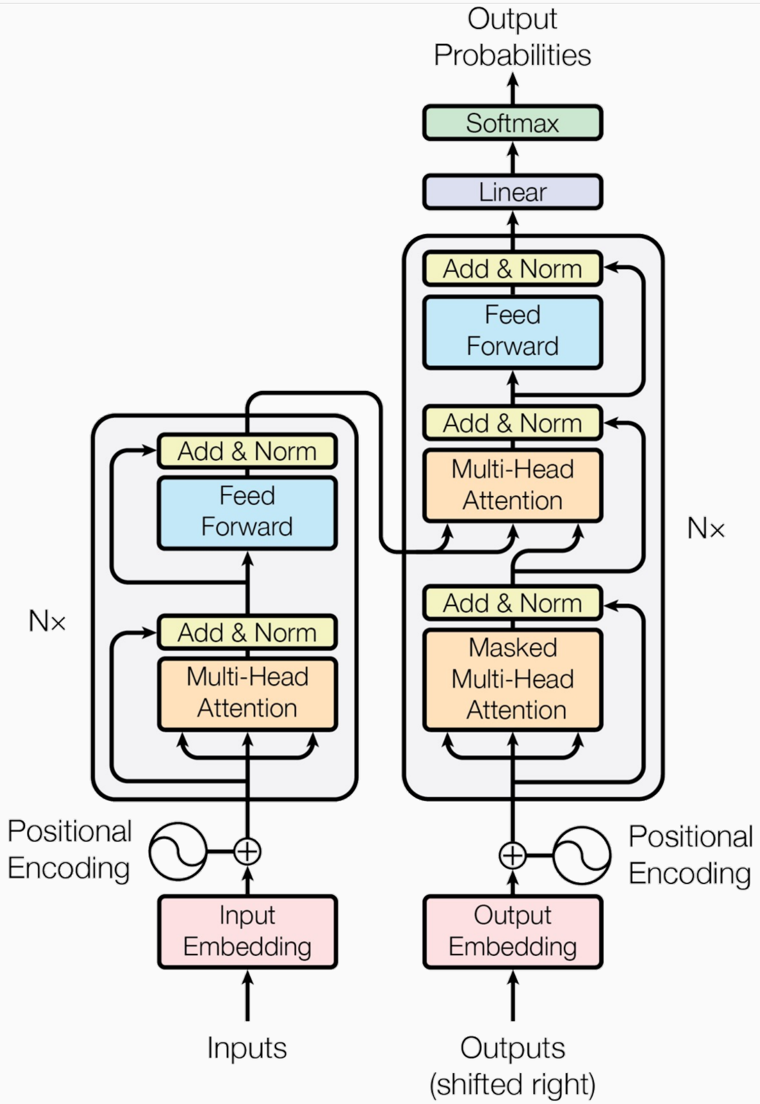
Encoder self-attention



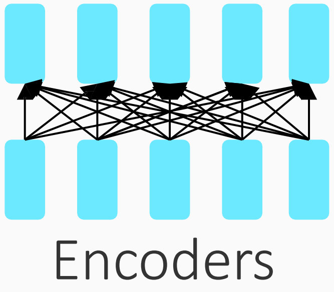
Masked decoder self-attention



Encoder-decoder self-attention



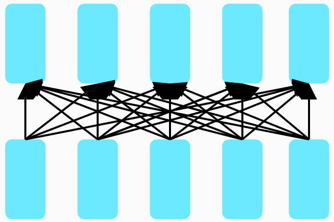
Transformers are the default building blocks for NLP today



Examples: BERT, RoBERTa, SciBERT.

Captures bidirectional context

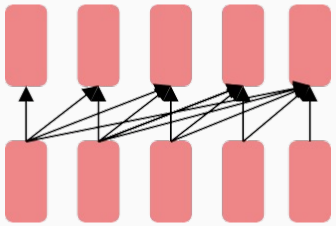
Transformers are the default building blocks for NLP today



Encoders

Examples: BERT, RoBERTa, SciBERT.

Captures bidirectional context



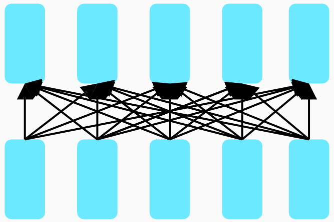
Decoders

Examples: GPT-2, GPT-3, LaMDA

Also known as: causal or auto-regressive language model

Natural if the goal is generation, but can not condition on future words

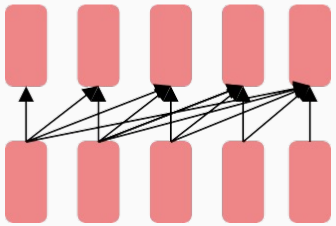
Transformers are the default building blocks for NLP today



Encoders

Examples: BERT, RoBERTa, SciBERT.

Captures bidirectional context

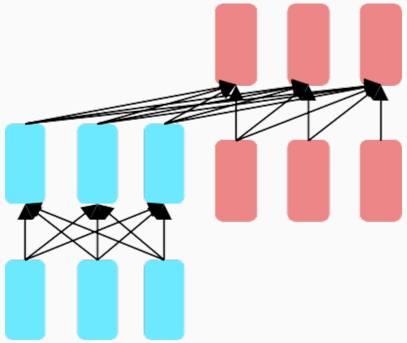


Decoders

Examples: GPT-2, GPT-3, LaMDA

Also known as: causal or auto-regressive language model

Natural if the goal is generation, but can not condition on future words

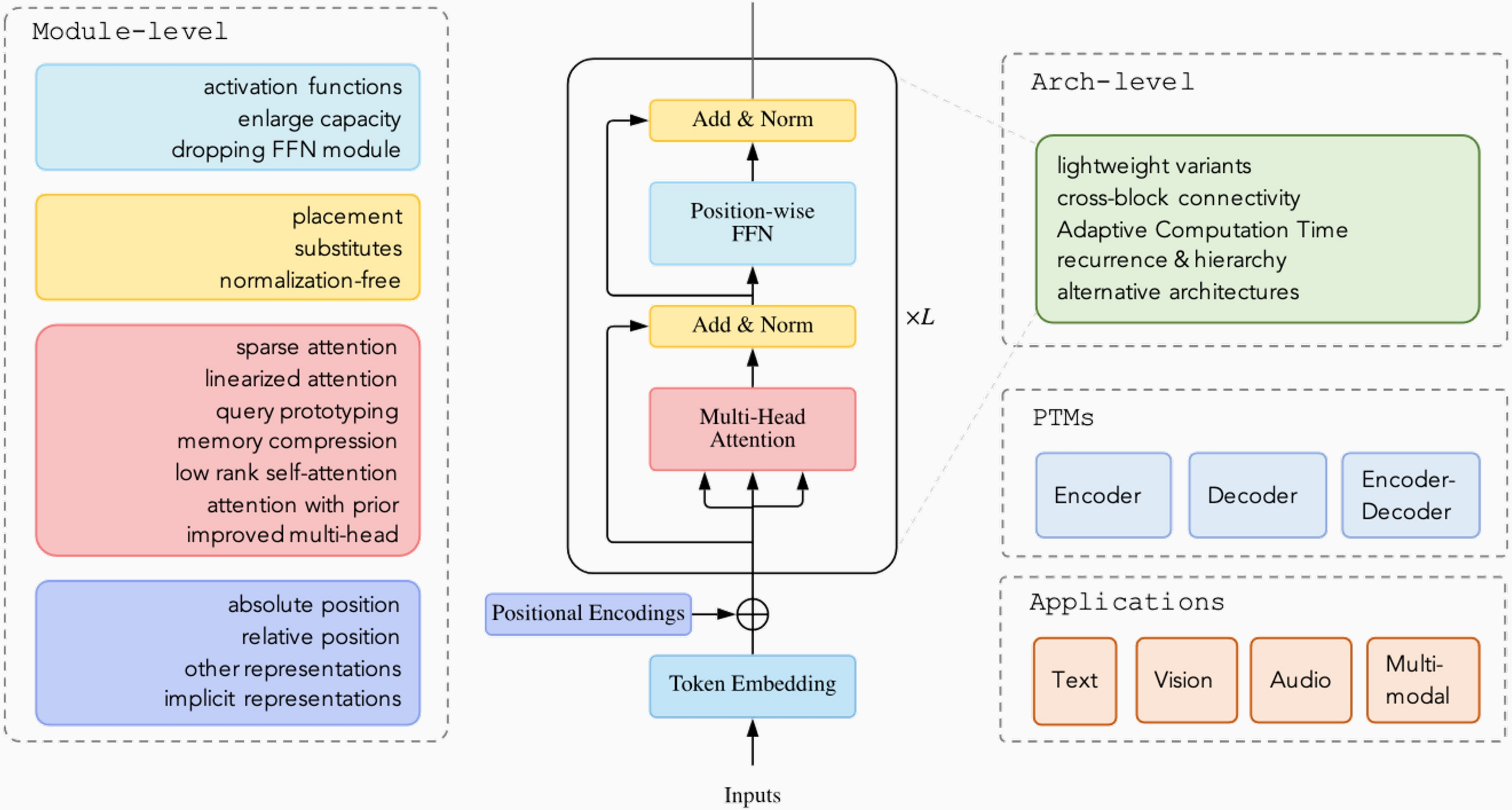


Encoder-Decoders

Examples: BART, T5, Meena

Conditional generation based on an encoded input

Each component of the transformer is a design choice



Lin, Tianyang, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. "A survey of transformers." *AI Open* (2022).

Outline

- Transformers: Recap
- What is BERT?
- Pre-training and fine-tuning
- The impact of BERT
- What does BERT “know”?

2018

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

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

BERT

Bidirectional Encoder Representations from Transformers

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

BERT

Bidirectional Encoder Representations from Transformers

Bi-directional context to get word representation

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

BERT

Only encoder transformer

Bidirectional Encoder Representations from Transformers

Bi-directional context to get word representation

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

BERT

Only encoder transformer

Bidirectional Encoder Representations from Transformers

Bi-directional context to get word representation

Train word representations using self-supervised learning

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

What is BERT?

A contextualized embedding like ELMo

- Word representations are computed based on the context in which they appear

What is BERT?

A contextualized embedding like ELMo

- Word representations are computed based on the context in which they appear

Differences from ELMo

- An encoder transformer-based model

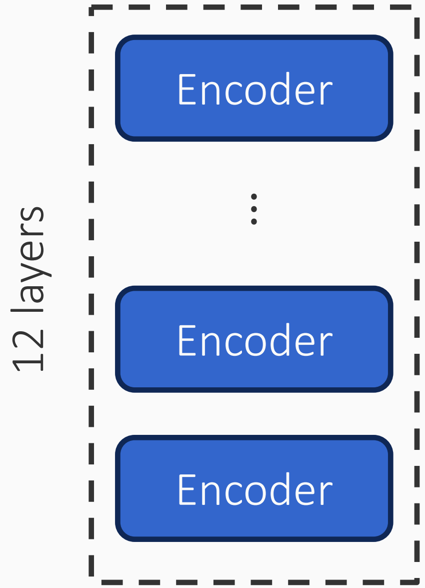
  Self-attention instead of recurrence to capture context

- Much larger data for training

  1B Word Benchmark (Chelba et al., 2014) → BooksCorpus (800M words) + English Wikipedia (2500M words)

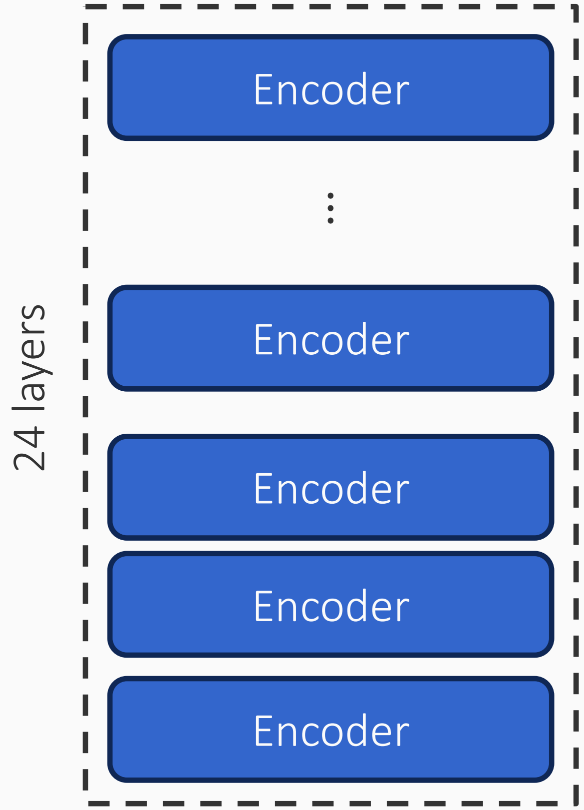
- New objectives for training
  - Masked language modeling objective
  - Next sentence prediction

BERT: A stack of transformer encoders



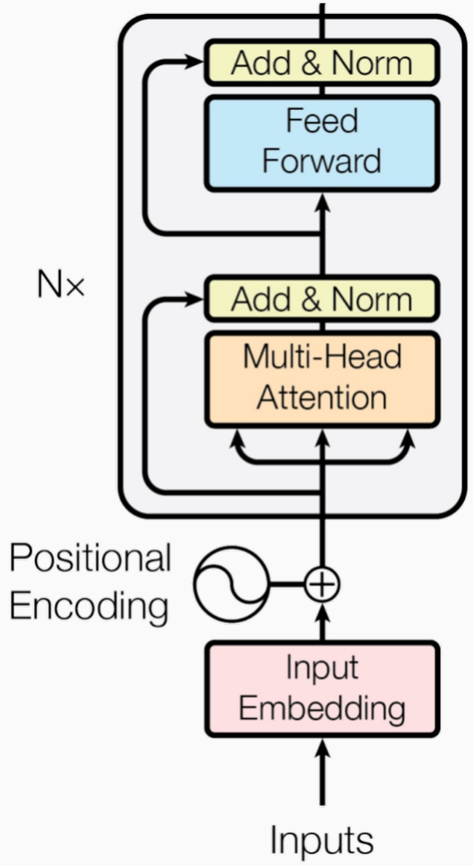
$BERT_{BASE}$

768 dimensional
embeddings per word

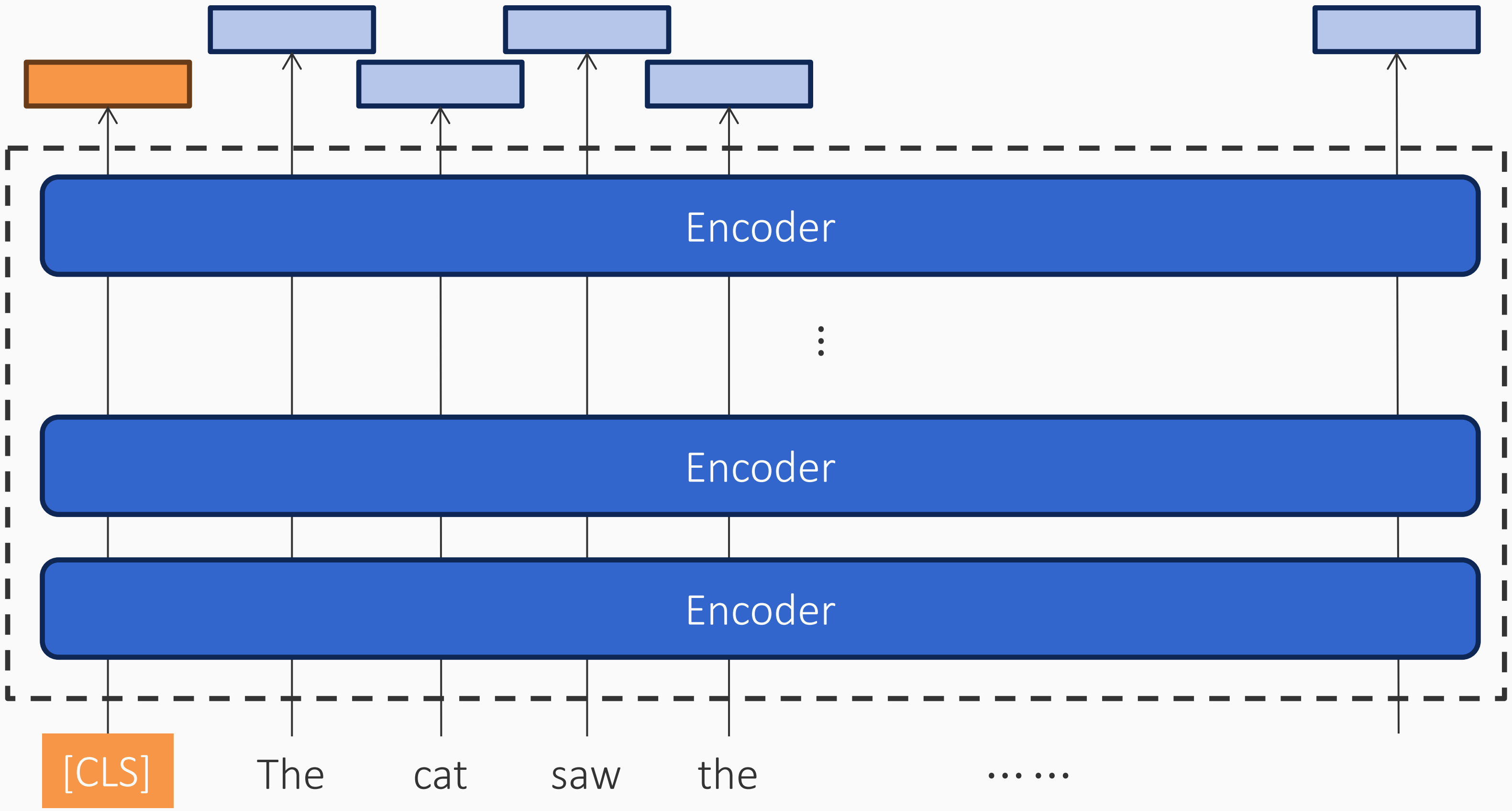


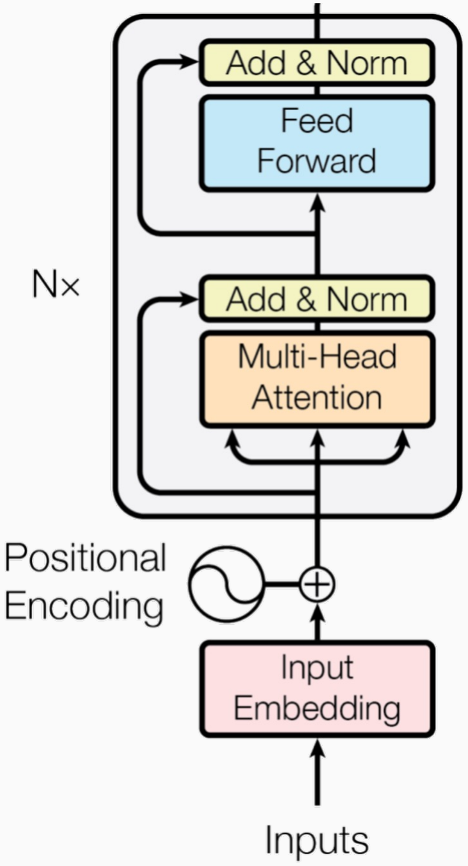
$BERT_{LARGE}$

1024 dimensional
embeddings per word



BERT: Architecture





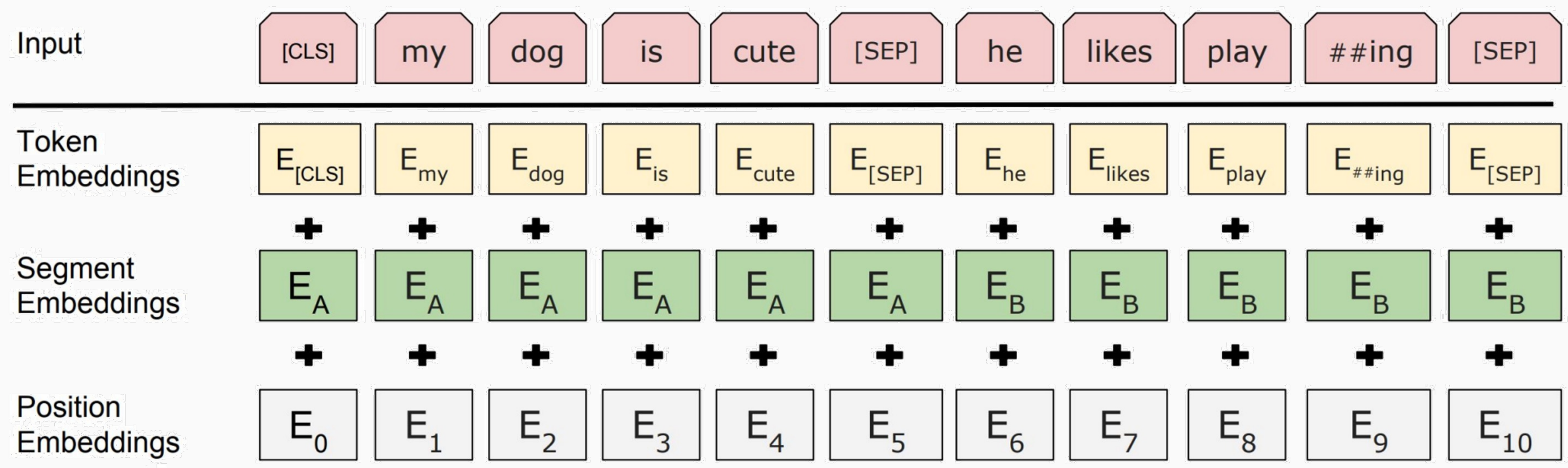
The original BERT could handle sequences of up to 512 tokens.
What happens if your sequence is bigger than that? Ideas?

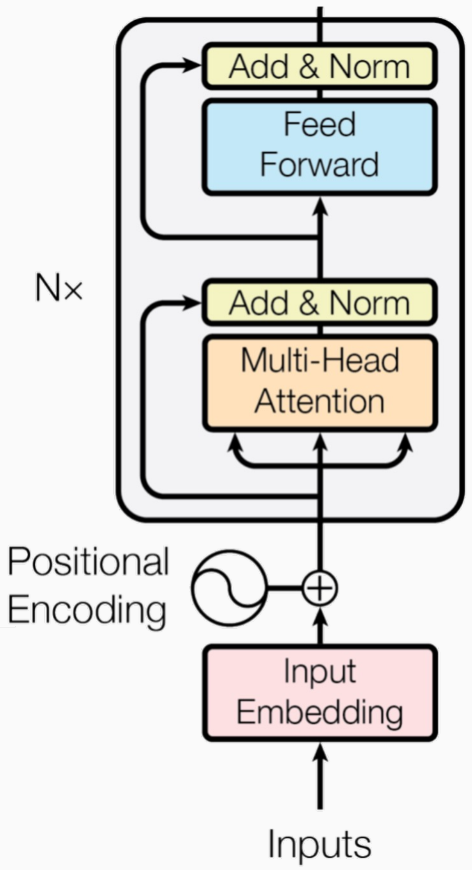
BERT: Input Representation

Use 30,000 WordPiece vocabulary on input.

Each token is sum of three embeddings

- Addition to transformer encoder: sentence embedding





Outline

- Transformers: Recap
- What is BERT?
- Pre-training and fine-tuning
- The impact of BERT
- What does BERT "know"?

Training versus Pre-train + Fine-tune

Standard NLP: For a given task, train a model for that task

- Invent task-specific features for that task
- Maybe use existing word embeddings

Training versus Pre-train + Fine-tune

Standard NLP: For a given task, train a model for that task

- Invent task-specific features for that task
- Maybe use existing word embeddings

An alternate workflow

- First train a contextualized embedding model that "knows" the language
  - Converts tokens into context-sensitive embeddings
  - This is the pre-training phase, which could be long running
- Use the pre-trained model for a task
  - Fine-tune the model for a relatively small number of updates

How is BERT pre-trained?

BERT provides embeddings for each token in its input

The goal: After pre-training, the words should have “good” contextualized embeddings for any task in the future

How can we train for this? What did ELMo do?

g:

How is BERT pre-trained?

BERT provides embeddings for each token in its input

The goal: After pre-training, the words should have “good” contextualized embeddings for any task in the future

How can we train for this? What did ELMo do?

BERT uses two surrogate tasks (and corresponding objectives) for pre-training:

1. Masked language modeling
2. Next sentence prediction

How is BERT pre-trained?

BERT provides embeddings for each token in its input

The goal: After pre-training, the words should have “good” contextualized embeddings for any task in the future

How can we train for this? What did ELMo do?

BERT uses two surrogate tasks (and corresponding objectives) for pre-training:

1. Masked language modeling
2. Next sentence prediction

BERT: Pre-training Objective (1): Masked Tokens

Randomly mask 15% of the tokens and train the model to predict them.

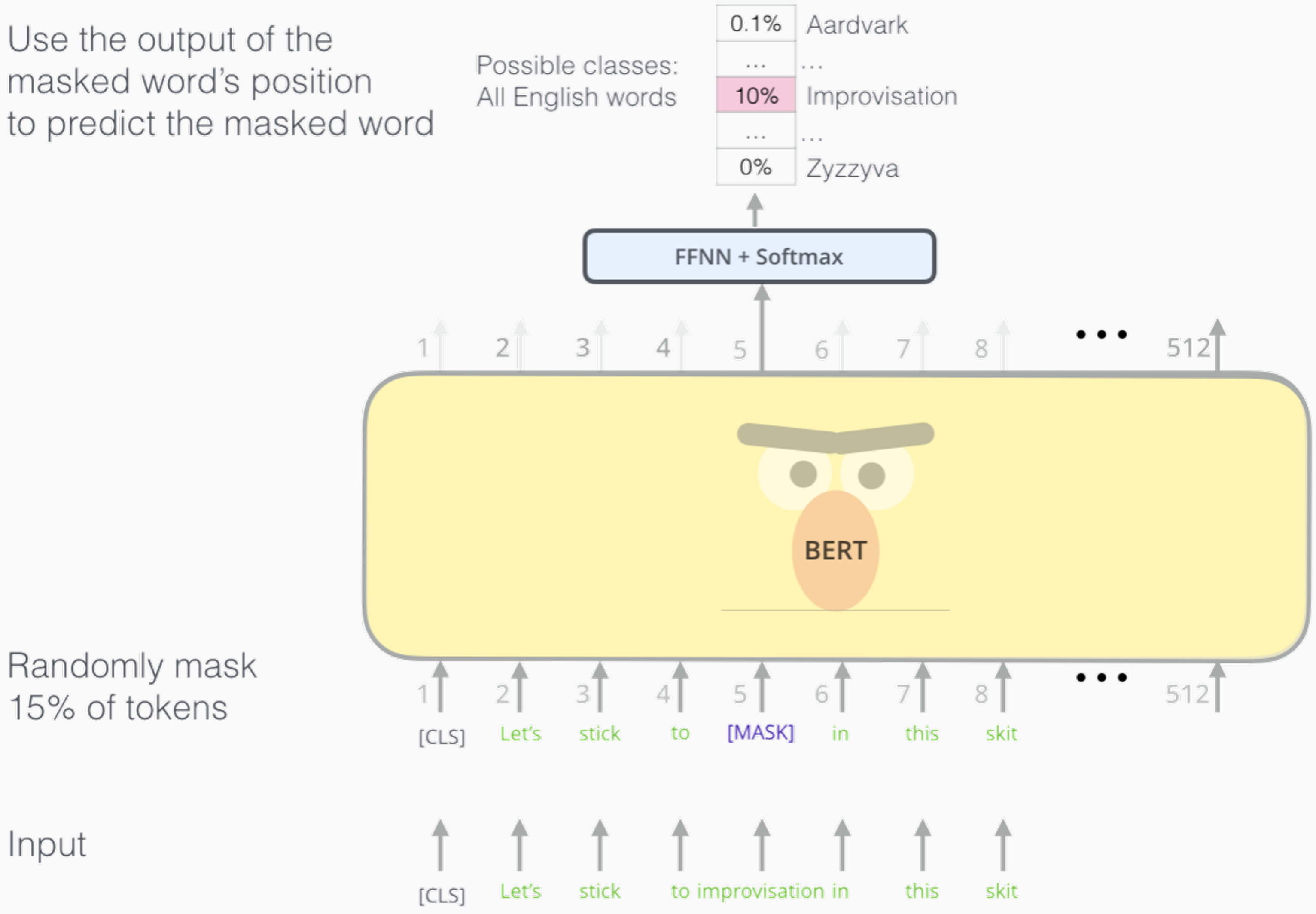
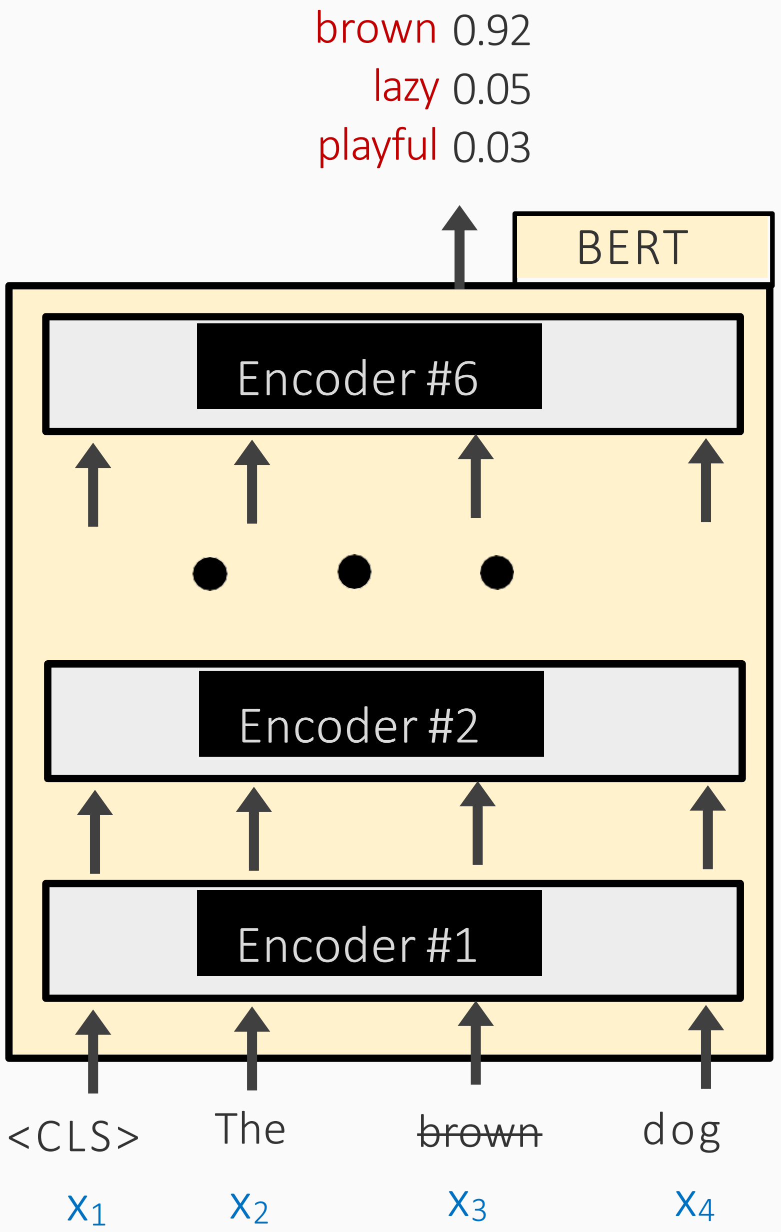


Image from https://jalammar.github.io/illustrated-bert/

The masked language modeling objective



Predict the Masked word (a la CBOW)

15% of all input words are randomly masked.

- 80% become [MASK]
- 10% become revert back
- 10% become are deliberately corrupted as wrong words

BERT: Pre-training Objective (1): Masked Tokens

store

gallon

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

- Too little masking: Too expensive to train
- Too much masking: Underdefined (not enough context)

Slide from Daniel Kashabi

How is BERT pre-trained?

BERT provides embeddings for each token in its input

The goal: After pre-training, the words should have “good” contextualized embeddings for any task in the future

How can we train for this? What did ELMo do?

BERT uses two surrogate tasks (and corresponding objectives) for pre-training:

1. Masked language modeling
2. Next sentence prediction

BERT: Pre-training Objective (2): Sentence Ordering

- Predict sentence ordering
- 50% correct ordering, and 50% random incorrect ones

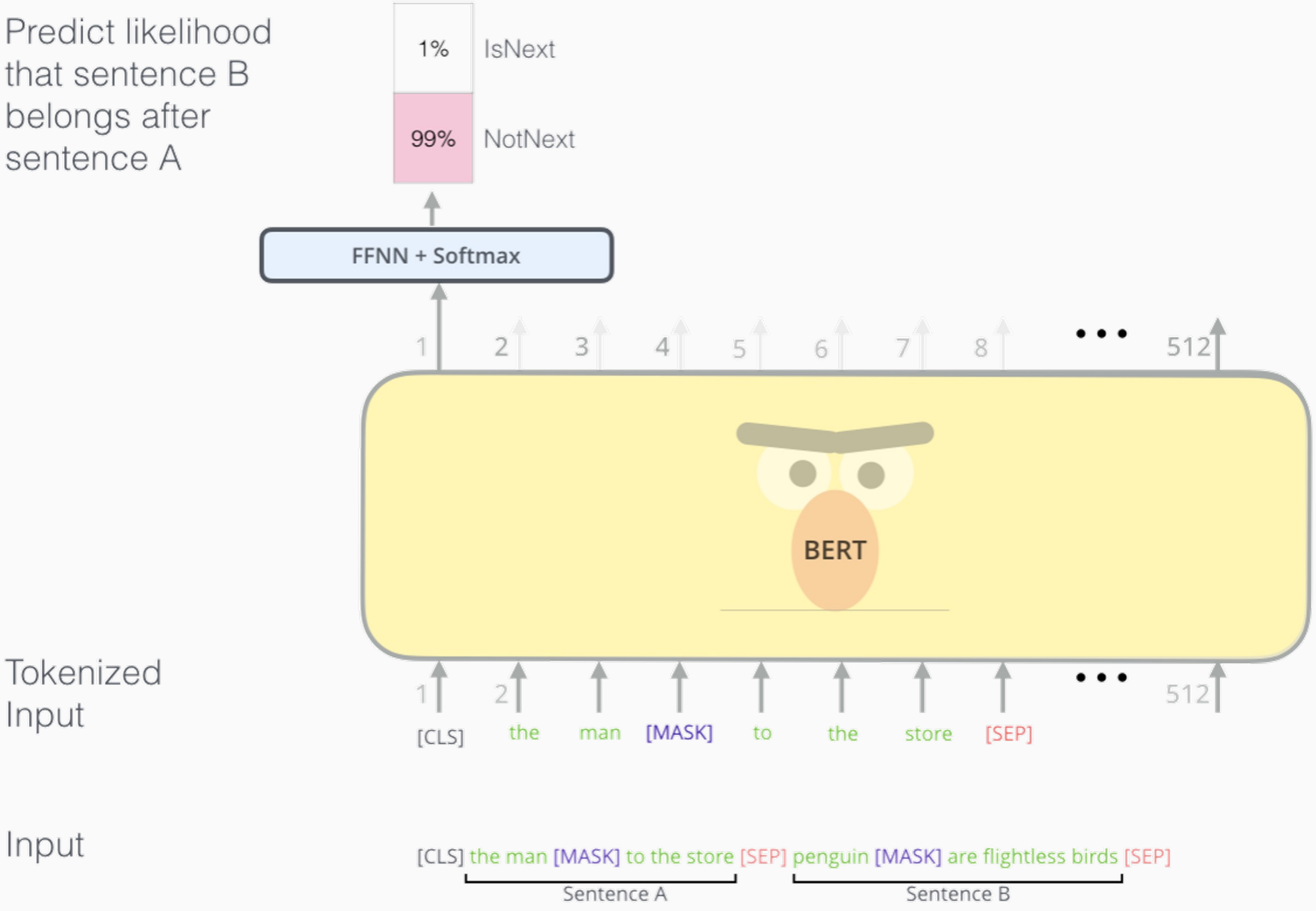
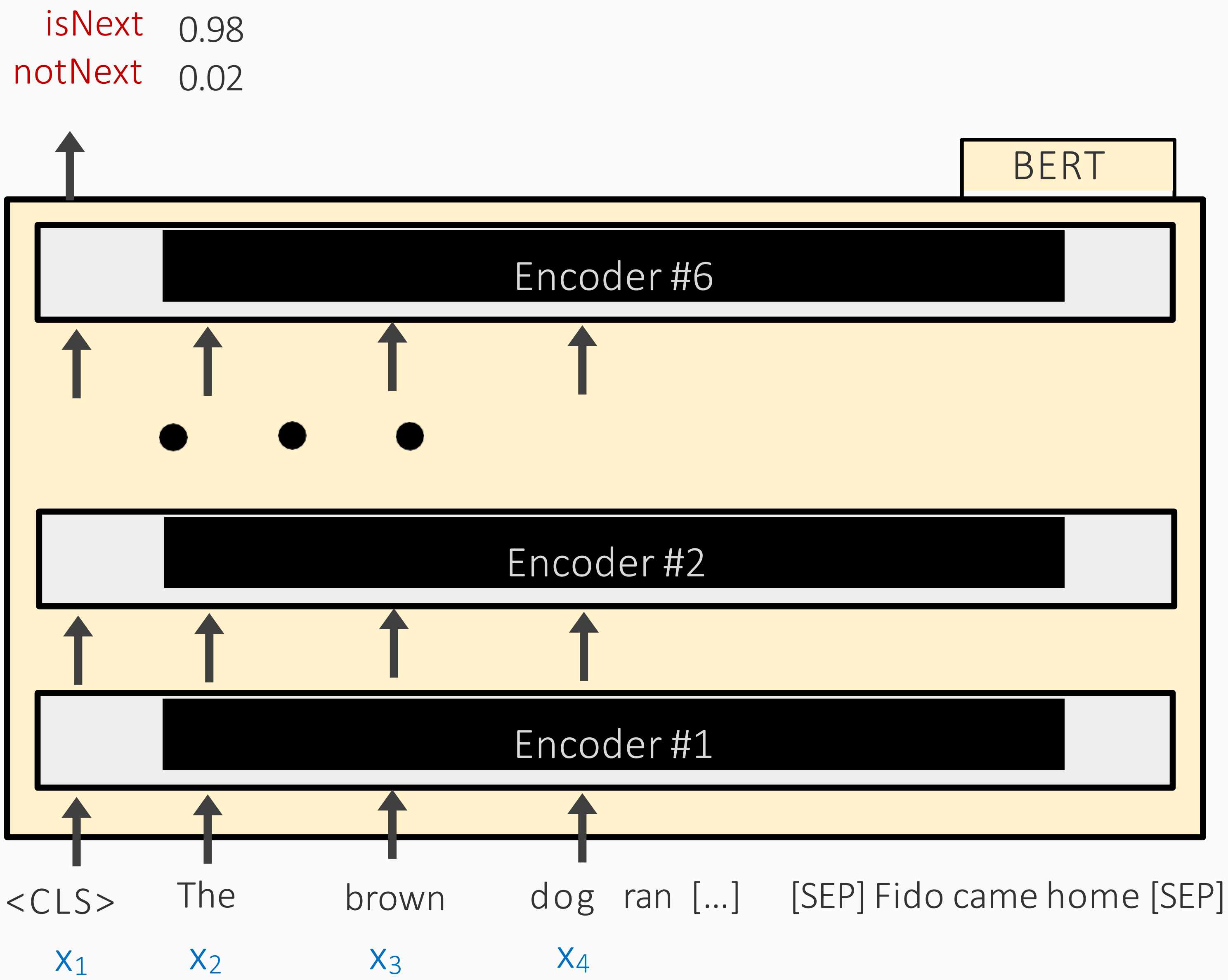


Image from https://jalammar.github.io/illustrated-bert/

The next sentence prediction objective



Two sentences are fed in at a time. Predict the if the second sentence of input truly follows the first one or not.

BERT: Pre-training Objective (2): Sentence Ordering

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 A** = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`	**Sentence B** = `Penguins are flightless.`
Label = `IsNextSentence`	**Label** = `NotNextSentence`

Training details

Trains model on unlabeled data over different pre-training tasks (self-supervised learning)

Data: Wikipedia (2.5B words) + BookCorpus (800M words)

Training Time: 1M steps (~40 epochs), 4 days

Optimizer: AdamW, 1e-4 learning rate, linear decay

BERT-Base: 12-layer, 768-hidden, 12-heads

BERT-Large: 24-layer, 1024-hidden, 16-heads

Trained on 16 or 64 TPUs (base or large resp.) for 4 days

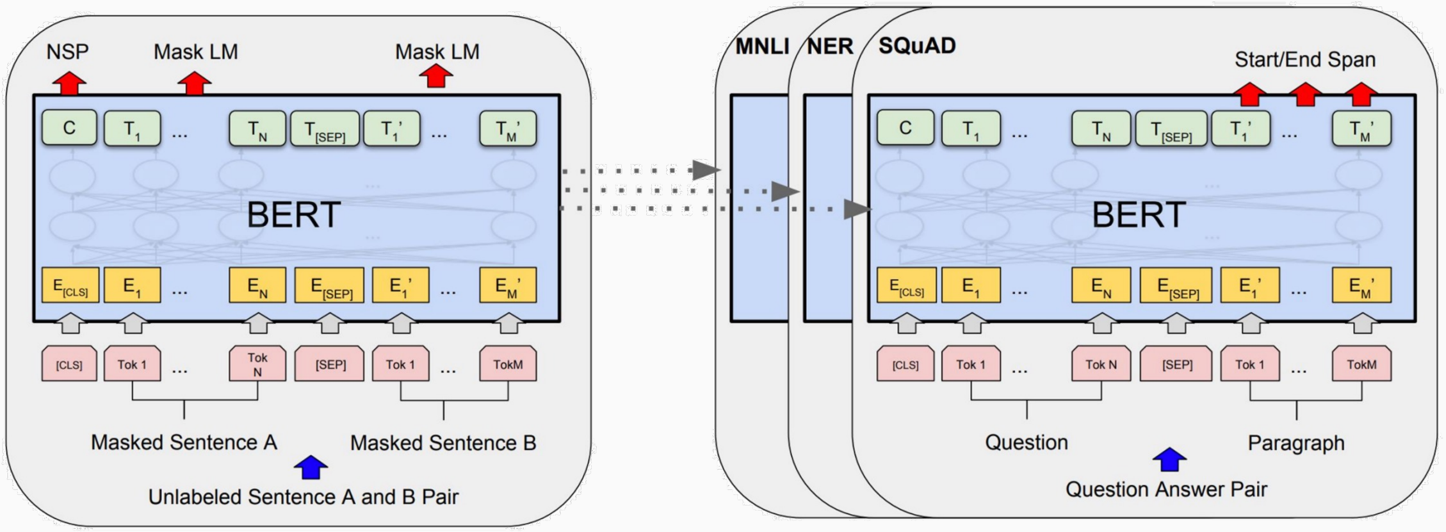
Fine-tuning BERT

"Pretrain once, finetune many times."

Idea: Make pre-trained model **usable** in **downstream tasks**

Initialized with pre-trained model parameters

Fine-tune model parameters using labeled data from downstream tasks



Pre-training Fine-Tuning

An Example Result: SWAG

```
A girl is going across a set of monkey bars.  She
(i)  jumps up across the monkey bars.
(ii)  struggles onto the bars to grab her head.
(iii)  gets to the end and stands on a wooden plank.
(iv)  jumps up and does a back flip.
```

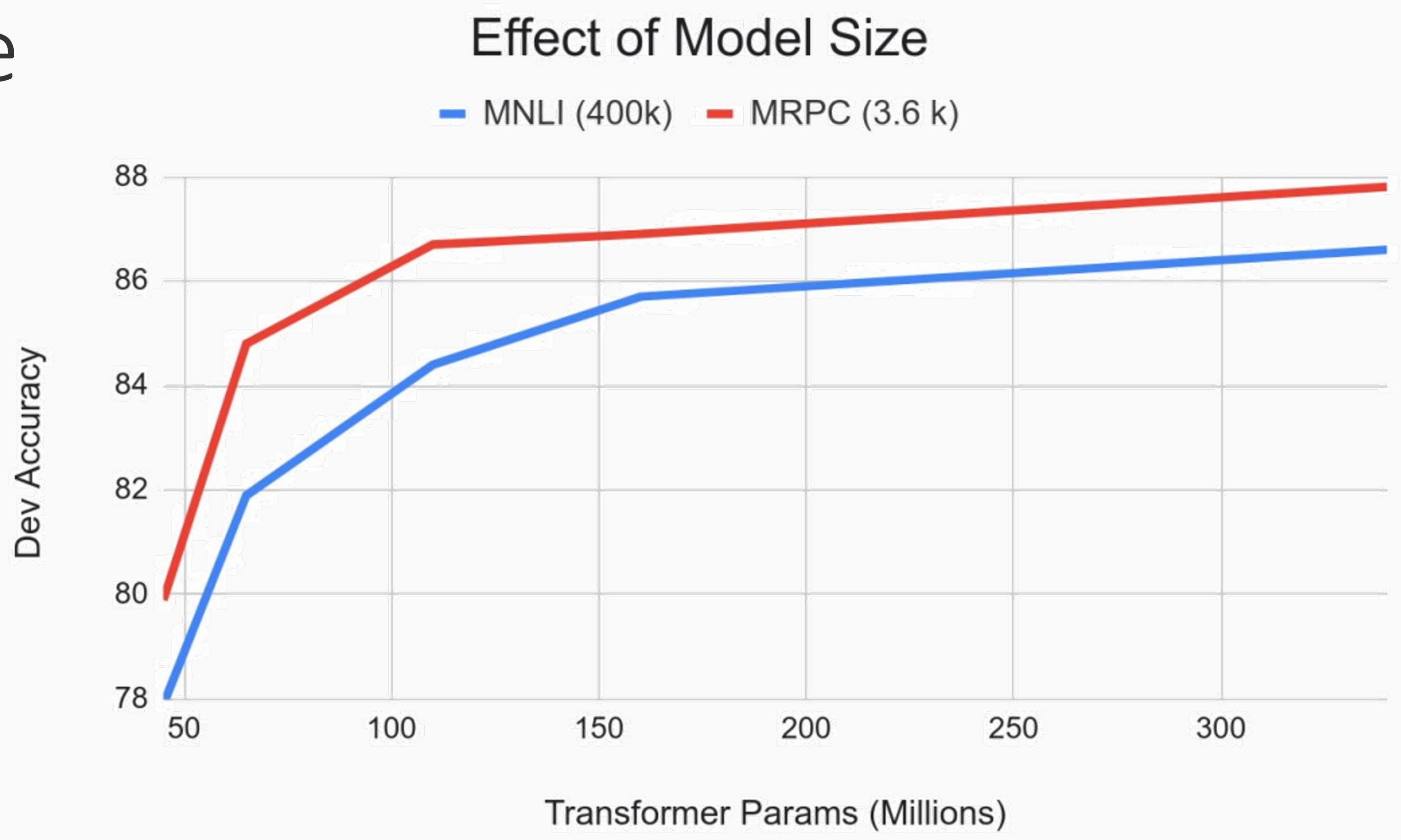
Leaderboard

Human Performance (88.00%)
Running Best
Submissions

Rank	Model	Test Score
1	**BERT (Bidirectional Encoder Representations from Transfo...** *Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova* 10/11/2018	**86.28%**
2	**OpenAI Transformer Language Model** *Original work by Alec Radford, Karthik Narasimhan, Tim Salimans, ...* 10/11/2018	**77.97%**
3	**ESIM with ELMo** *Zellers, Rowan and Bisk, Yonatan and Schwartz, Roy and Choi, Yejin* 08/30/2018	**59.06%**
4	**ESIM with Glove** *Zellers, Rowan and Bisk, Yonatan and Schwartz, Roy and Choi, Yejin* 08/29/2018	**52.45%**

- Run each Premise + Ending through BERT.
- Produce logit for each pair on token 0 ([CLS])

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*

Outline

- Transformers: Recap
- What is BERT?
- Pre-training and fine-tuning
- The impact of BERT
- What does BERT “know”?

Why did no one think of this before?

Concretely, why wasn’t contextual pre-training popular before 2018 with ELMo?

Good results on pre-training is >1,000x to 100,000 more expensive than supervised training.

What Happened After BERT?

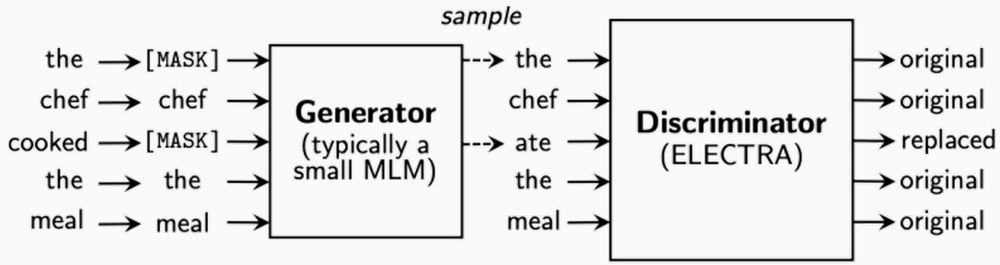
RoBERTa (Liu et al., 2019)

- Drops the next sentence prediction loss!
- Trained on 10x data (the original BERT was actually under-trained)
- Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)
- Still one of the most popular models to date

Model	**data**	**bsz**	**steps**	**SQuAD** (v1.1/2.0)	**MNLI-m**	**SST-2**
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

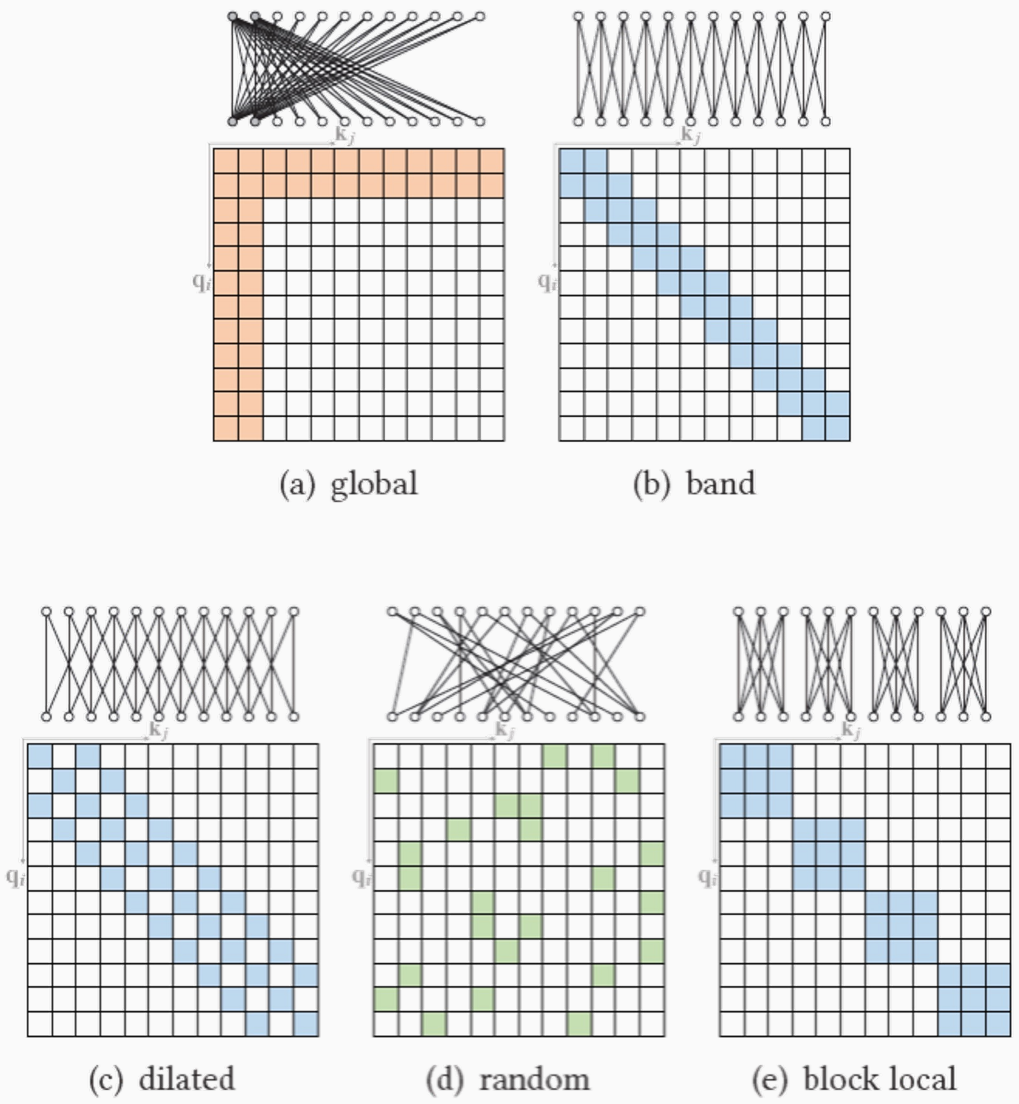
What Happened After BERT?

- RoBERTa (Liu et al., 2019)
  - Drops the next sentence prediction loss!
  - Trained on 10x data (the original BERT was actually under-trained)
  - Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)
  - Still one of the most popular models to date
- ALBERT (Lan et al., 2020)
  - Increasing model sizes by sharing model parameters across layers
  - Less storage, much stronger performance but runs slower..
- ELECTRA (Clark et al., 2020)
  - Two models generator and discriminator
  - It provides a more efficient training method



What Happened After BERT?

- Models that handle long contexts (512 tokens)
  - Longformer, Big Bird, …
- Multilingual BERT
  - Trained single model on 104 languages from Wikipedia. Shared 110k WordPiece vocabulary
- BERT extended to different domains
  - SciBERT, BioBERT, FinBERT, ClinicalBERT, …
- Making BERT smaller to use
  - DistillBERT, TinyBERT, …



Outline

- Transformers: Recap
- What is BERT?
- Pre-training and fine-tuning
- The impact of BERT
- What does BERT “know”?

Probing BERT

Motivating question: BERT (and its siblings) are successful at a wide range of NLP tasks. Why?

- What linguistic knowledge does BERT encode?
- What world knowledge does it encode?
- What information is encoded in each layer of the model?

Ideas?

Probing BERT

Motivating question: BERT (and its siblings) are successful at a wide range of NLP tasks. Why?

- What linguistic knowledge does BERT encode?
- What world knowledge does it encode?
- What information is encoded in each layer of the model?

How can we set up an experiment to test for this?
Answer: A productive research area that is generally called *probing*

Probing BERT

Motivating question: BERT (and its siblings) are successful at a wide range of NLP tasks. Why?

What linguistic knowledge does BERT encode?
What world knowledge does it encode?
What information is encoded in each layer of the model?

How can we set up an experiment to test for this?
Answer: A productive research area that is generally called *probing*

Two broad approaches for probing:

1. Predictive probing: Train classifiers for standard tasks on the representations at various layers
   – E.g.: Tenney et al (2019), Liu et al (2019), Kovaleva et al (2019), Pimentel et al (2020) and many others

Probing BERT

Motivating question: BERT (and its siblings) are successful at a wide range of NLP tasks. Why?

What linguistic knowledge does BERT encode?
What world knowledge does it encode?
What information is encoded in each layer of the model?

How can we set up an experiment to test for this?
Answer: A productive research area that is generally called *probing*

Two broad approaches for probing:

1. Predictive probing: Train classifiers for standard tasks on the representations at various layers
   – E.g.: Tenney et al (2019), Liu et al (2019), Kovaleva et al (2019), Pimentel et al (2020) and many others
2. Descriptive probing: Looking at patterns (e.g. geometric) in the embeddings not necessarily with respect to a particular task
   – E.g. Reif et al (2019), Ethayarajh (2019), Zhou and Srikumar (2021) and others

What BERT knows? (according to probes)

Encodes information about parts of speech and syntax

- Hewitt and Manning showed that dependency trees can be directly extracted from BERT (It was not trained to produce or know the dependency formalism!)
- But surprisingly it is not sensitive to word order, malformed inputs and does not know about negation

Knows semantic roles and what kinds of entities can play certain roles

- Encodes information about entity types, relations, semantic roles, and proto-roles
- Struggles with numbers

Encodes *some* world knowledge

- But seems to struggle with chaining that knowledge to reason about the world

Rogers et al 2020. "A Primer in BERTology: What we know about how BERT works."

Where does BERT store the knowledge?

Some BERT heads seem to specialize in certain types of syntactic relations

- But no single head has the complete syntactic tree information

Lower layers have the most linear word order information

- Word order information gets jumbled after layer 4

Syntactic information is best represented in some of the middle layers

- Layers 6-9 for BERT-base and 14-19 for BERT-large

Final layers of BERT are the most task specific

Rogers et al 2020. "A Primer in BERTology: What we know about how BERT works."

What does fine-tuning for a task do?

- Fine-tuning pushes task labels far away from each other
- Fine-tuning makes the representation “better” suited for the specific task, but in doing so, could make it worse for other tasks
- Lower layers do not change much during fine tuning
- Higher layers change, but largely retain their geometric structure

Zhou and Srikumar (2022). “A closer look at how fine-tuning changes BERT”

Rogers, Anna, Olga Kovaleva, and Anna Rumshisky. "A Primer in BERTology: What we know about how BERT works." *arXiv preprint arXiv:2002.12327* (2020).

A Primer in BERTology: What We Know About How BERT Works

Anna Rogers
Center for Social Data Science
University of Copenhagen
arogers@sodas.ku.dk

Olga Kovaleva
Dept. of Computer Science
University of
Massachusetts Lowell
okovalev@cs.uml.edu

Anna Rumshisky
Dept. of Computer Science
University of
Massachusetts Lowell
arum@cs.uml.edu

Abstract

Transformer-based models have pushed state of the art in many areas of NLP, but our understanding of what is behind their success is still limited. This paper is the first survey of over 150 studies of the popular BERT model. We review the current state of knowledge about how BERT works, what kind of information it learns and how it is represented, common modifications to its training objectives and architecture, the overparameterization issue, and approaches to compression. We then outline directions for future research.

and provide an overview of the current proposals to improve BERT's architecture, pre-training, and fine-tuning. We conclude by discussing the issue of overparameterization, the approaches to compressing BERT, and the nascent area of pruning as a model analysis technique.

2 Overview of BERT Architecture

Fundamentally, BERT is a stack of Transformer encoder layers (Vaswani et al., 2017) that consist of multiple self-attention ''heads''. For every input token in a sequence, each head computes key, value, and query vectors, used to create a weighted

Summary

- BERT and the family
- An encoder; Transformer-based networks trained on massive piles of data.
- Incredible for learning contextualized embeddings of words
- It’s very useful to pre-train a large unsupervised/self-supervised LM then fine-tune on your particular task (replace the top layer, so that it can work)
- However, they were not designed to generate text.