Economics 2355: Transformer-Based Language Models

Melissa Dell

Harvard University

March 2021

Outline

Overview
Contextualized Word Embeddings
GPT
BERT
RoBERTa
DistilBERT
ALBERT
T5
GPT2/GPT3
Transformers XL
XLNet
Longformer
BigBird
Overview and What to Use

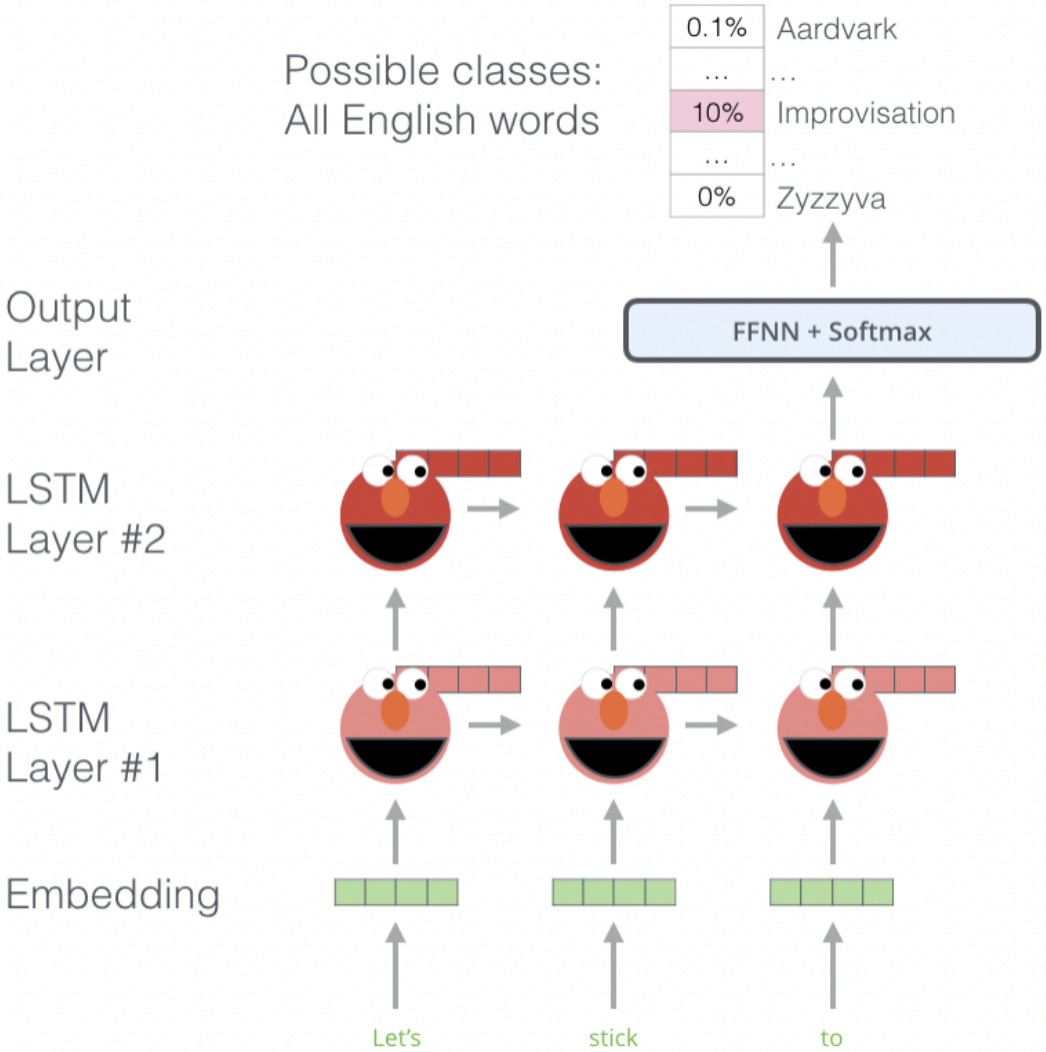
Contextualized Word Embeddings

- With word embeddings from a model like Word2Vec or GloVe, a given word will always have the same embedding, regardless of its context
- This is a problem, as many words have multiple meanings depending on the context in which they are used
- A transformer based model is not required to contextualize embeddings (although these models came to dominate for reasons we'll see)

ELMO

- ELMo was a prominent model in the development of contextualized word embeddings
- ELMo is a language model whose objective is to predict the next word in a sequence
- It also helped to popularize the approach of using a pre-trained language model to create representations for various downstream tasks

ELMo is based on an LSTM

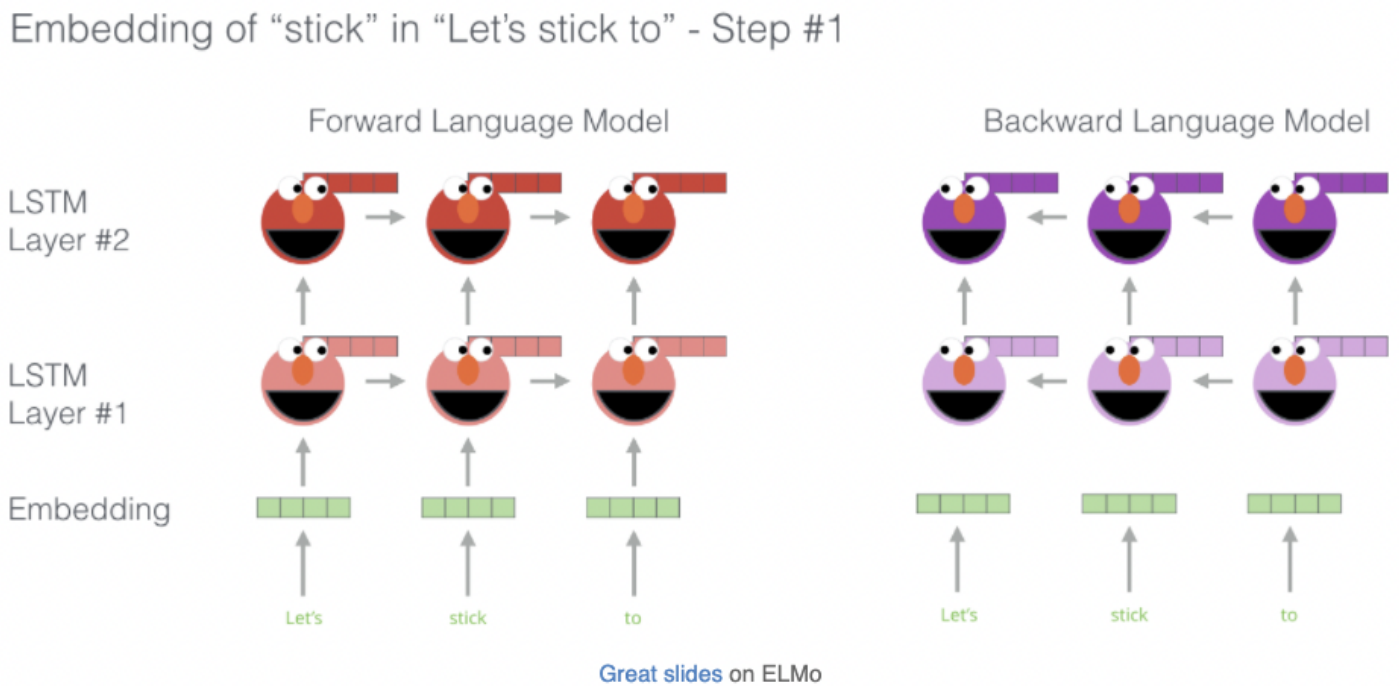


A step in the pre-training process of ELMo: Given "Let's stick to" as input, predict the next most likely word – a *language modeling* task. When trained on a large dataset, the model starts to pick up on language patterns. It's unlikely it'll accurately guess the next word in this example. More realistically, after a word such as "hang", it will assign a higher probability to a word like "out" (to spell "hang out") than to "camera".

http://jalammar.github.io/illustrated-bert/

ELMo is based on an LSTM

It is bi-directional, to allow contexts from both directions to matter

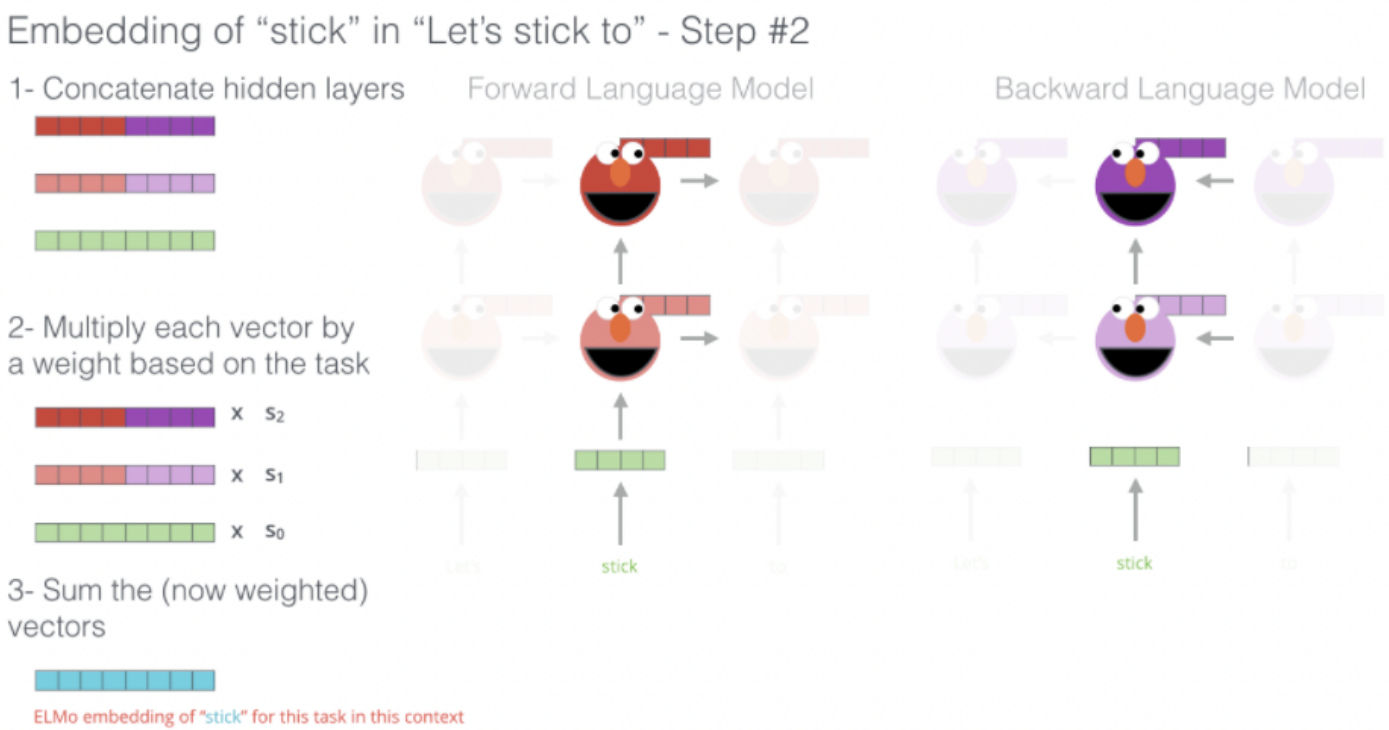


http://jalammar.github.io/illustrated-bert/

Creating Contextualized Embeddings

- ELMo creates contextualized embeddings by concatenating the hidden states and initial embeddings across the forward and backward LSTMs
- Then it takes a weighted sum of the hidden states and initial embedding vectors

Creating Contextualized Embeddings



http://jalammar.github.io/illustrated-bert/

To create the initial embeddings, the model combines character level embeddings

Outline

Overview

Contextualized Word Embeddings

GPT

BERT

RoBERTa

DistilBERT

ALBERT

T5

GPT2/GPT3

Transformers XL

XLNet

Longformer

BigBird

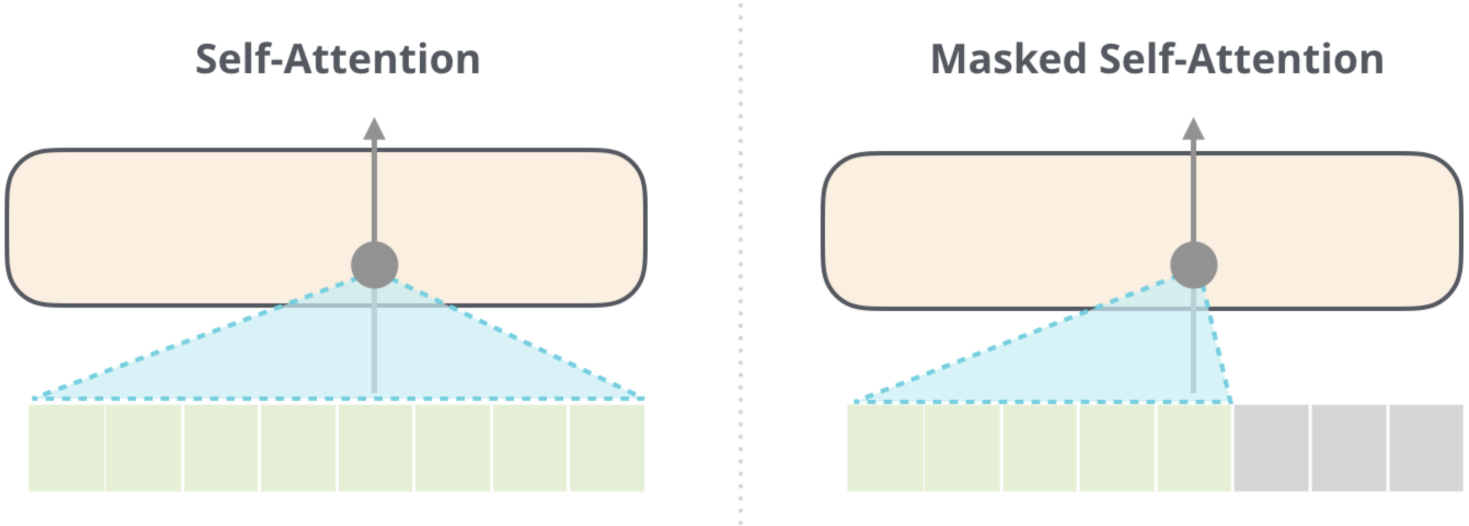
Overview and What to Use

Open AI GPT Model

- The first example of a stand-alone transformer based language model, to my knowledge, was the Open AI GPT
- The original transformer contained a language model - the decoder side. That is, the objective of the decoder side is to predict what word comes next
- It does this by masking future tokens
- The decoder language model does not have to be combined with the encoder for machine translation, it can be used as a standalone language model

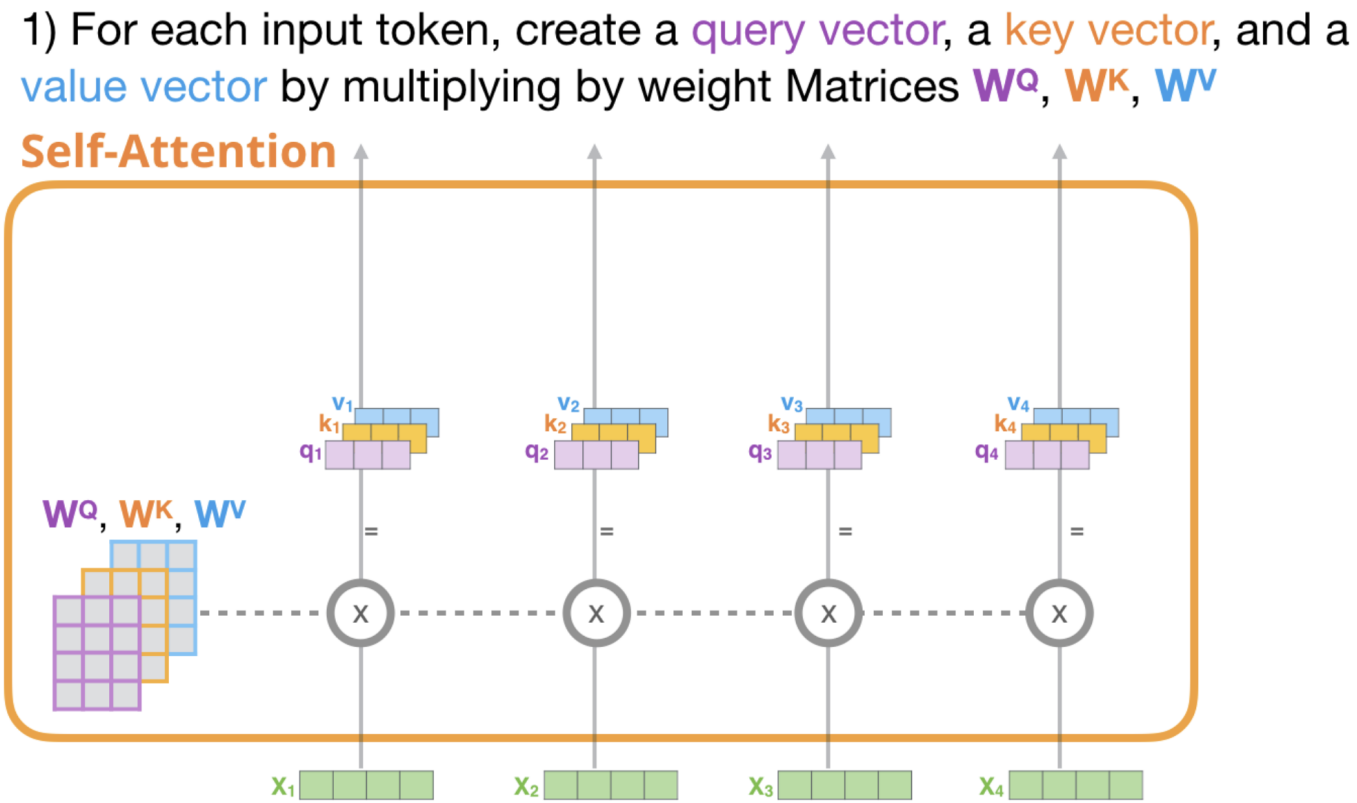
The Decoder Uses Masked Self-Attention

In the encoder self-attention blocks, a position can see tokens from the right. This is not the case for the decoder blocks. The idea here is akin to predicting the next word, given the previous contexts.



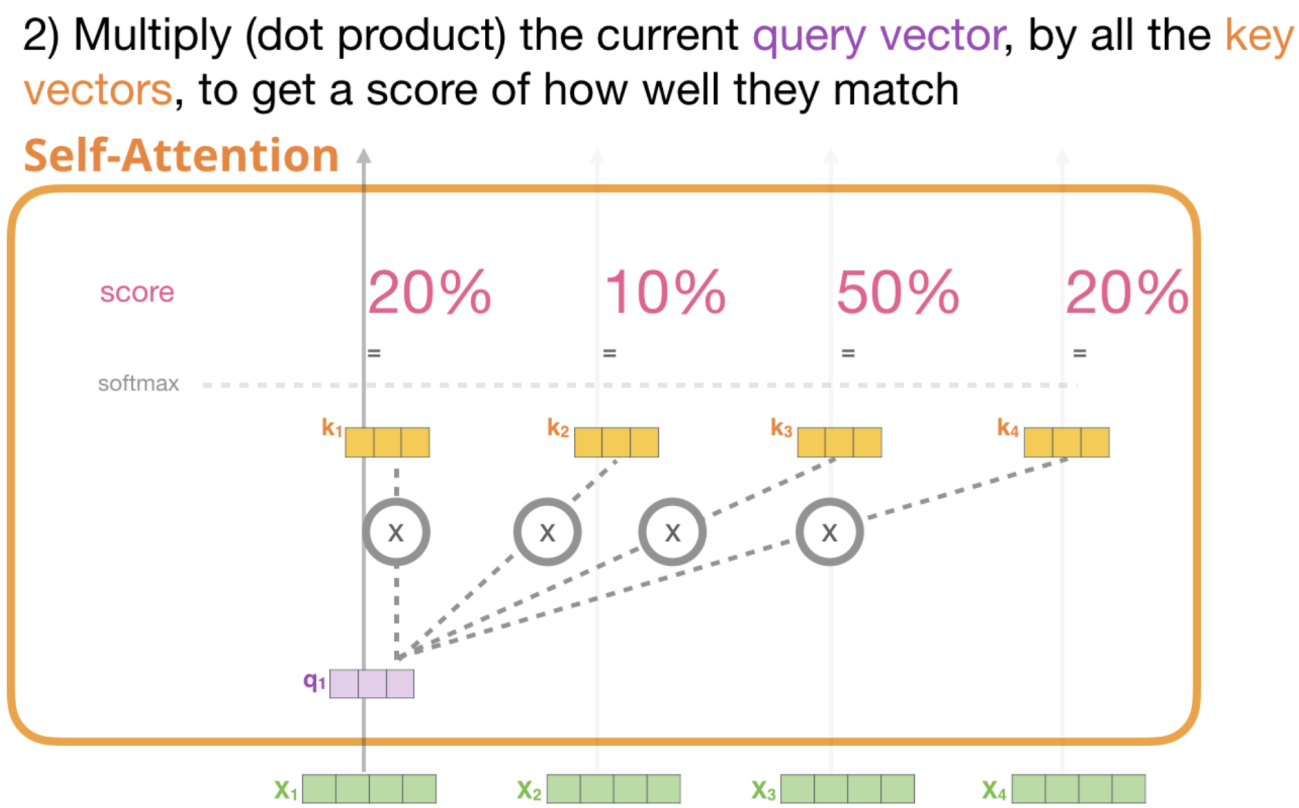
https://jalammar.github.io/illustrated-gpt2/

A Recap of Self-Attention without Masking: Step 1



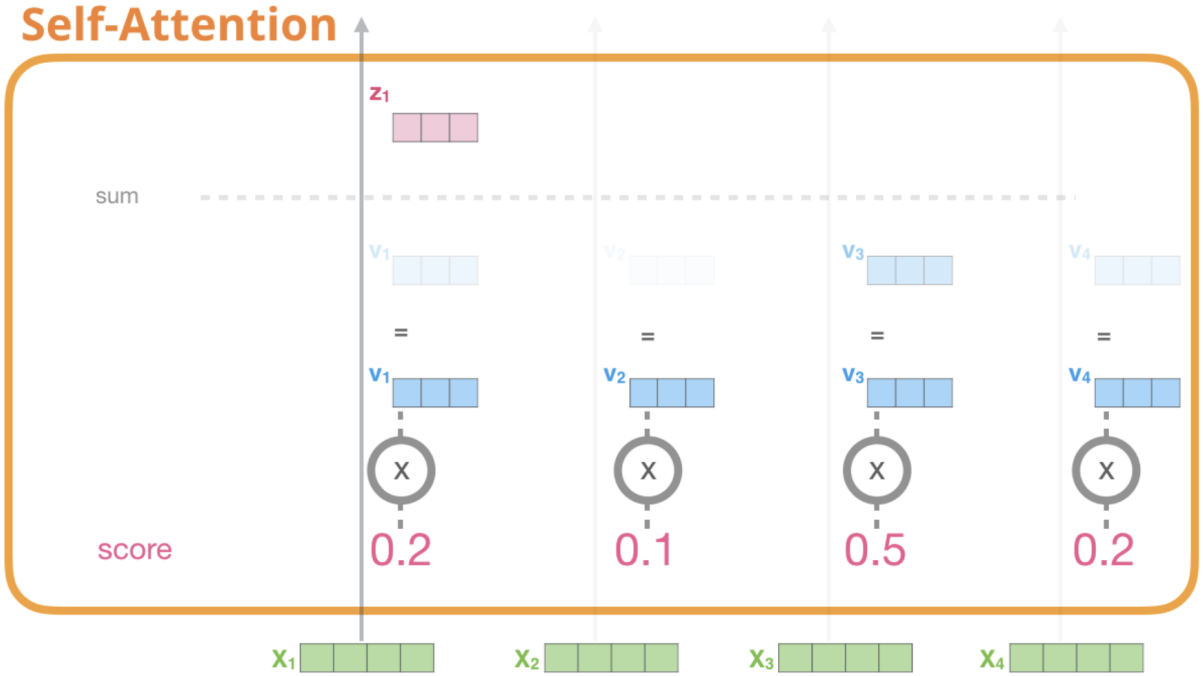
https://jalammar.github.io/illustrated-gpt2/

A Recap of Self-Attention without Masking: Step 2



https://jalammar.github.io/illustrated-gpt2/

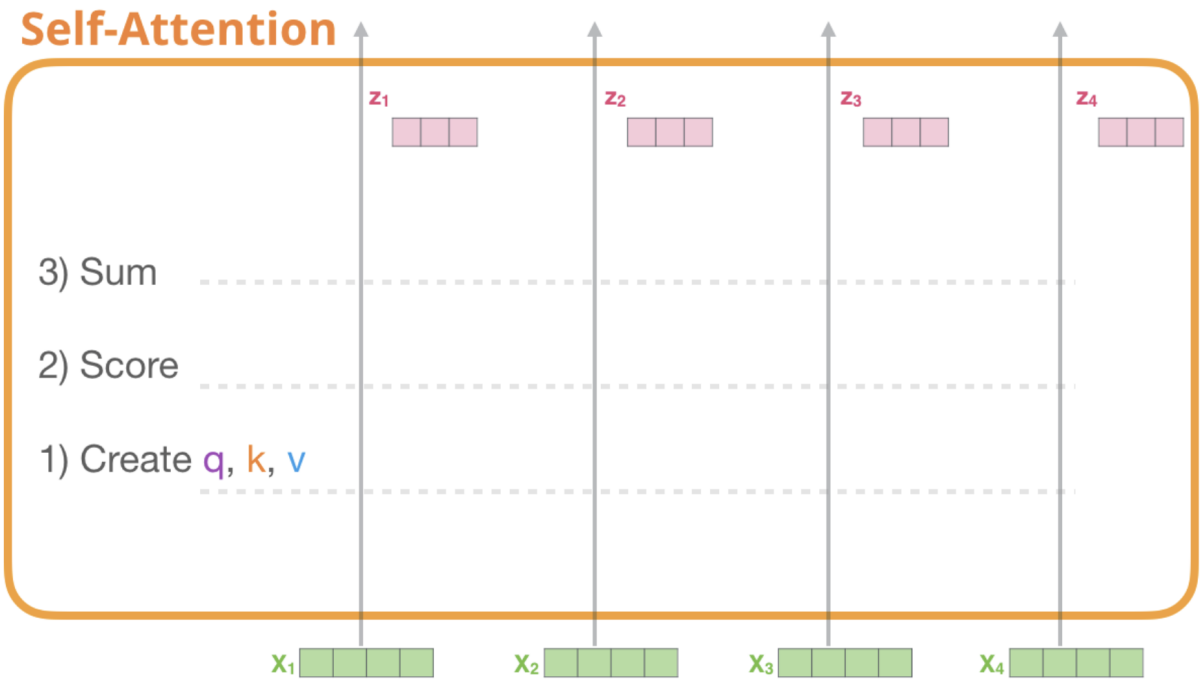
A Recap of Self-Attention without Masking: Step 3



The lower the score, the more transparent we're showing the value vector. That's to indicate how multiplying by a small number dilutes the values of the vector.

https://jalammar.github.io/illustrated-gpt2/

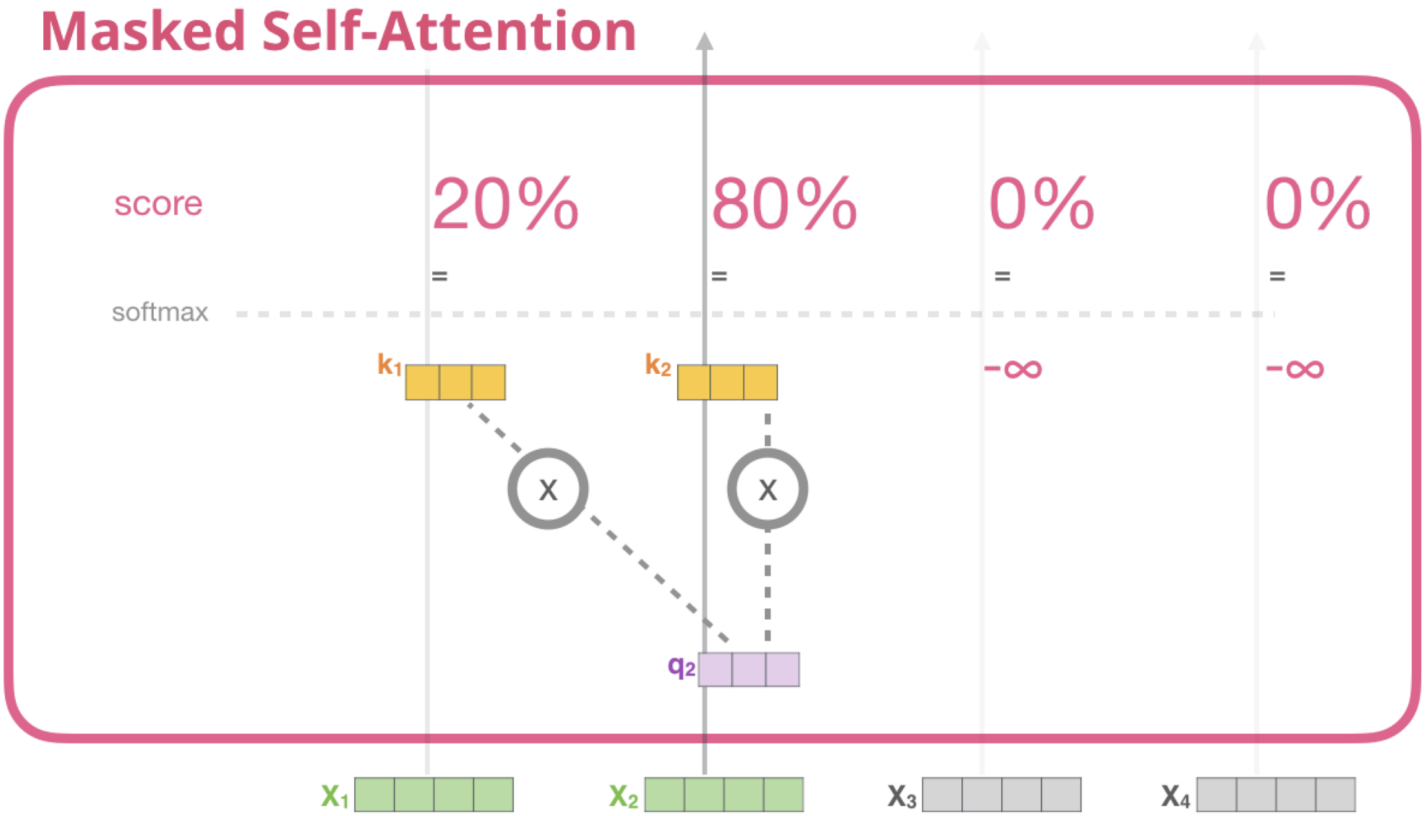
A Recap of Self-Attention without Masking: Step 4



https://jalammar.github.io/illustrated-gpt2/

Masked Self-Attention

Masked self-attention is identical to self-attention without masking, except at step 2.



https://jalammar.github.io/illustrated-gpt2/

Masked Self-Attention

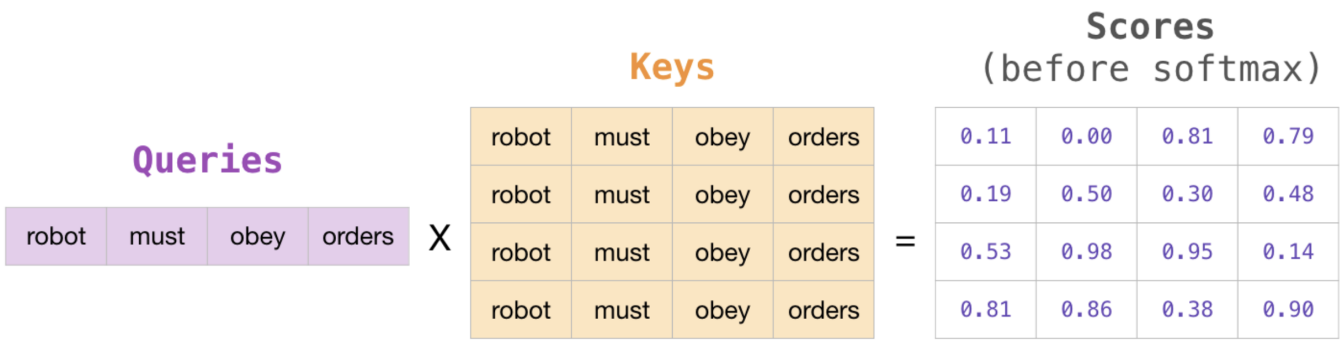
Consider a sentence with four words:

	Features				Labels
position:	1	2	3	4	
Example: 1	robot	must	obey	orders	must
2	robot	must	obey	orders	obey
3	robot	must	obey	orders	orders
4	robot	must	obey	orders	<eos>

https://jalammar.github.io/illustrated-gpt2/

Masked Self-Attention

Calculate the attention scores by multiplying the query matrix by the key matrix:



https://jalammar.github.io/illustrated-gpt2/

In practice, each word is associated with a vector

Masked Self-Attention

After creating the scores, multiply by an attention mask, which will be a triangle

https://jalammar.github.io/illustrated-gpt2/

Masked Self-Attention

Then apply softmax

Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

Softmax
(along rows)

Scores

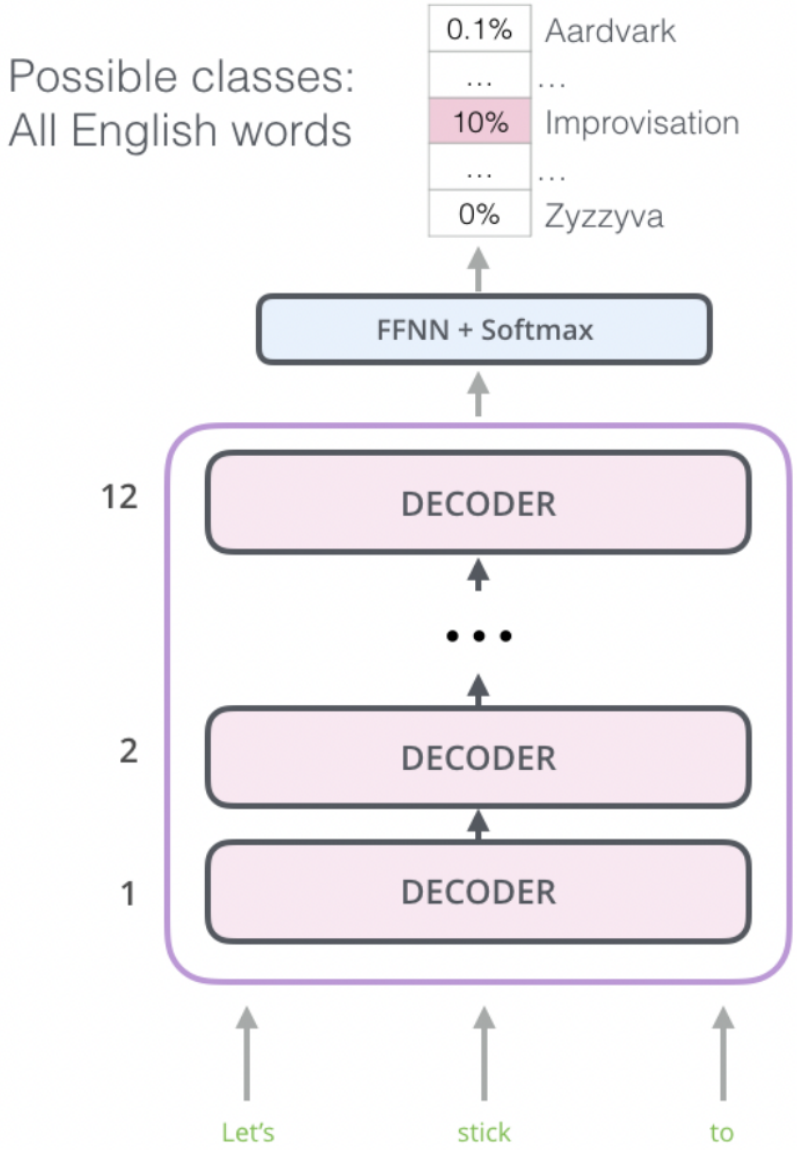
1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26

https://jalammar.github.io/illustrated-gpt2/

GPT

- The original GPT stacked 12 transformer decoder blocks
- Because there is no encoder, it omitted the encoder-decoder attention layers
- The training objective was to predict the next word

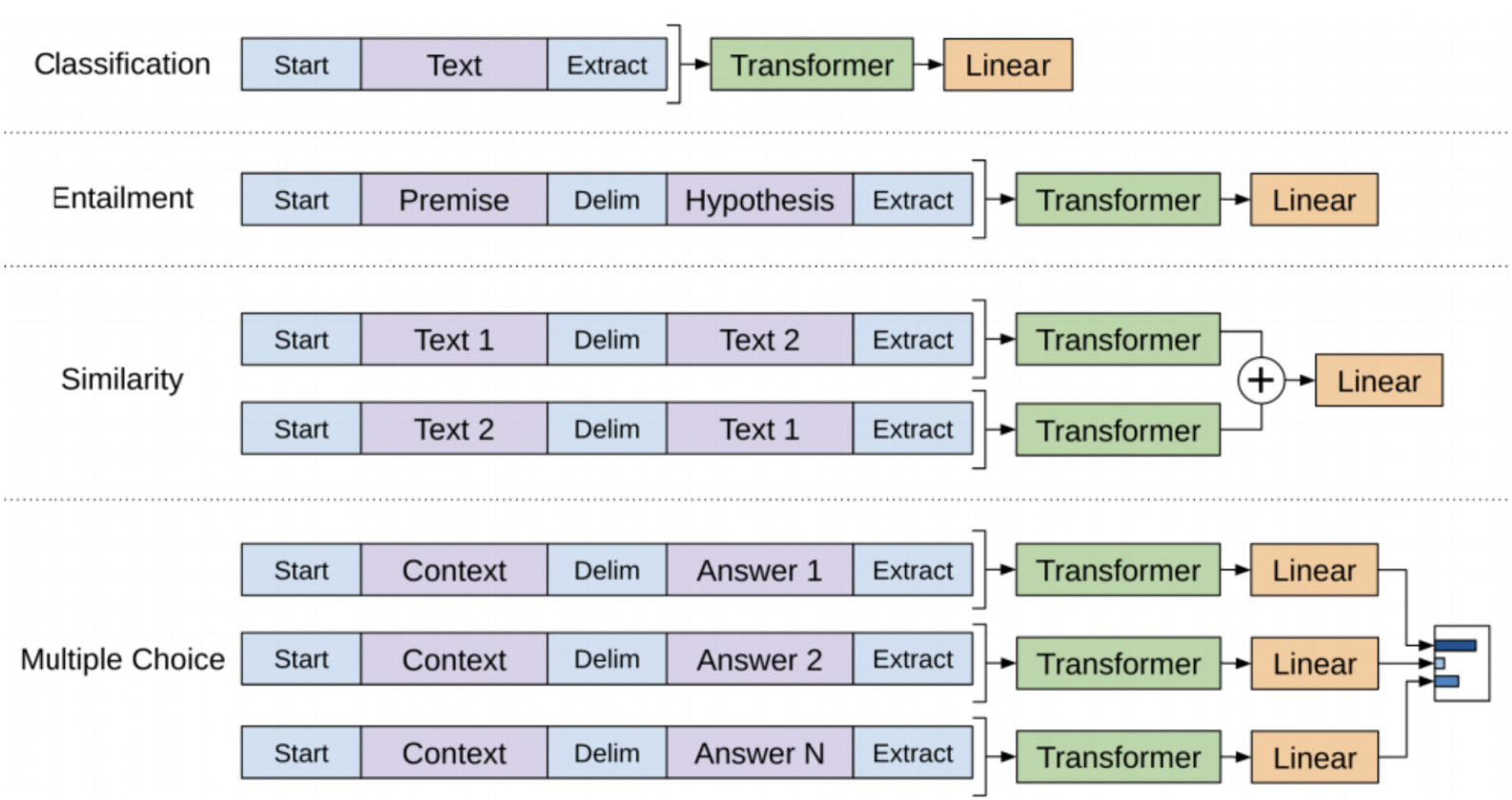
GPT



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

GPT

Inputs can be transformed to configure the model to perform different downstream tasks



Transformers for Language Modeling

- At first it would seem more intuitive to stack decoder blocks, as the decode is a language model.
- But if you just use the decoder blocks, the language model is not bi-directional. You could feed the sequence into the decoder backwards, but then you would need a way to combine the forwards and backwards representations
- The encoder blocks are not a language model. To create representations for each word, the encoder can attend to all other positions in the sequence that was fed into the model
- We'll now see a very influential paper whose innovation was to realize that encoder blocks can be used to create a bidirectional language model through masking

Outline

Overview

Contextualized Word Embeddings

GPT

BERT

RoBERTa

DistilBERT

ALBERT

T5

GPT2/GPT3

Transformers XL

XLNet

Longformer

BigBird

Overview and What to Use

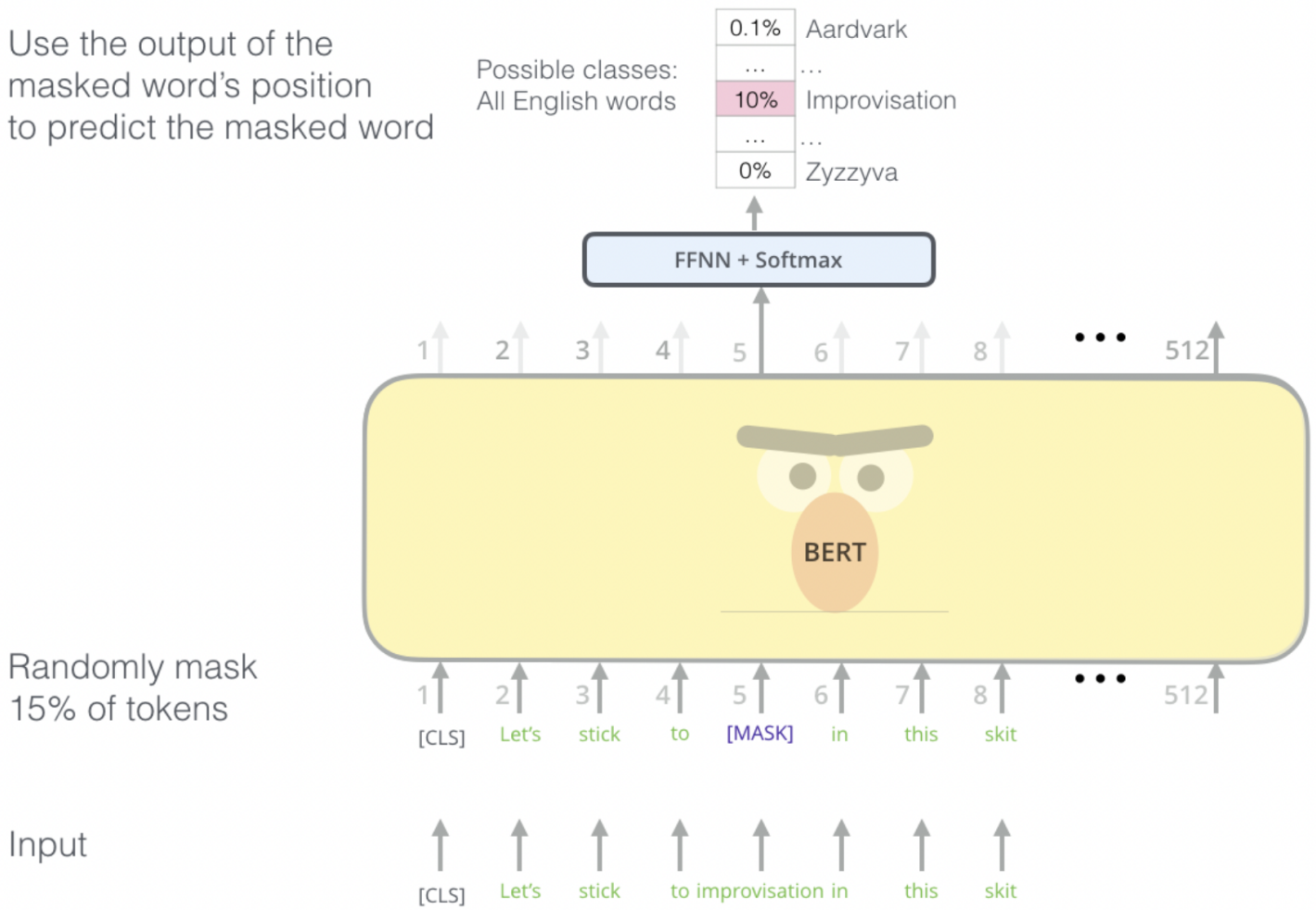
BERT

- BERT, which stands for *Bidirectional Encoder Representations for Transformers*, marked a major leap forward in NLP, developing a Transformer-based language model architecture
- While it continues the Sesame Street theme, the name is also instructive
  - Bidirectional refers to the fact that it considers contexts from both directions (as with the bidirectional LSTM);
  - Encoder refers to the fact that it stacks encoder transformer blocks
  - Its most remarkable feature is creating representations that can be used for a very diverse set of downstream tasks

Bidirectional Encoding

- The transformer encoder reads the sequence all at once, making it bidirectional
- BERT trains the encoder as a language model by using an approach called Masked Language Model (MLM): 15% of tokens are sampled at random. Of those, 80% are replaced with the MASK token. 10% are replaced with a random word, and 10% are kept the same
- The model is trained by predicting these 15% of tokens. The numbers are a bit arbitrary, and there weren't experiments reported in the paper on this

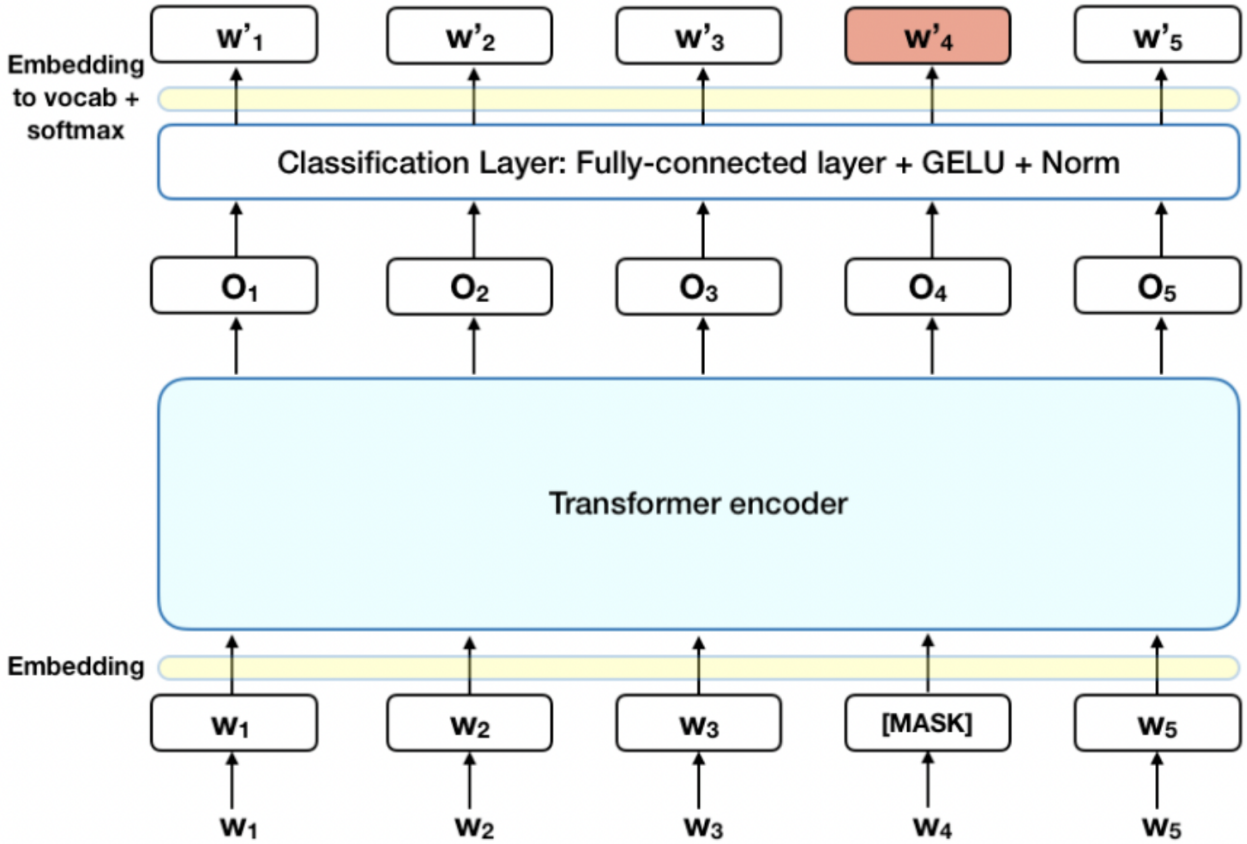
Masking



http://jalammar.github.io/illustrated-bert/

Masking

The model predicts all words, but only the masked ones are used to compute the loss



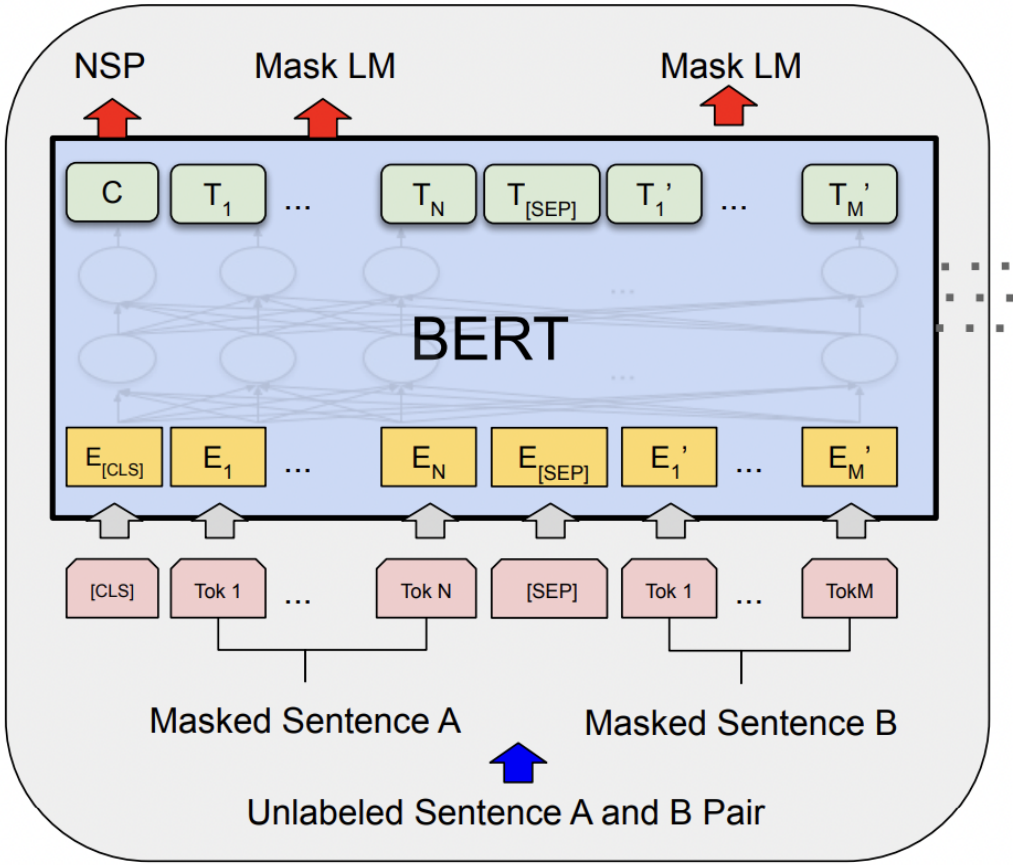
https://www.lyrn.ai

Next Sentence Prediction

- A second key part of the training objective is next sentence prediction
- The model receives pairs of sentences as inputs and predicts whether the second sentence comes directly after the first in the original corpus
- Half the time it is the next sentence, the other half the time it is chosen randomly

Next Sentence Prediction

It does this by inserting a special *CLS* token at the begin of the first sentence:

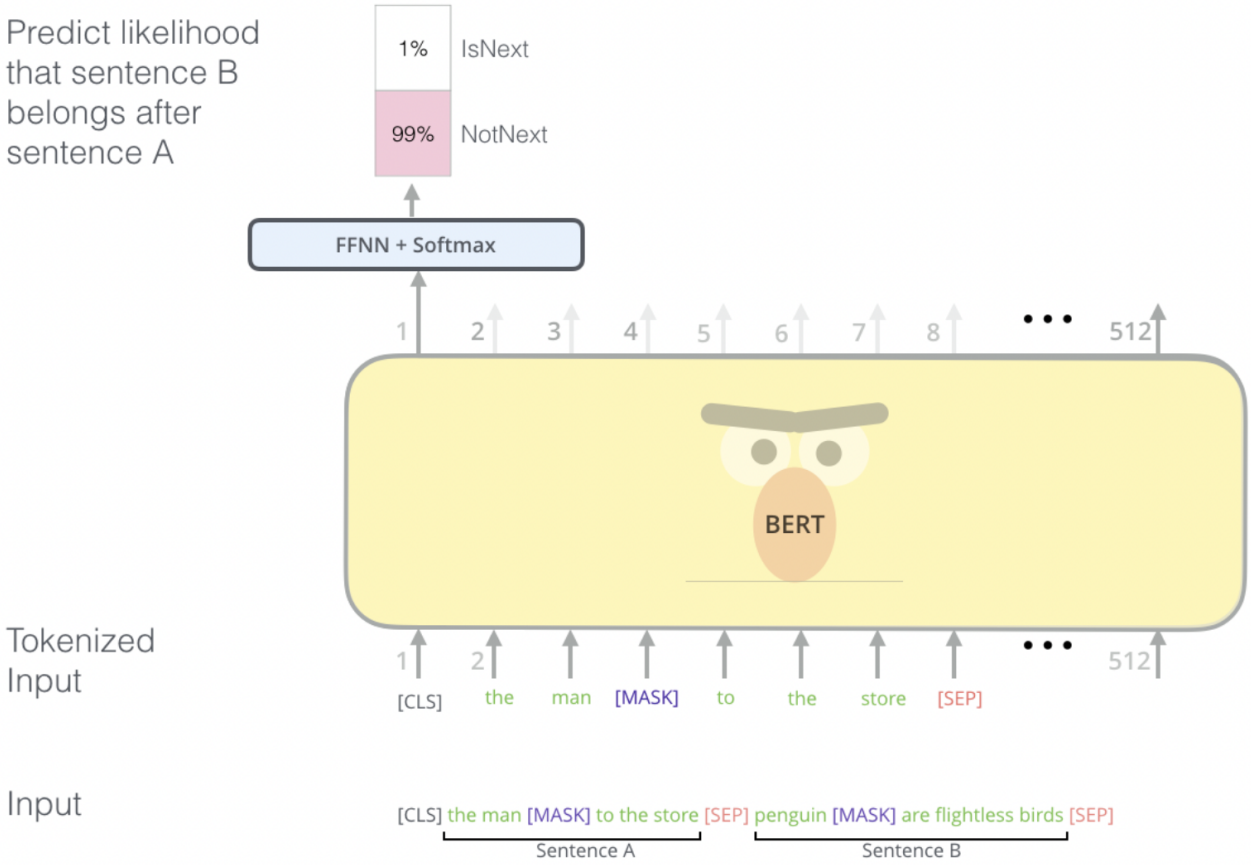


Pre-training

Devlin et al.

Next Sentence Prediction

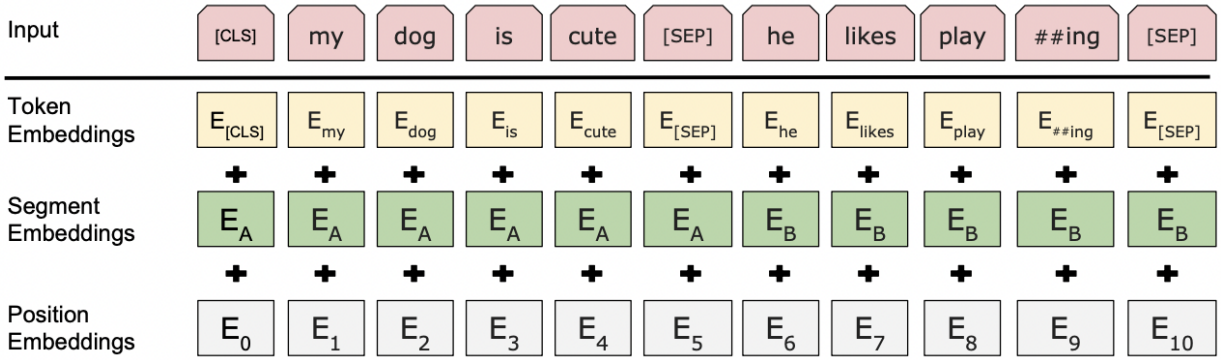
The class token connects with a FC layer to an output with two classes (is next/not next), rather than to a softmax for every word in the output vocabulary



http://jalammar.github.io/illustrated-bert/

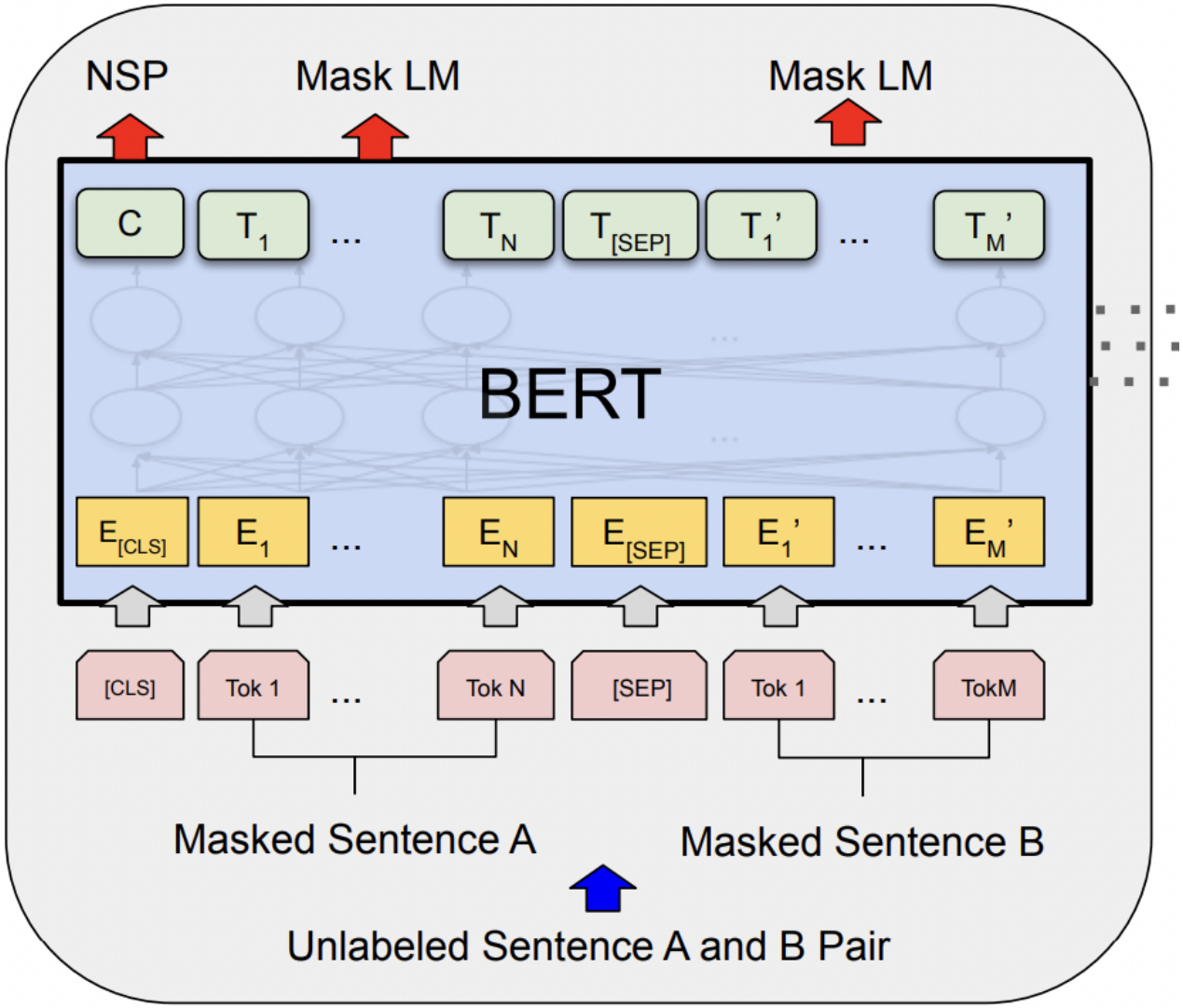
Training Input

Here is how tokens are input at training:



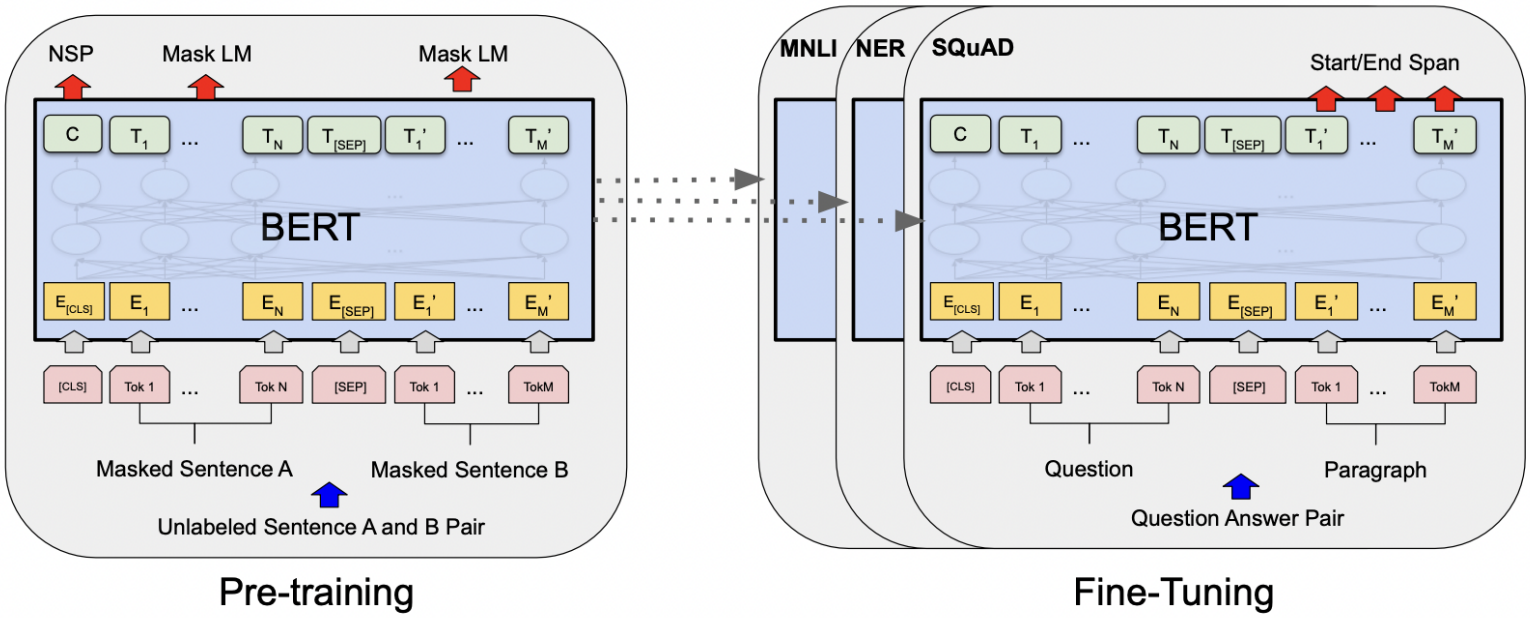
Devlin et al.

Recall the Training Architecture



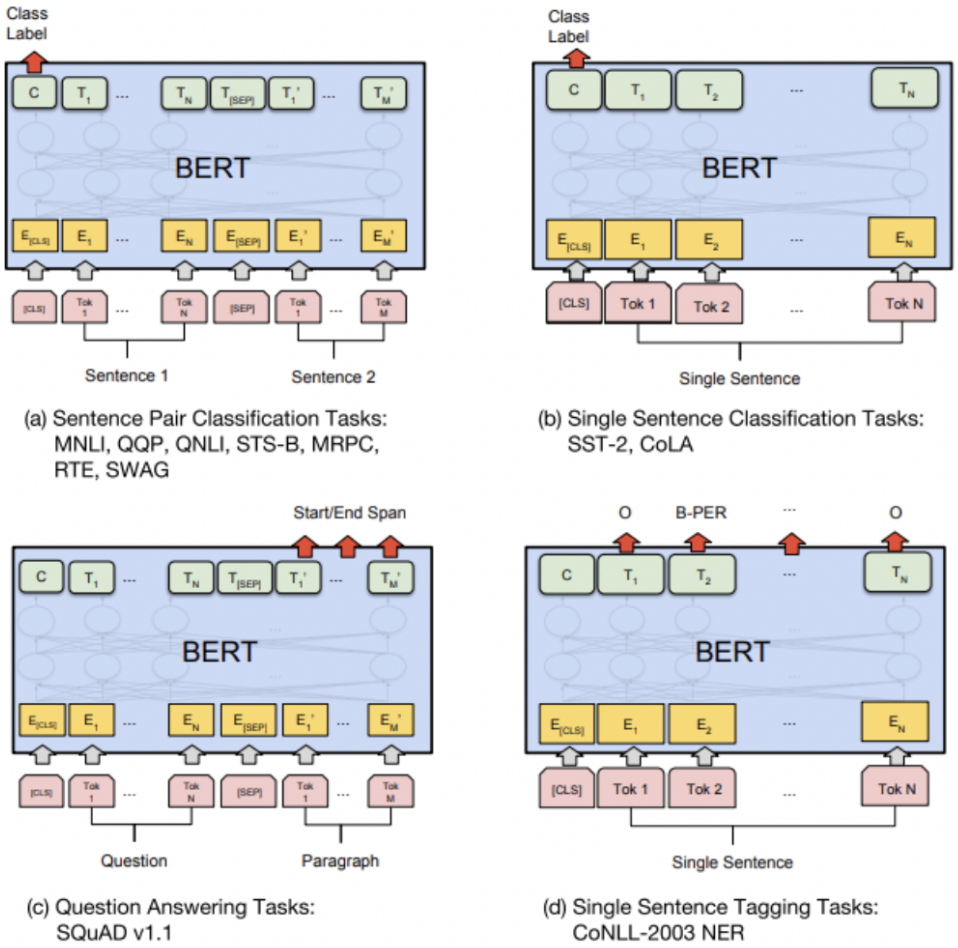
Pre-training

Adapt to Fine-Tuning on Specific Tasks



Devlin et al.

Adapt to Fine-Tuning on Specific Tasks

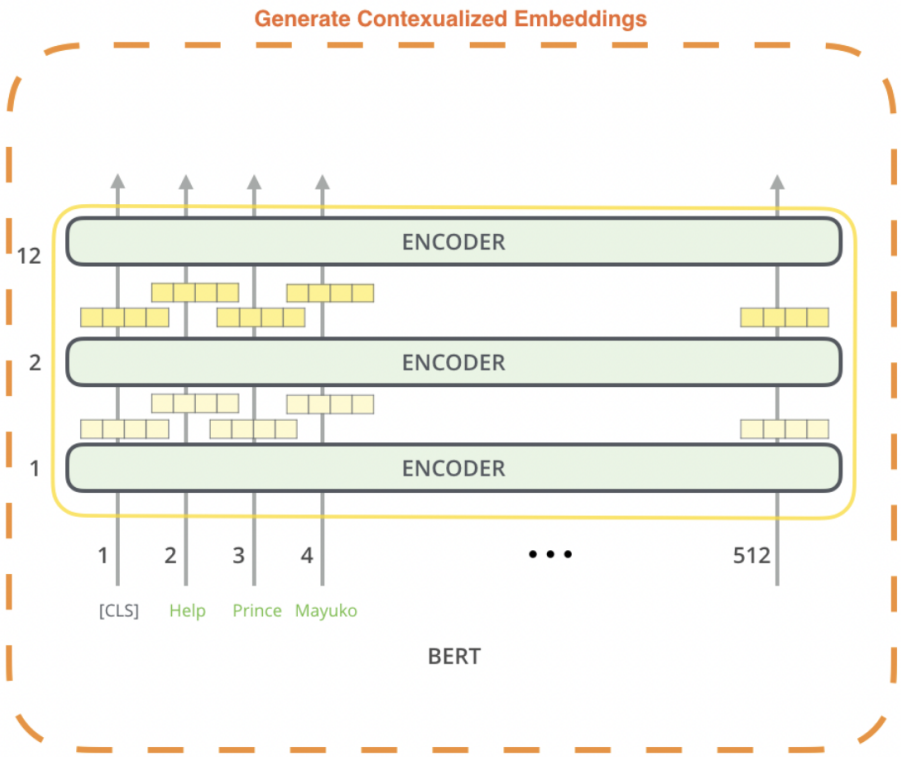


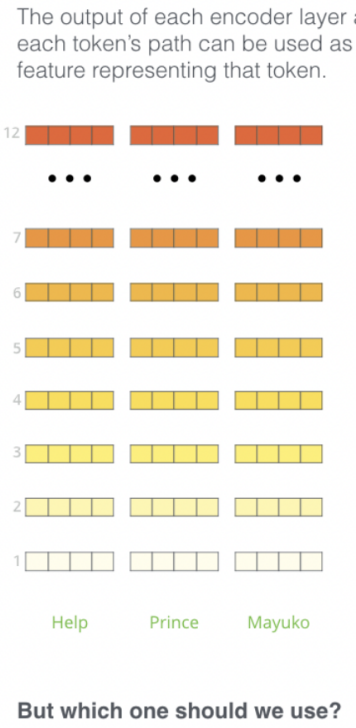
Devlin et al.

An Alternative to Fine-Tuning

- Take the pre-trained BERT embeddings
- Feed them into your own existing downstream model
- The paper shows, with an example from named entity recognition, that this works almost as well as fine-tuning
- Your mileage will probably vary

Which Layer to Use?

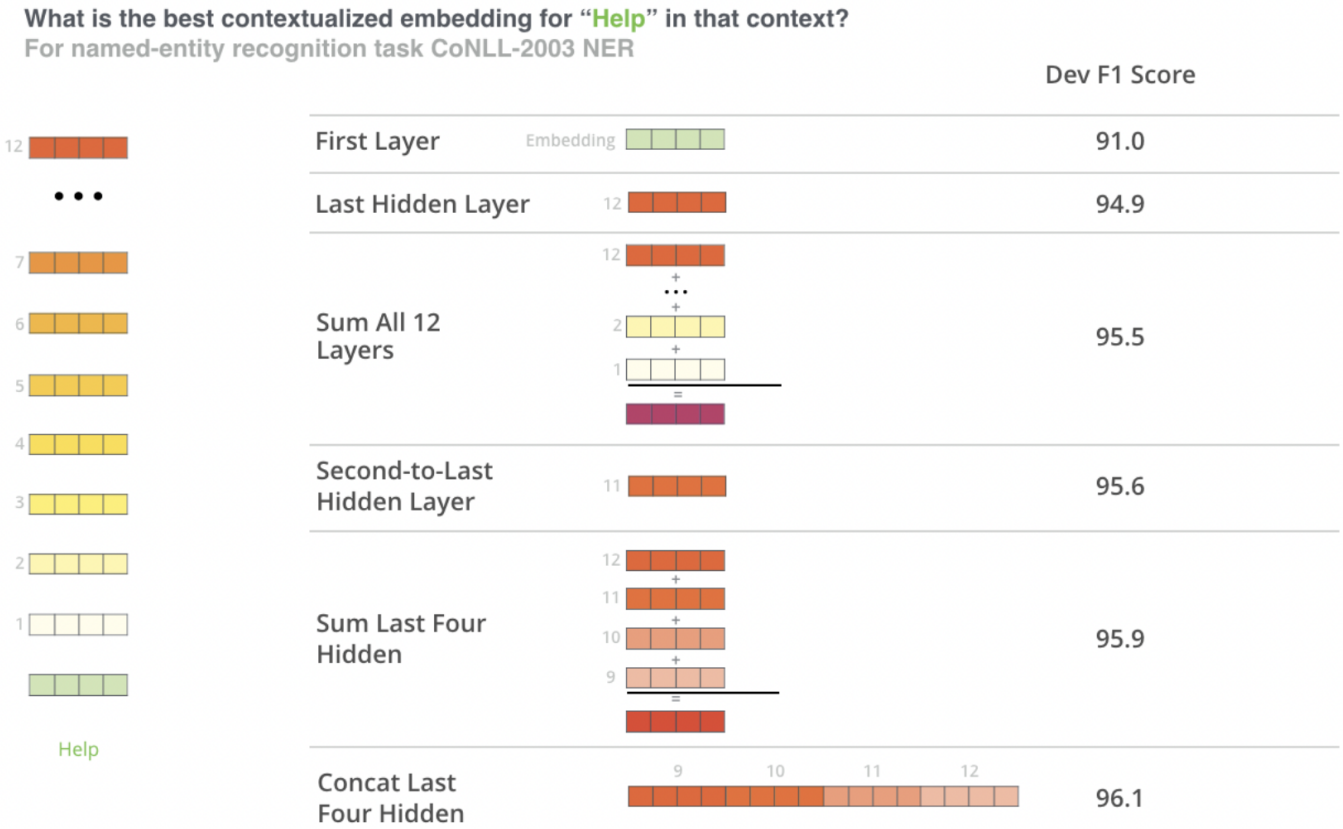




http://jalammar.github.io/illustrated-bert/

What Layer to Use

The fine-tuned model achieves a score of 96.4

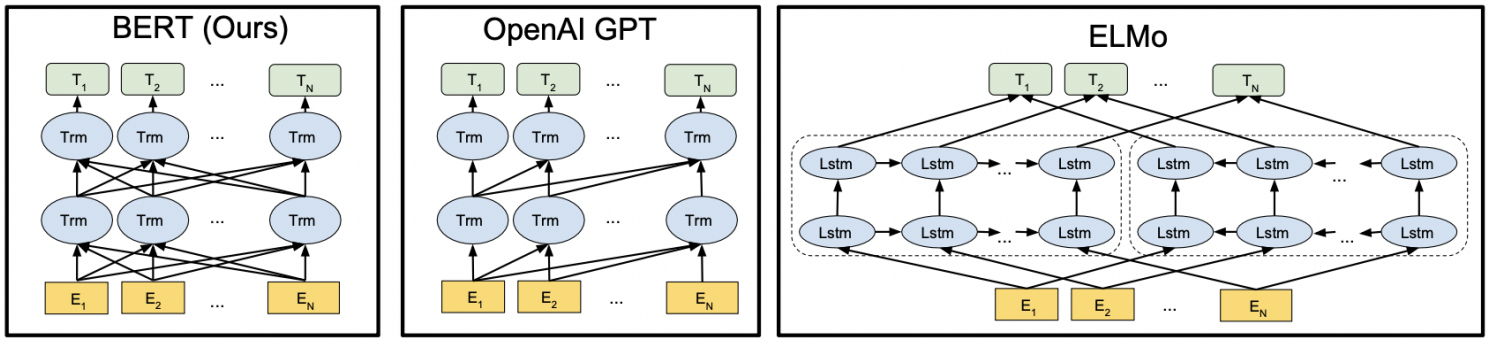


http://jalammar.github.io/illustrated-bert/

Which Layer to Use

- Likely depends on the task
- General consensus that upper layers are more task specific
- A paper "How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings" finds that representations of the same word in different contexts are less similar in the upper layers, suggesting that the upper layers produce more context specific representations

BERT Summary



Devlin et al.

BERT Options

- BERT Base: 110M parameters
- BERT Large: 340M parameters

Outline

Overview

Contextualized Word Embeddings

GPT

BERT

RoBERTa

DistilBERT

ALBERT

T5

GPT2/GPT3

Transformers XL

XLNet

Longformer

BigBird

Overview and What to Use

RoBERTa

- RoBERTa is produced by Facebook AI Research
- The **model** is the same as BERT
- Trains on way more data for longer with a larger batch size (theme in this literature of how to achieve SOTA)

RoBERTa

- BERT was trained on 16 GB data (books corpus and Wikipedia); RoBERTa adds 144GB of additional data
- Comes from the Common Crawl News Dataset (63 million articles; 76 GB), Web text corpus (38 GB), and Stories from Common Crawl (31 GB)
- Trained on 1024 Tesla v100 GPUs
- Results in a 2-20% improvement over BERT

RoBERTa Performance

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

RoBERTa Pre-Trained Models

Pre-trained models

Model	Description	# params	Download
`roberta.base`	RoBERTa using the BERT-base architecture	125M	roberta.base.tar.gz
`roberta.large`	RoBERTa using the BERT-large architecture	355M	roberta.large.tar.gz
`roberta.large.mnli`	`roberta.large` finetuned on MNLI	355M	roberta.large.mnli.tar.gz
`roberta.large.wsc`	`roberta.large` finetuned on WSC	355M	roberta.large.wsc.tar.gz

RoBERTa Pre-Trained Models

- A Winograd schema is a pair of sentences that differ in only one or two words and that contain an ambiguity that is resolved in opposite ways in the two sentences and requires the use of world knowledge and reasoning
  - "The city councilmen refused the demonstrators a permit because they [feared/advocated] violence."
  - If the word is "feared", then "they" presumably refers to the city council; if it is "advocated" then "they" presumably refers to the demonstrators.
- The Multi-Genre Natural Language Inference (MultiNLI) corpus is a crowd-sourced collection of 433k sentence pairs annotated with textual entailment information.

MultiNLI

Examples

Premise	Label	Hypothesis
Fiction		
The Old One always comforted Ca'daan, except today.	*neutral*	Ca'daan knew the Old One very well.
Letters		
Your gift is appreciated by each and every student who will benefit from your generosity.	*neutral*	Hundreds of students will benefit from your generosity.
Telephone Speech		
yes now you know if if everybody like in August when everybody's on vacation or something we can dress a little more casual or	*contradiction*	August is a black out month for vacations in the company.
9/11 Report		
At the other end of Pennsylvania Avenue, people began to line up for a White House tour.	*entailment*	People formed a line at the end of Pennsylvania Avenue.

Outline

Overview

Contextualized Word Embeddings

GPT

BERT

RoBERTa

DistilBERT

ALBERT

T5

GPT2/GPT3

Transformers XL

XLNet

Longformer

BigBird

Overview and What to Use

A Lightweight BERT

- While RoBERTa gives the best performance, you might instead want a lightweight model to reduce the computational burden
- DistilBERT has half the number of parameters of BERT, but retains 97% of the accuracy on benchmark tasks
- RoBERTa took 4 times as long to train as BERT, whereas DistilBERT trained 4 times faster
- Trained on the same data as BERT (16GB text, 3.3 billion words)

DistilBERT

- The basic idea is to approximate the larger network learned by BERT with a smaller network
- Distillation is a compression technique in which a small model is trained to reproduce a large model
- One of the key optimization functions used for posterior approximation in Bayesian Statistics is Kulback Leiber divergence, which is used here as well

Outline

Overview

Contextualized Word Embeddings

GPT

BERT

RoBERTa

DistilBERT

ALBERT

T5

GPT2/GPT3

Transformers XL

XLNet

Longformer

BigBird

Overview and What to Use

ALBERT

- Another lightweight option is “A Light BERT for Self-Supervised Learning of Language Representations”
- ALBERT shares parameters between transformer layers (reducing parameters by 70%)
- It also uses small embeddings for the input vectors, which are context independent (128 d versus 768 d for the hidden state representations); intuition is that it requires larger embeddings to learn context-dependent representation
- ALBERT base reduces parameters by 89%
- Can scale up the size of the hidden representations (while still sharing parameters); ALBERT XXL scales up the hidden representations to 4096, still has a 30% parameter reduction relative to BERT

ALBERT

- ALBERT XXL out-performed current state-of-the-art on some important benchmarks (2020)
- Smaller ALBERT did only modestly worse than BERT, with far fewer parameters

Outline

Overview

Contextualized Word Embeddings

GPT

BERT

RoBERTa

DistilBERT

ALBERT

T5

GPT2/GPT3

Transformers XL

XLNet

Longformer

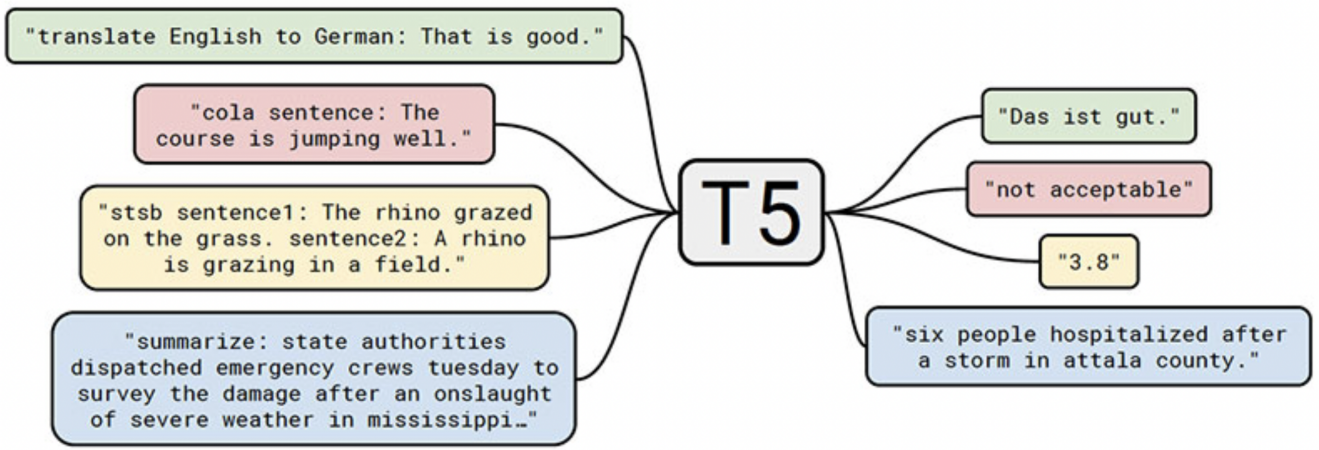
BigBird

Overview and What to Use

Going the Other Direction: T5

- A recent enormous model by Google: Text-to-Text Transformer
- Suggests every NLP model be treated as text-to-text: perspective on transfer learning
- Produces text as output for all tasks, even those like classification that don't normally have a text output (engineering convenience)
- The model understands what task is to be performed with a task specific prefix added to the original sentence (i.e. translate French to English, summarize, etc.)
- SOTA on more than 20 tasks; on many benchmarks, doing as well as humans

T5



T5

- Model trained on a new dataset called Colossal Cleaned Crawl Corpus
- 11 billion parameters
- **General theme**: achieve SOTA by training a very large NN on an enormous unlabeled dataset, and then fine-tune on small labeled data
- This entire literature has shown that with this general approach, can achieve far better performance than most NLP researchers thought possible

Problems

- The model is huge: thirty times larger than BERT
- Too much for commodity hardware to handle, would need to use Google's Cloud TPU platform
- Will likely be smaller versions, akin to ALBERT

Problems

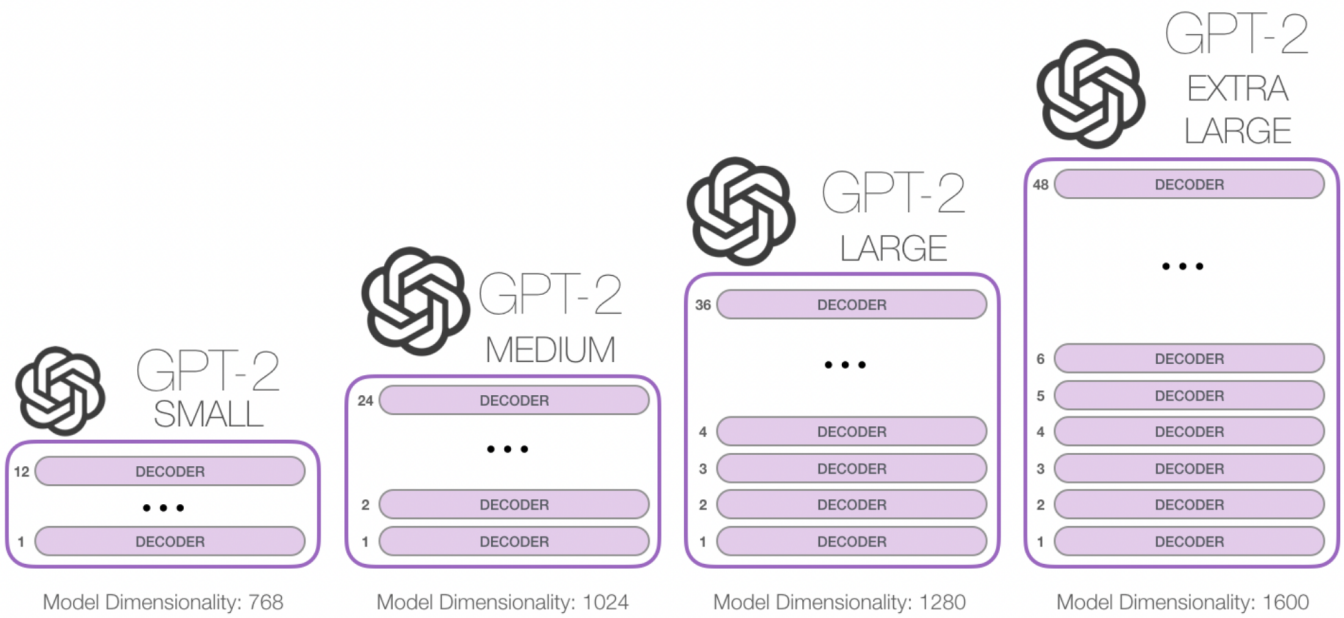
- Pushing human performance on benchmarks, but if you work with these systems, know that they can often fail in unhuman-like ways; you would know if you were talking with a T5 chatbot versus a human customer service representative
- Language understanding is not solved. Suggests we need more challenging and realistic text datasets (far from obvious ex ante how well the off-the-shelf model would perform on our NLP tasks)
- Biases: Winogender Schemas show that can often correctly classify sentences in stereotypical situations (i.e. male doctor, female nurse) but not vice versa

Outline

GPT2/GPT3

- Speaking of large models, we would be remiss not to mention Open AI's GPT2/GPT3, the successors to the GPT model we saw earlier
- If you understand how the decoder transformer blocks work, you understand GPT2/3 at a general level
- Since it is a language model that generates text one word at a time, it is good for text generation tasks: give it a prompt and it will continue the text
- Unlike the other models we are discussing, it is proprietary

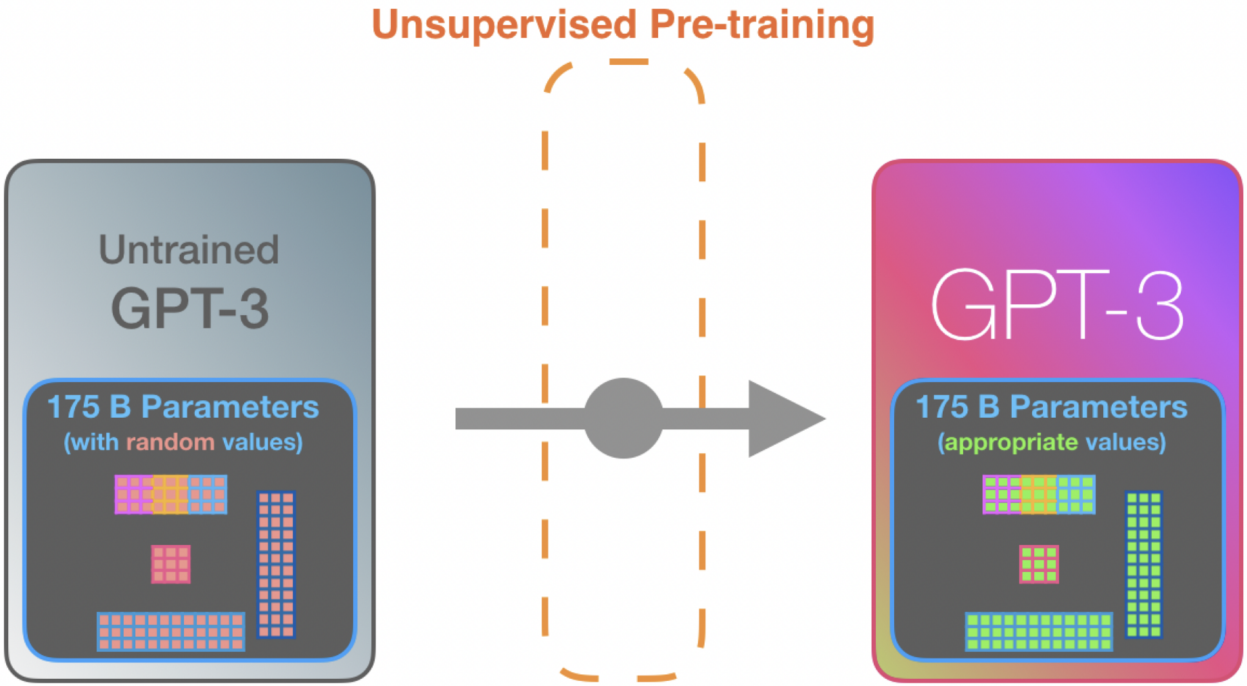
GPT2



https://jalammar.github.io/illustrated-gpt2/

GPT3

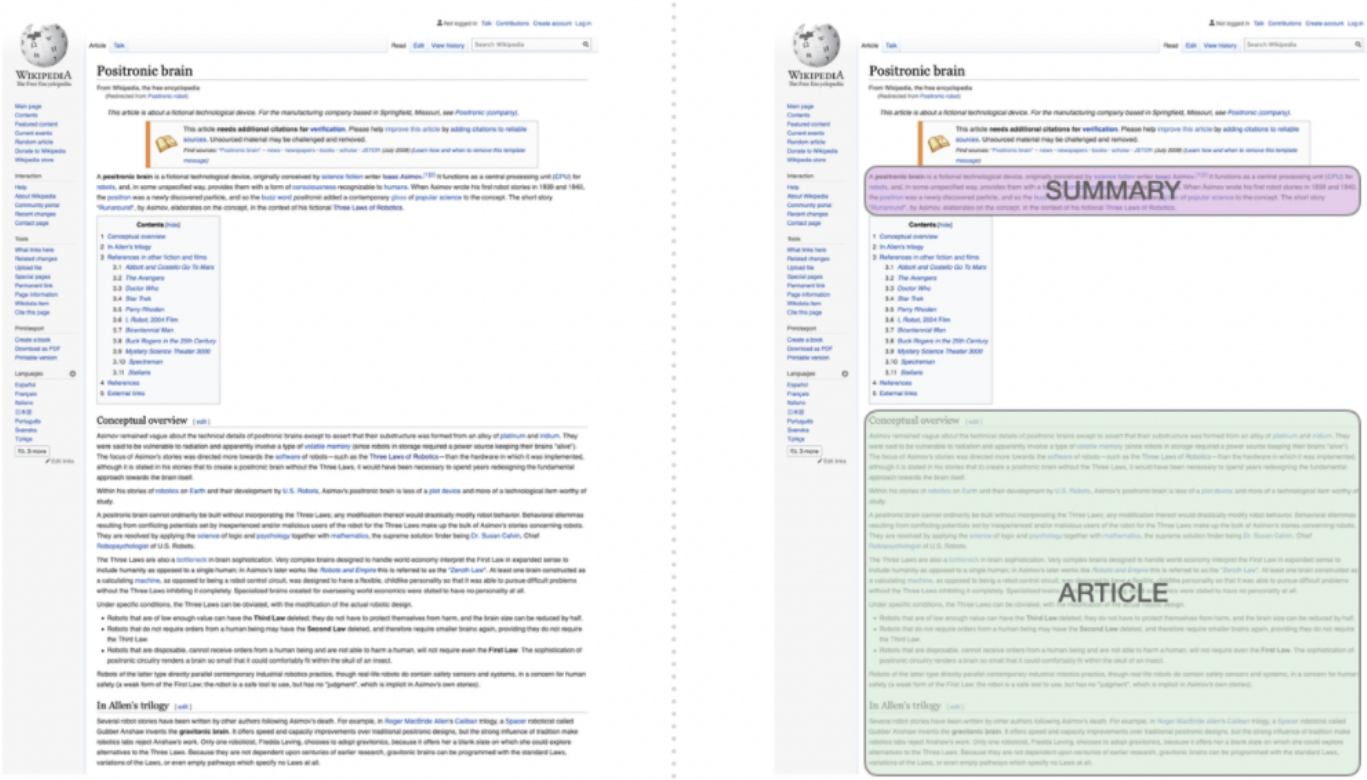
GPT3 blew this out of the water with 175B parameters



https://jalammar.github.io/how-gpt3-works-visualizations-animations/

Context window is 2048 tokens and 96 layers deep. 355 GPU years to train, at a cost of $4.6 million

Also Trained for Downstream Tasks Like Summarization

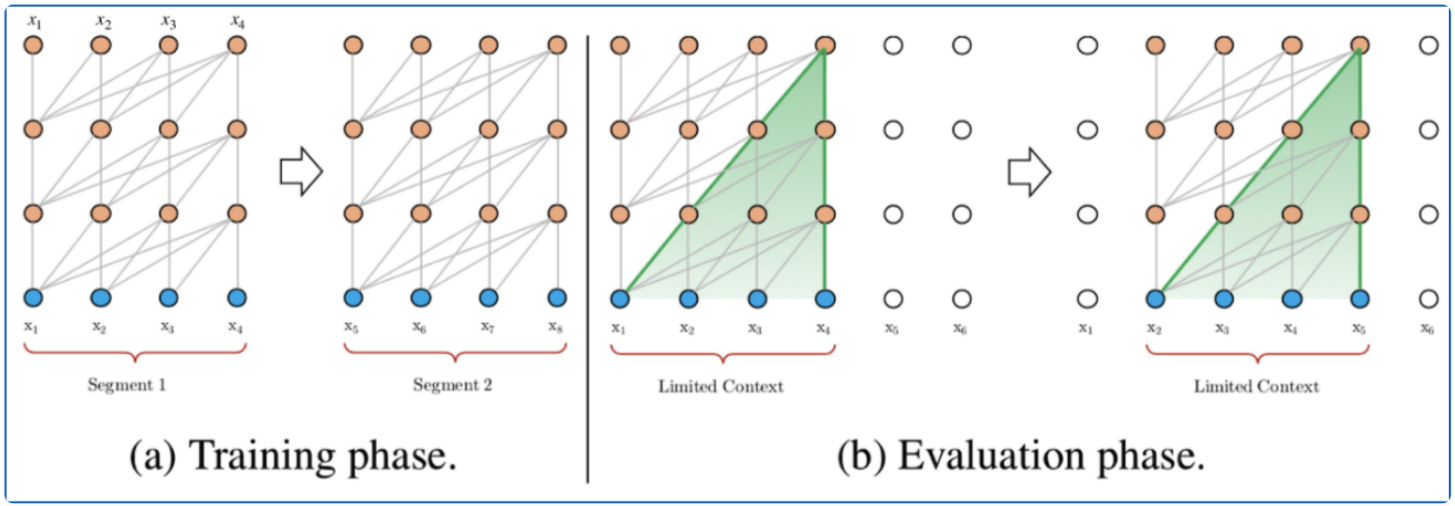


Outline

Attention

- Computing attention is $o(n^2)$ in the length of the input sequence
- BERT limits the length of input sequences to 512
- There is no way to attend across these segments

Standard Attention Computation



Training and Evaluation of the vanilla Transformer language model. Source: Transformer-XL

Transformer-XL

Introduces two features to overcome the above shortcoming:

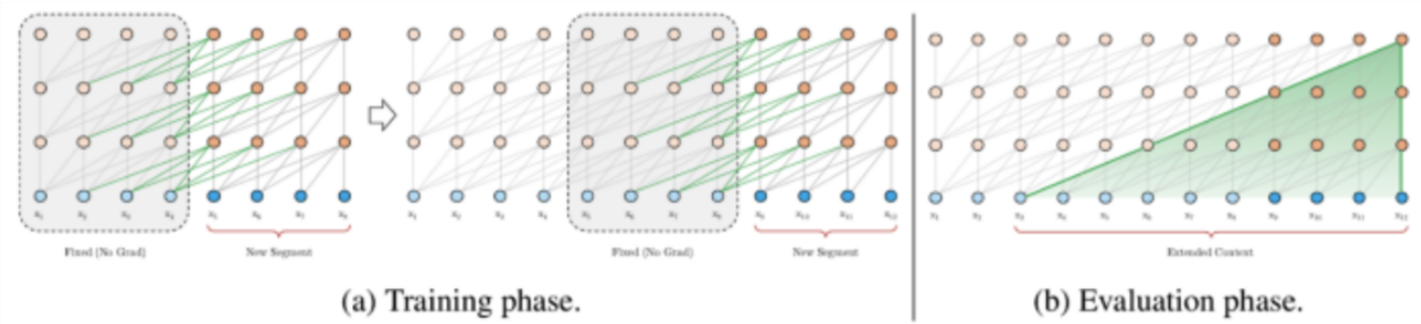
- Recurrence Mechanism
- Relative Positional Encoding

Recurrence Mechanism

To integrate long term dependencies, each hidden layer receives two inputs

- The output of the previous hidden layer of that segment, as in Bert
- The output of the previous hidden layer from the previous segment to create long-term dependencies.

Recurrence Mechanism



Training and Evaluation of the Transformer-XL language model. Source: Transformer-XL

The two inputs are concatenated and then used to calculate the Key and the Value matrices of the current Head of the current layer of the current segment

Relative Positional Encoding

- The original positional encoding handles each segment separately and, as a result, tokens from different segments have the same positional encoding.
- For example, the first token of the first and the second segments will have the same encoding, although their position and importance are different
- The paper presents a new positional encoding that is part of each attention module, as opposed to encoding position only before the first layer, and is based on the relative distance between tokens and not their absolute position (incorporates several learned vectors)

Outline

XLNet

- XLNet uses TransformerXL, with its main contribution being a novel training objective
- BERT and ELMo improved on SOTA by incorporating left and right contexts
- XLNet takes this further by predicting each word in a sequence using any combination of other words in the sequence
- Presented difficult, and at times ambiguous contexts from which to infer whether or not a word is in a sentence; intuition is that this allows it to squeeze more info from the training data

XLNet

- An autoregressive language model predicts $P(x_i|x_{<i})$; ELMo also looks at $P(x_i|x_{>i})$
- Consider a sequence with T=4 tokens. Now consider the set of all 4! permutations $Z = [1,2,3,4],[1,2,4,3],...,[4,3,2,1]$. The XLNet model is auto-regressive over all such permutations
- Samples from all possible permutations, since there are many

Example

- Consider the permutation [3, 2, 4, 1].
- When calculating the probability of the 1st element in that order (i.e., token 3), the model has no context as the other tokens have not yet been seen.
- So the mask would be [0, 0, 0, 0].
- For the 2nd element (token 2), the mask is [0, 0, 1, 0] as its only context is token 3.
- Following that logic, the 3rd and 4th elements (tokens 4 and 1) have masks [0, 1, 1, 0] and [0, 1, 1, 1].
- Stacking all those in the token order gives the attention matrix

Takes A Long Time to Train

ELMo	40 GPU days
BERT	450 GPU days
XLNet	2000 GPU days

Takes A Long Time to Train

	BERT	RoBERTa	DistilBERT	XLNet
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base:** 66	**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*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base:** 8 x V100 x 3.5 days; 4 times less than BERT.	**Large:** 512 TPU Chips x 2.5 days; 5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT	2-15% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	**Base**: 16 GB BERT data **Large**: 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

Outline

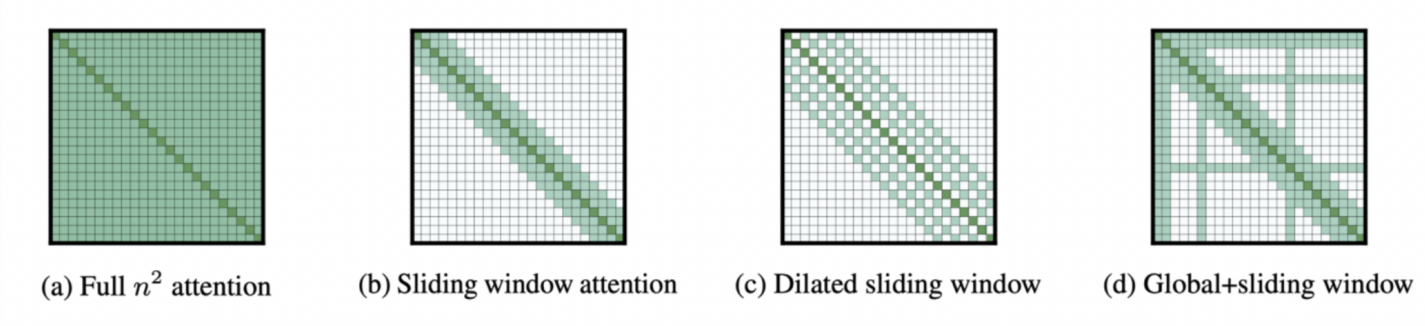
Longformer

- To process a sequence of n tokens requires n^2 attention calculations for each attention head during the forward pass
- BERT addresses this by enforcing a hard limit of 512 (sub-word) tokens, more than enough to process the overwhelming majority of sequences in most benchmark datasets
- However, this is not sufficient if you need to attend within longer texts, like entire documents
- Still know the original word order through the Transformer's positional encodings

Basic Idea of Longformer

- Use an attention sliding window of size w
- This allows a multilayer transformer model to have a receptive field that covers the entire document
- Same intuition as the receptive field within CNNs, as convolutions are sliding windows

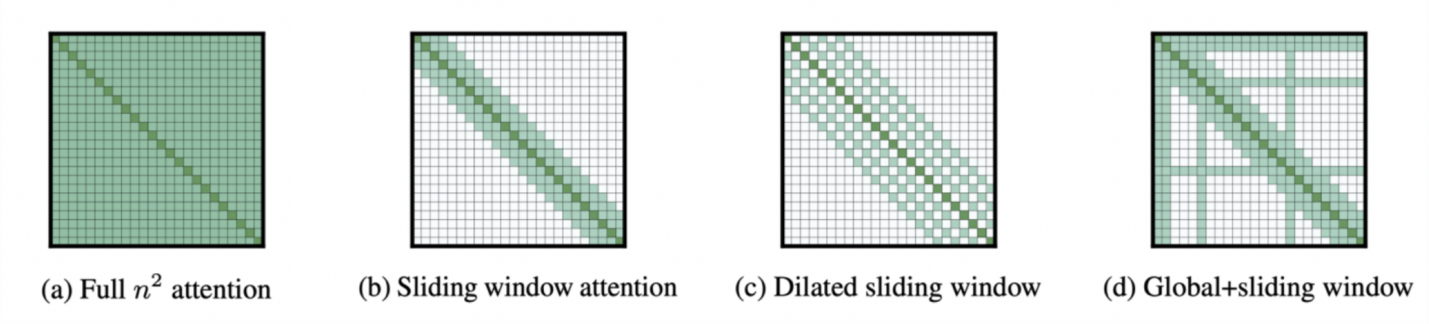
Longformer



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Dilated sliding window allows for an even larger receptive field

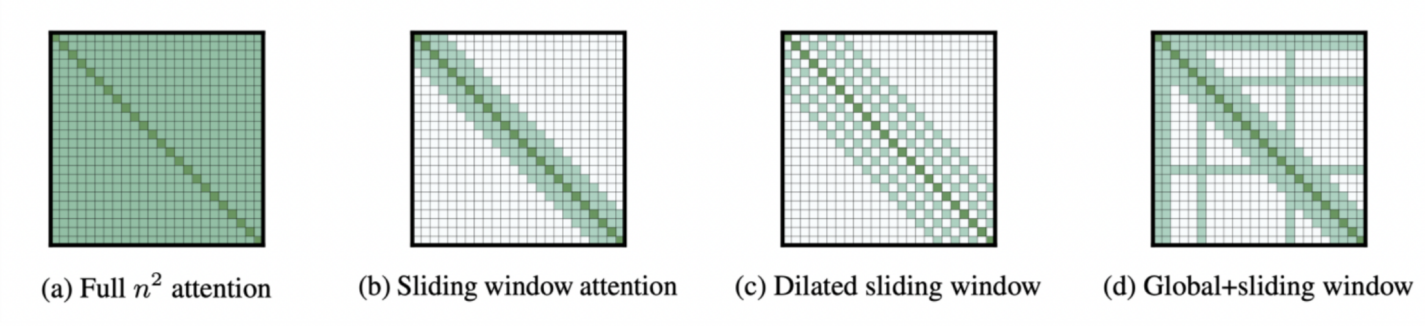
Longformer



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

The CLS token is supposed to be able to aggregate the entire sequence into a single representation to allow for classification.

Longformer



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

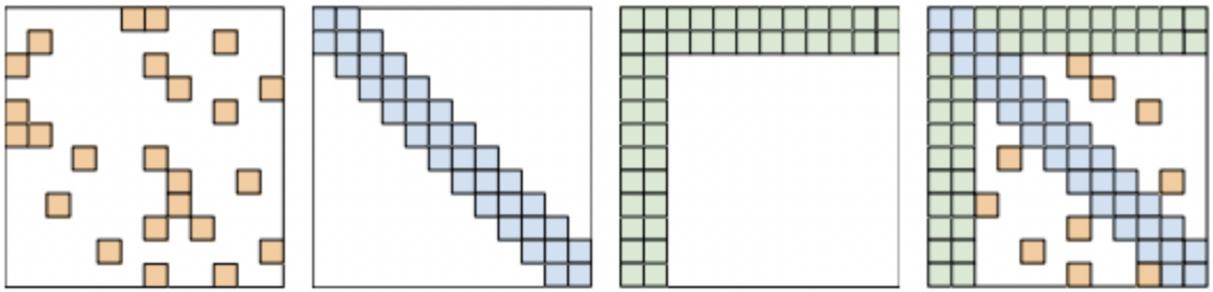
Task-specific, global attention on special tokens. This attention is symmetric in that every token in the sequence can attend to the special token, as it can attend all of them.

Outline

BigBird: Transformers for Long Text

- BigBird is an even more recent model by Google that has a similar intuition
- If we want to attend throughout long text, we need a way around exponential global attention
- Can process sequences 8x longer than BERT, because attention is o(n)

BigBird



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 1: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with $r = 2$, (b) sliding window attention with $w = 3$ (c) global attention with $g = 2$. (d) the combined BIGBIRD model.

Outline

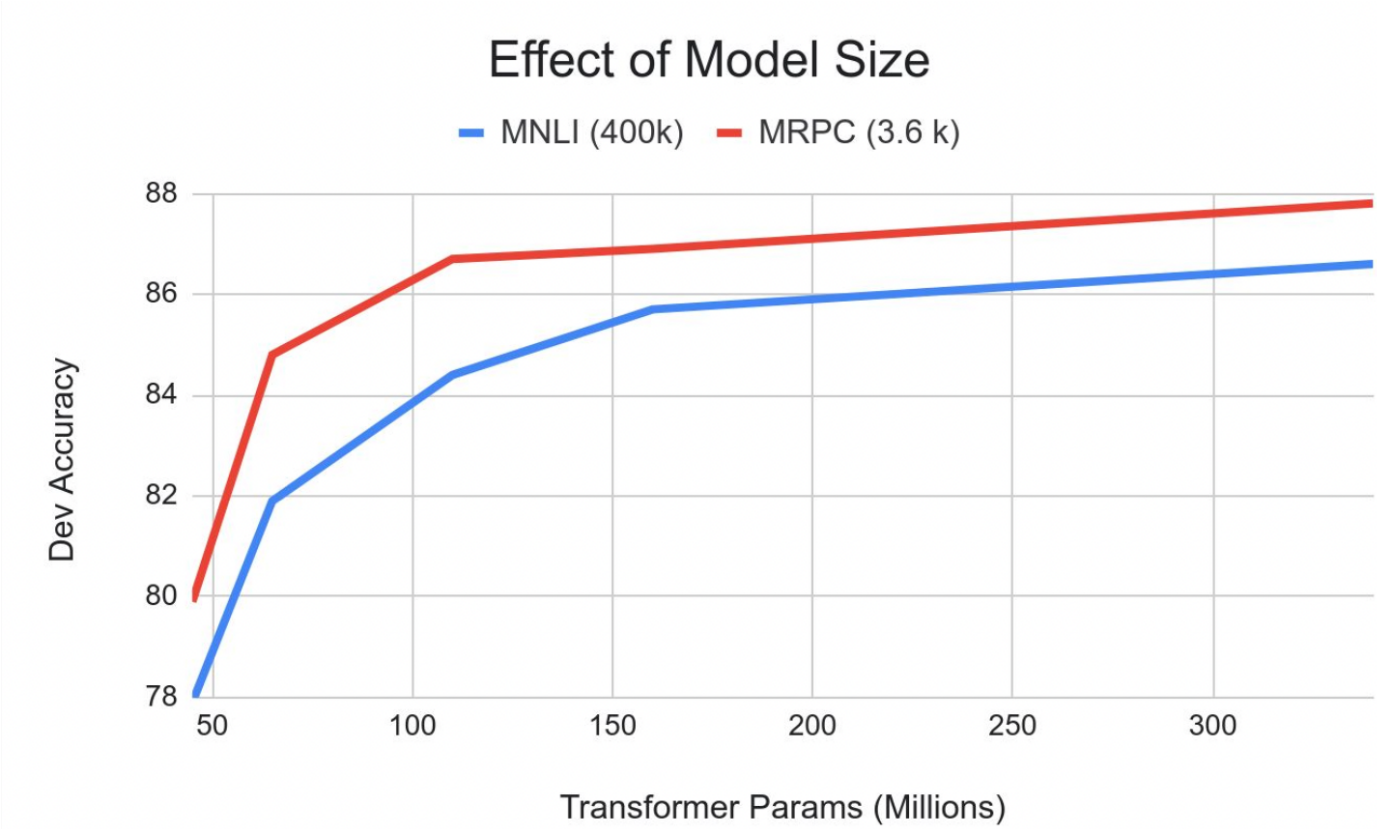
Why Have Transformers Come to Dominate Language Modeling?

1. Self-attention removes locality bias
   - By design, an LSTM pays the most attention to the closest contexts
   - As long as it is within the sequence window, long-distance context has equal opportunity to be important in the transformer
2. The entire input can be processed in parallel, whereas an LSTM requires sequential processing
   - The parallel compute power of GPUs can be fully harnessed, allowing for massively larger models to be trained

Key Theme: Bigger is Better

- One of the clearest lessons to emerge from this literature is that bigger models, trained on more data for longer, achieve SoTA, on a wide variety of downstream tasks
- Performance gains have not yet asymptoted, even when looking at very large models (i.e. T5 has 11 B parameters)
- This is true even when fine-tuning models to tasks with a small number of labels

Bigger is Better



Remarkable the degree to which DL architectures have been able to solve overfitting and vanishing gradient problems

Bigger is Better

T5 did a bunch of ablation studies to understand which features matter, i.e.

- Model size
- Amount of training data
- Domain/noise in training data
- Pre-training objectives
- Ensembling
- Details of fine-tuning
- Multi-task training

Bigger is Better

- The only things that made much of a difference were model size and amount of training data
- Best model has 11B parameters (BERT large has 330M) trained on 120B words of cleaned Common Crawl Text
- ALBERT is light in terms of parameters, not speed of pre-training. The version that beats BERT has fewer parameters but is trained for significantly longer

Bigger is Better

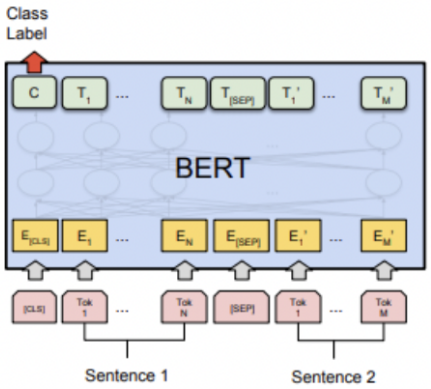
Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

Being light in terms of parameters is still very beneficial in terms of being able to deploy or fine-tune it on your local machine

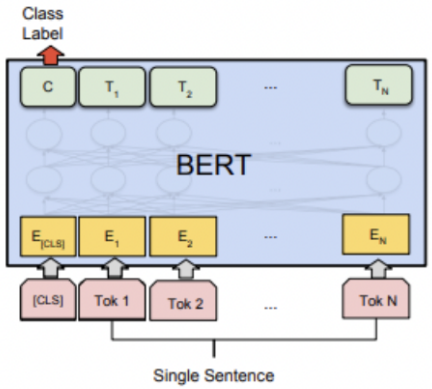
Language Modeling and Downstream Tasks

- Before the past few years, it was common to design different customized architectures for different types of downstream language tasks. These could be highly specialized to particular tasks
- The Transformer-based language models obviated this, providing a way to fine-tune most downstream tasks you'd want to do within the architecture of the language model (other models like ELMO had a similar idea, but post-BERT we've really seen this take off)
- This has made them enormously influential, underscoring just how much performance on downstream tasks is about having good representations (achieved through training massive models on massive data), versus about engineered domain knowledge

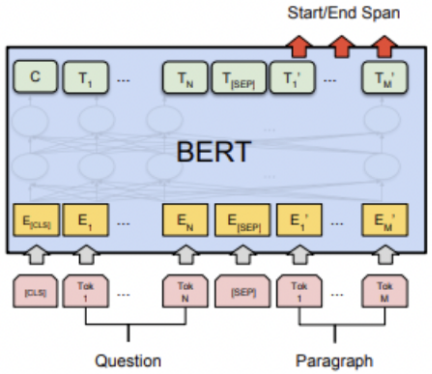
BERT and Fine-Tuning



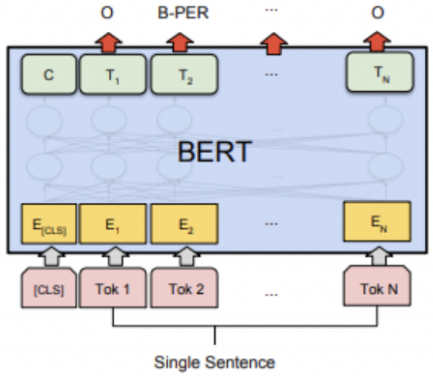
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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

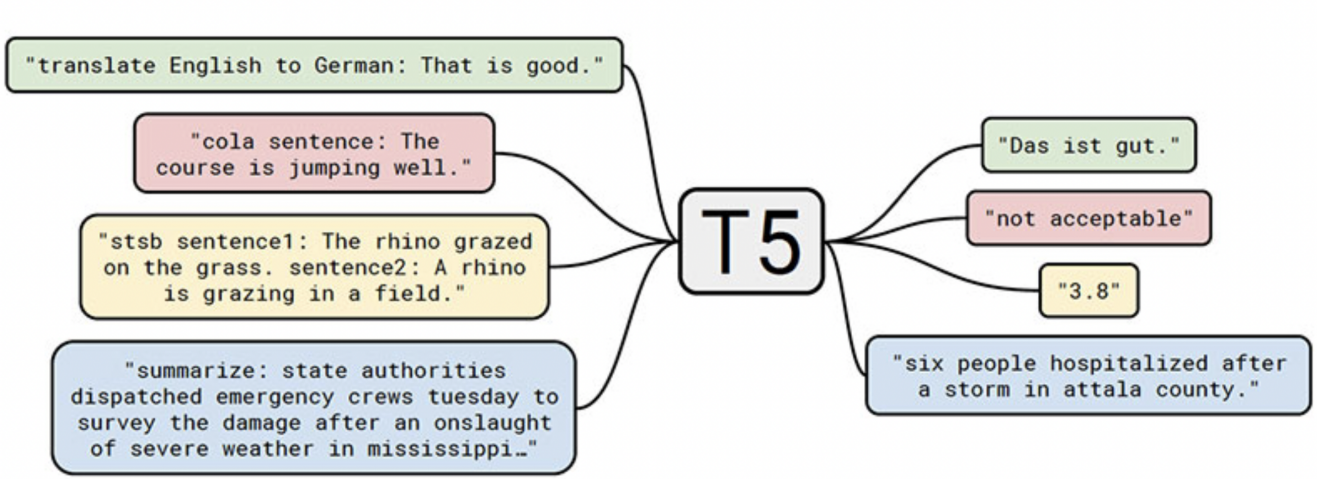


(c) Question Answering Tasks: SQuAD v1.1



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

T5: The Limits of Transfer Learning



Summary

- Bottom line: pre-trained language models work extremely well for downstream tasks
- The main thing that makes them work better is building bigger models trained for longer on more data. This is very expensive
- Fortunately, Google/Facebook/Microsoft have a lot of resources and a business incentive to pre-train them and have for the most part open-sourced their innovations (Open AI/Microsoft excepted)
- Distillation and parameter sharing (ALBERT) make it possible to do inference and potentially fine-tune on commercial hardware
- These models remove the need to design lots of customized architectures for downstream tasks

What to Use

- Different models have different architectures, training objectives, and training data
- Papers will say something about differences with other models and what matters but far from comprehensive (expensive to do a bunch of ablations)
- Hence, the influence of these different factors - which will matter differentially for different downstream tasks - is conflated
- You need to experiment (i.e. with pre-trained embeddings)

What to Use?

- Is there a pre-trained version that approximates your task (i.e. RoBERTa News; BERTweet)
- ALBERT, RoBERTa, or XLNet are good places to start. If you are heavily computationally constrained, try inference with one of the lightweight ALBERT models
- Long text/documents: BigBird or perhaps Longformer
- This literature moves fast