

# Transformers and Pre-trained Language Models

Reza Shirani

# Outline

- Transformers
  - RNNs vs. Transformers
  - Transformer architecture
    - Self-attention
    - Attention Visualization
- Pretrained language models
  - Pre-training general language representations
  - Transformer-based models
  - How to compare? (GLUE benchmark)
  - How to improve?
  - State-of-the-art transformer-based models
- How to use? (Notebook)

# Transformer

## Attention Is All You Need (Vaswani et. al, 2017)

**IDEA:** Get rid of CNNs and RNNs:  
“Just use attention”

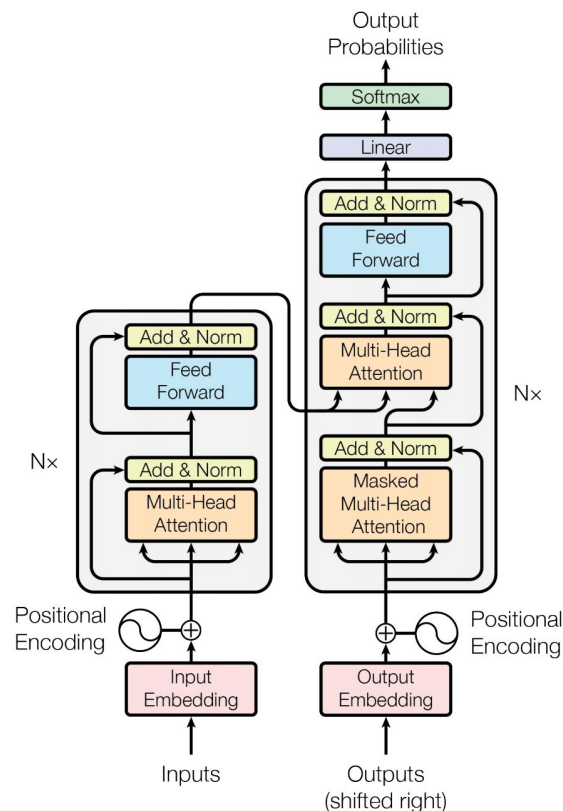
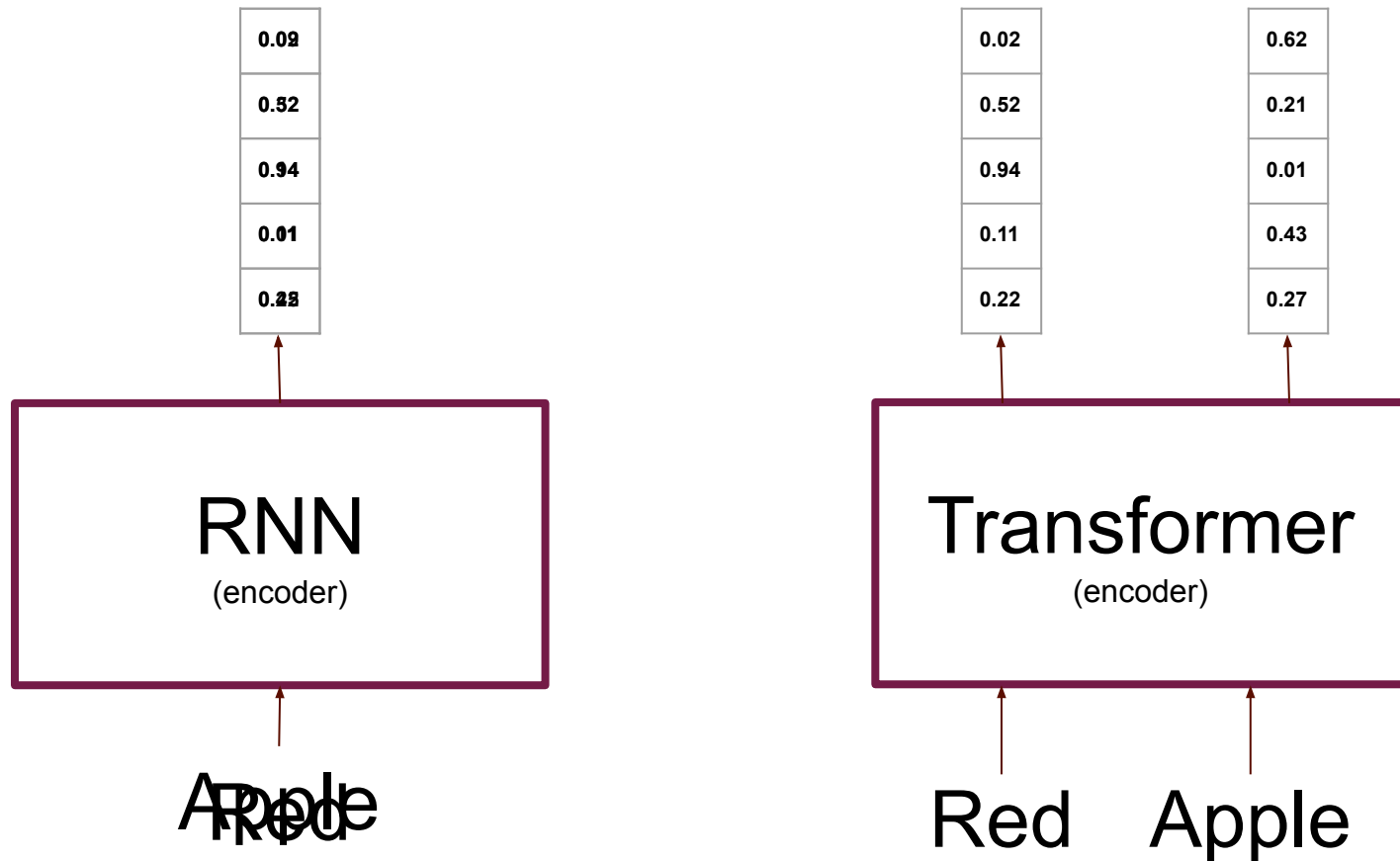


Figure 1: The Transformer - model architecture.

# RNNs vs. Transformers



# RNNs vs. Transformers

## Challenges with RNNs

- Long range dependencies
- Gradient vanishing and explosion
- Large number of training steps
- Recurrence prevents Parallel computation

## Transformer Networks

- Facilitate long-distance dependencies
- No Gradient vanishing and explosion
- Fewer training steps
- Parallel computation

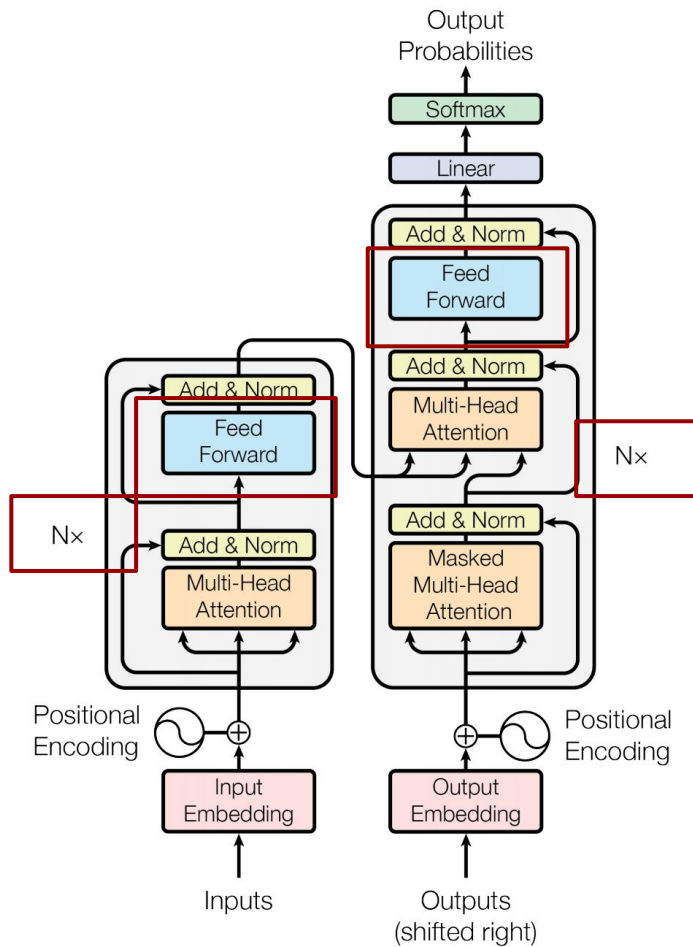
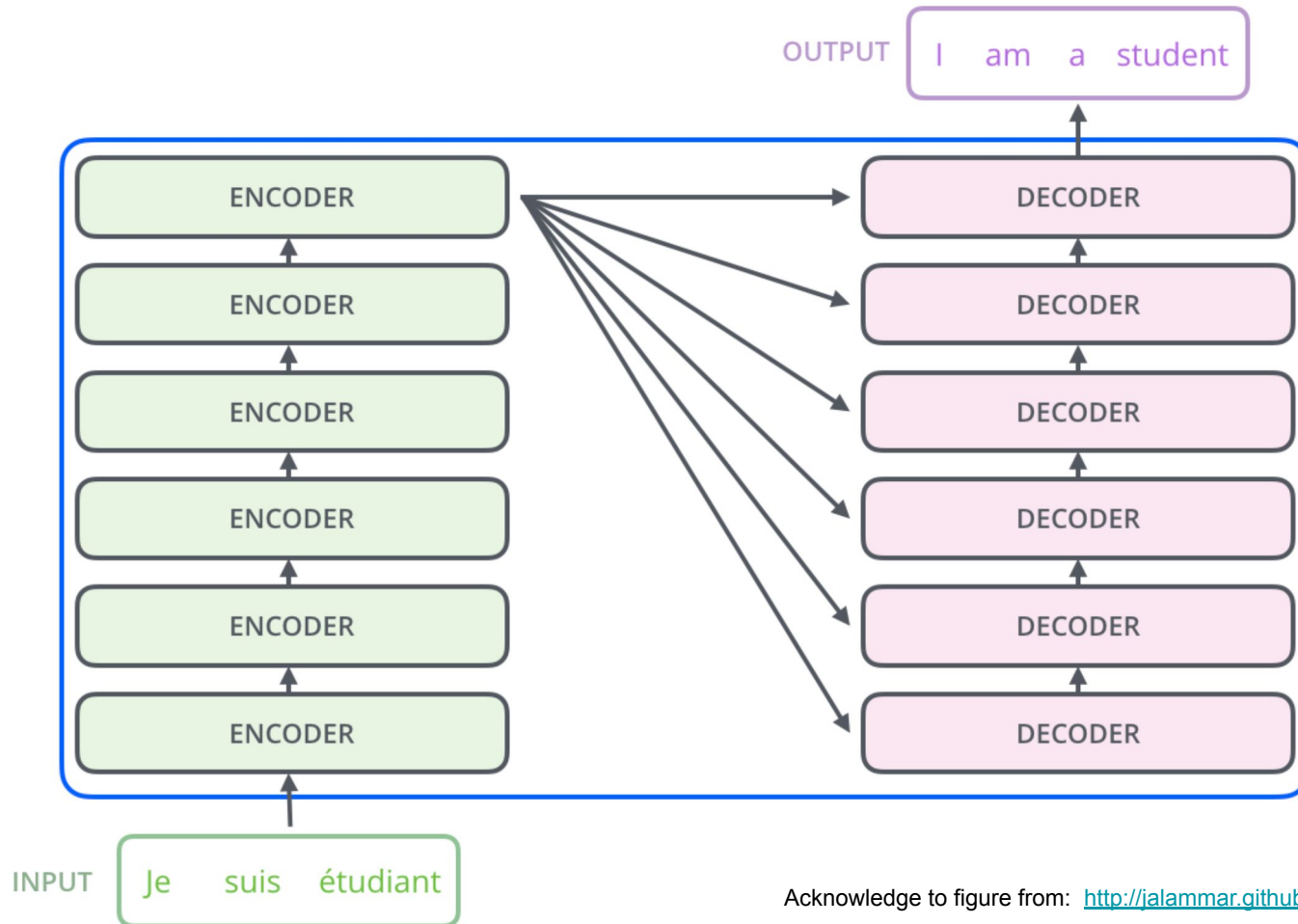


Figure 1: The Transformer - model architecture.



Acknowledge to figure from: <http://jalamar.github.io/illustrated-transformer/>

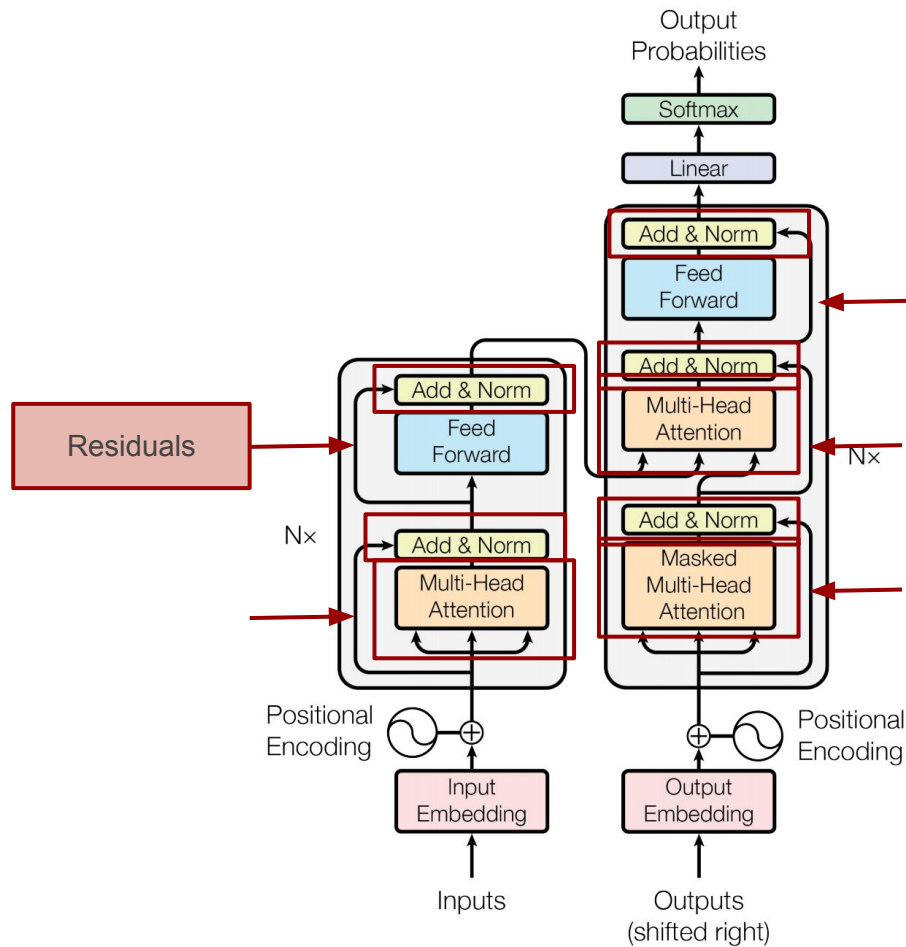
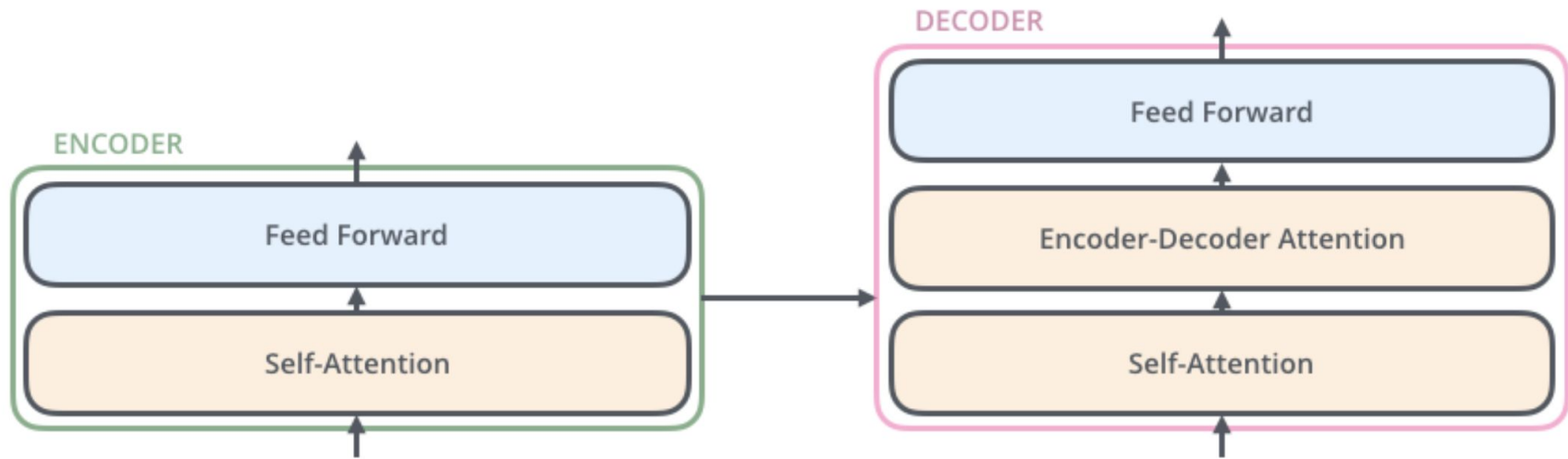


Figure 1: The Transformer - model architecture.

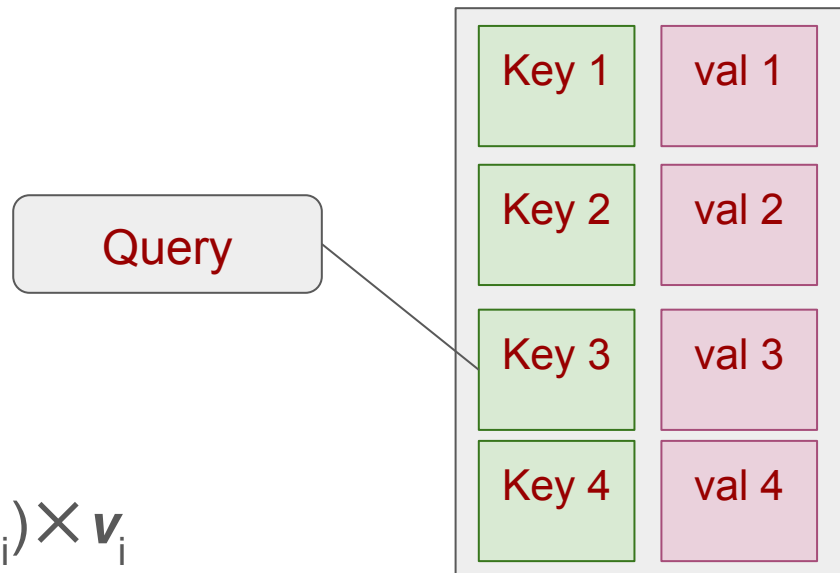




Acknowledge to figure from: <http://jalammar.github.io/illustrated-transformer/>

# Self-attention

Mimics the retrieval of the **Value**  $v_i$ , for a **Query**  $q$  based on **Key**  $k_i$  in database.



$$\text{attention}(q, k, v) = \sum_i \text{similarity}(q, k_i) \times v_i$$

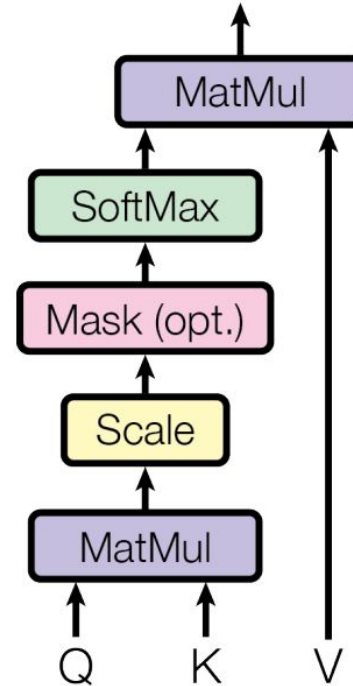
# Self-attention

Scaled dot Product:

**Problem:** the scale of dot product increases as dimensions get larger

**Fix:** scale by the size of the vector

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

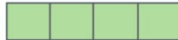


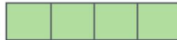
Input

Thinking

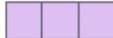
Machines

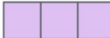
Embedding

$x_1$  

$x_2$  

Queries

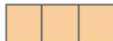
$q_1$  

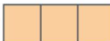
$q_2$  



$W^Q$

Keys

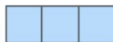
$k_1$  

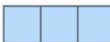
$k_2$  



$W^K$

Values

$v_1$  

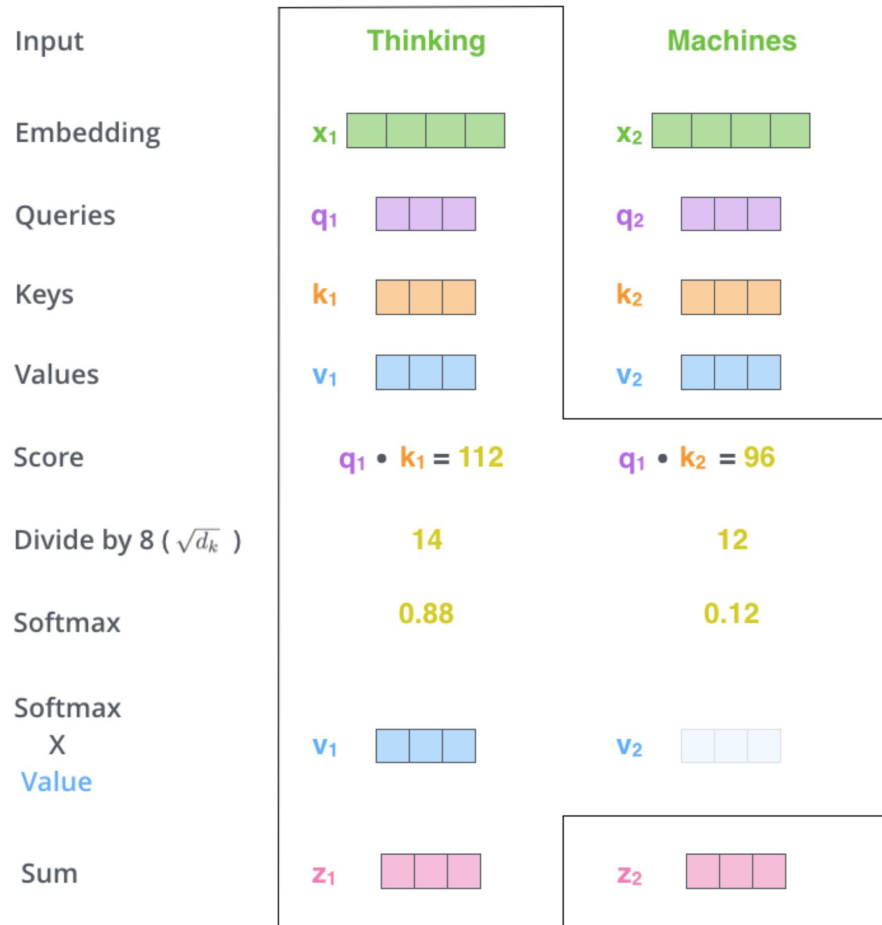
$v_2$  



$W^V$

Multiplying  $x_1$  by the  $W^Q$  weight matrix produces  $q_1$ , the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

Acknowledge to figure from: <http://jalamar.github.io/illustrated-transformer/>



Acknowledge to figure from: <http://jalammar.github.io/illustrated-transformer/>

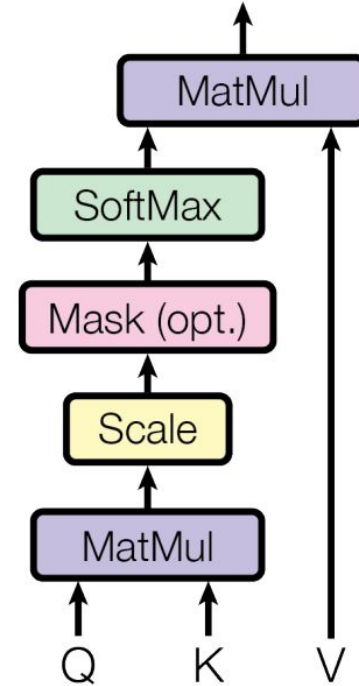
# Scaled Dot-Product Attention

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V$$

=

$Z$

The self-attention calculation in matrix form



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

# “Multi-headed” Attention

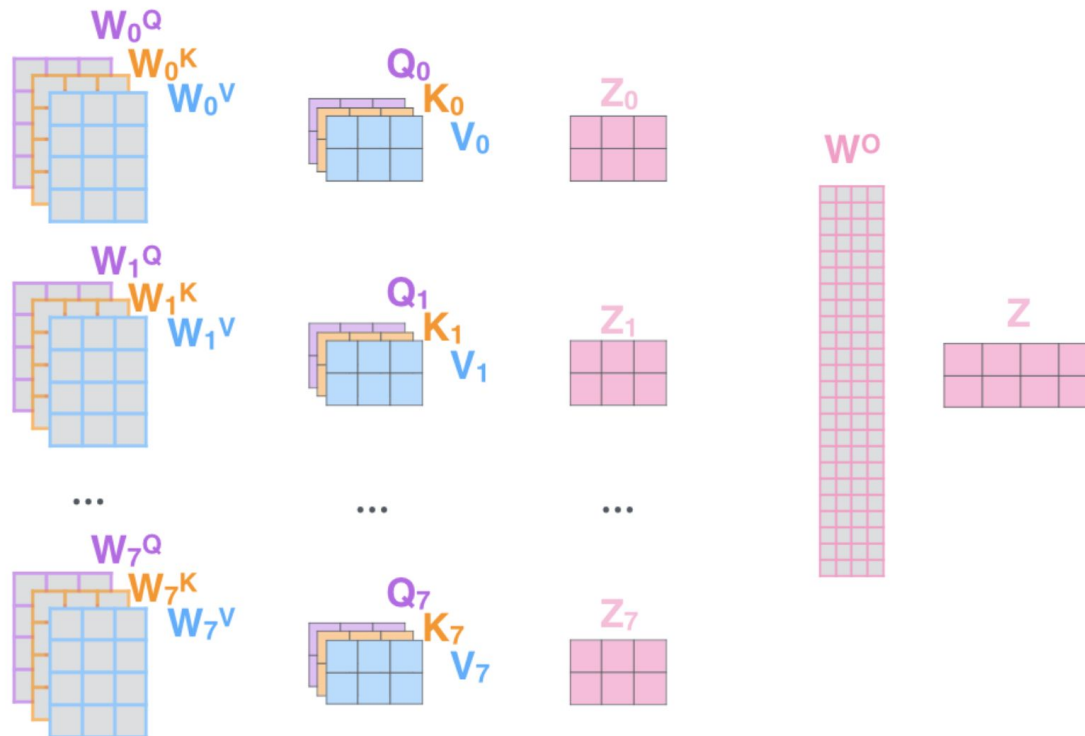
Improves the performance of the attention layer in two ways:

- It expands the model’s ability to focus on different positions.
- It gives the attention layer multiple “representation subspaces”.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where  $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

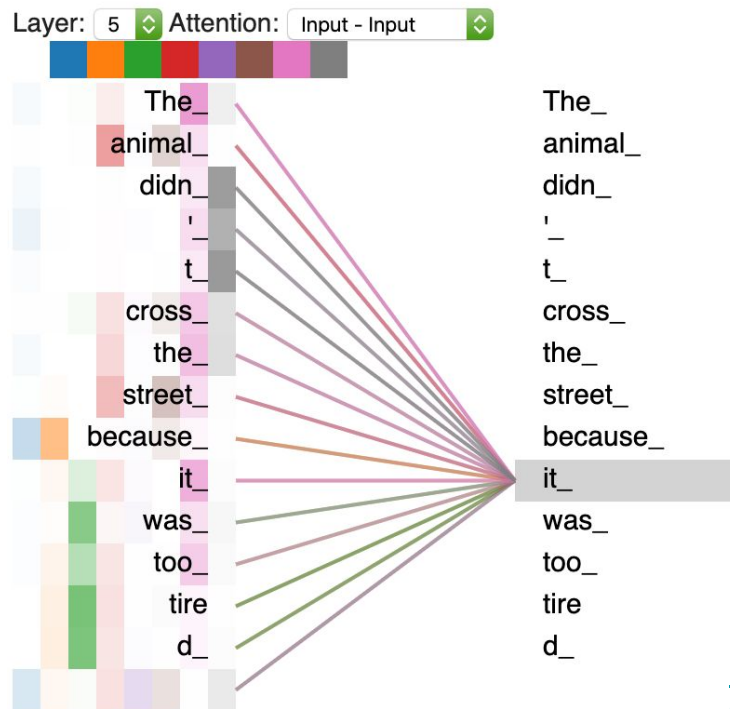
# “Multi-headed” Attention



Acknowledge to figure from: <http://jalammar.github.io/illustrated-transformer/>



# Attention Visualization



[https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello\\_t2t.ipynb#scrollTo=OJKU36QAfQOC](https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=OJKU36QAfQOC)

# Layer Normalization

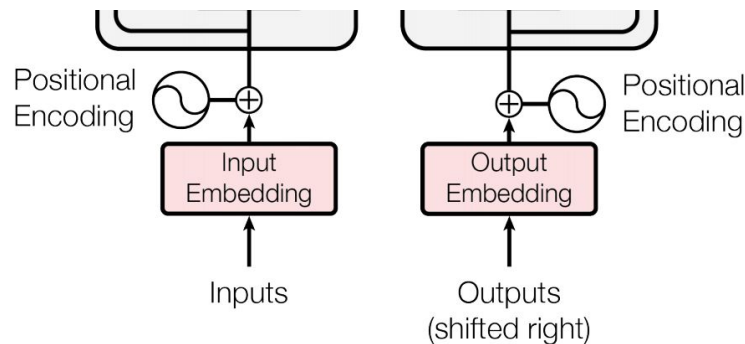
- Normalize each layer to have 0 mean and 1 variance.
- This reduces “covariate shift” (i.e. gradient dependencies between each layer) and therefore fewer iterations are needed.

# Positional Encoding

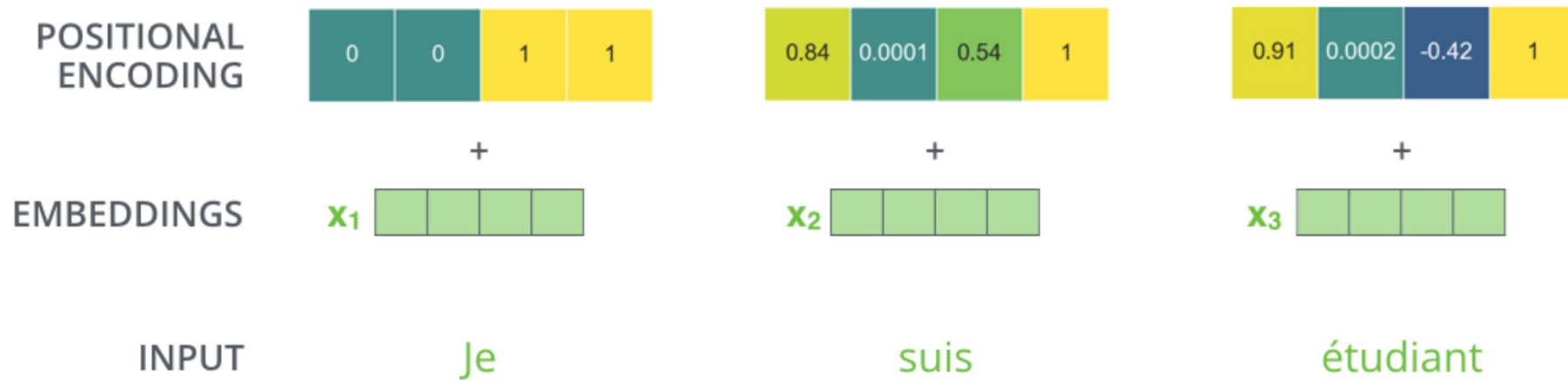
- Embedding to distinguish each position

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



# Models Input



A real example of positional encoding with a toy embedding size of 4

# Evaluation for Transformer

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	<b><math>3.3 \cdot 10^{18}</math></b>	
Transformer (big)	<b>28.4</b>	<b>41.8</b>	$2.3 \cdot 10^{19}$	

# Transformers Implementations

- The Annotated Transformer: Pytorch implementation by Harvard University  
<http://nlp.seas.harvard.edu/2018/04/03/attention.html>
- SEQUENCE-TO-SEQUENCE MODELING WITH NN.TRANSFORMER AND TORCHTEXT: [https://pytorch.org/tutorials/beginner/transformer\\_tutorial.html](https://pytorch.org/tutorials/beginner/transformer_tutorial.html)
- Transformer in Tensor2Tensor library:  
<https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/models/transformer.py>
- Tensorflow Tutorial: [https://www.tensorflow.org/tutorials/text/transformer#positional\\_encoding](https://www.tensorflow.org/tutorials/text/transformer#positional_encoding)

# Pre-training General Language Representations

# Pre-training

**Multi-task learning:** A general term for training on multiple tasks (e.g. predicting eye movements and text summarization)

**Transfer learning:** is a type of multi-task learning where we only care about one of the tasks

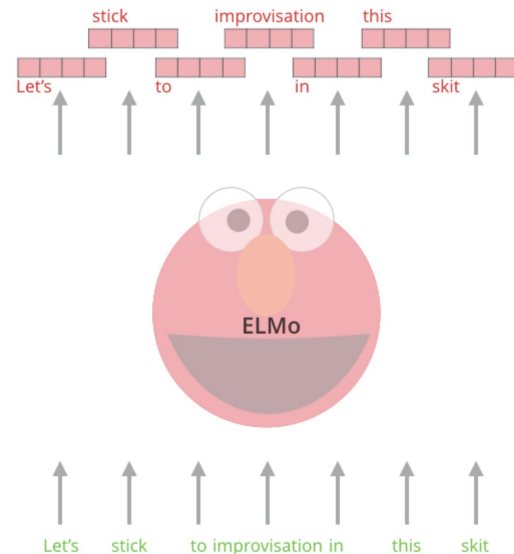
**Pre-training:** Train on a high-resourced task then train on a downstream task with limited label data.



# Word Embeddings in Transfer Learning

- Labeled data are limited
- Enormous unlabeled text corpus
- Pre-trained word embeddings can be transferred to other supervised tasks
- ELMo (Peters et al., 2018):
  - Context matters
  - Predicts the next word left-to-right and next word right-to-left independently

ELMo  
Embeddings

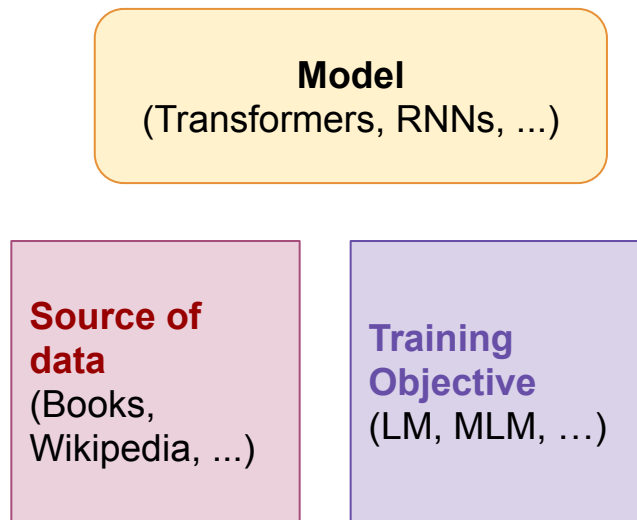


Words to embed

Acknowledge to figure from: <http://jalammar.github.io/illustrated-bert/>

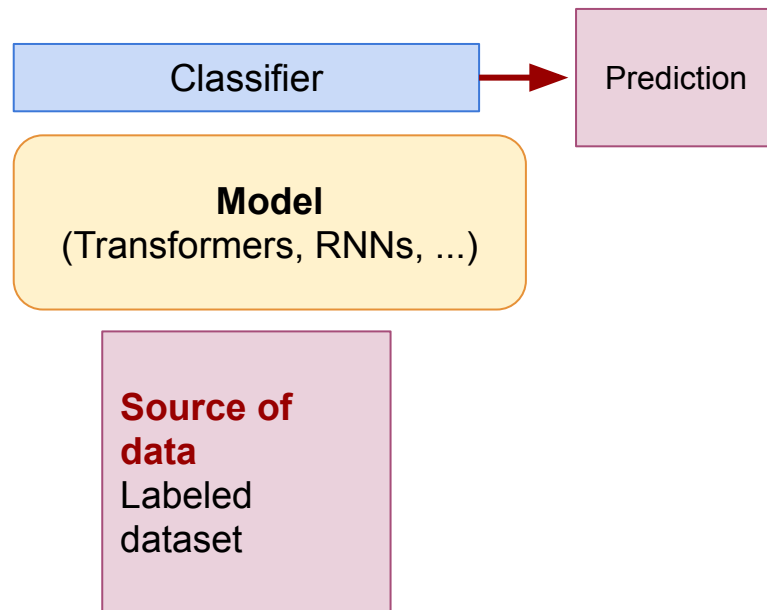
## Step 1: Pre-training process

**Unsupervised** learning on a very large amount of data



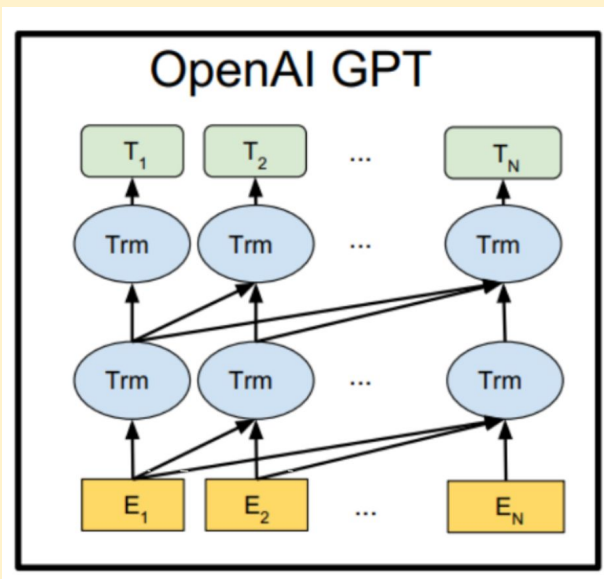
## Step 2: Fine-tuning

**Supervised** training on a specific task with a much smaller labeled dataset



# OpenAI GPT: Pre-training a Transformer Decoder (Unidirectional) for Language Modeling (Radford et al. 2018)

Model: Masked self-attention



**Idea:** Use decoder part of transformer to predict the next word left-to-right

**Source of data**  
7,000 books

**Training Objective**  
Language Modeling

# BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al. 2019)

**Problem:** We need a bi-directional model!

**Idea:** Use Encoder part of transformer

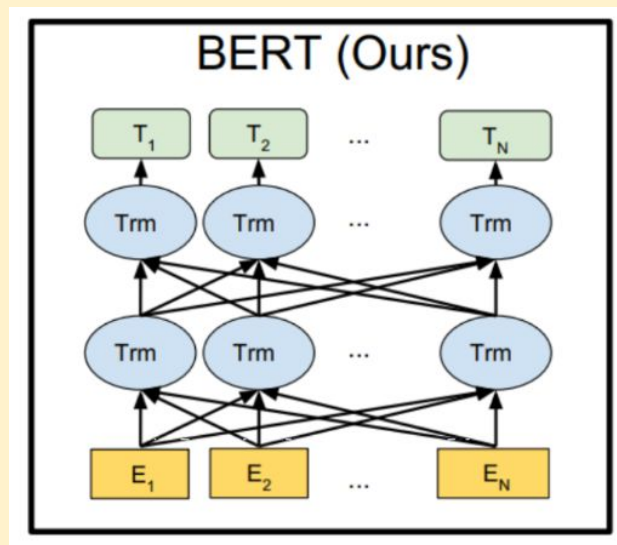
## Source of data

BooksCorpus  
(800M words),  
English  
Wikipedia  
(2,500M  
words)

## Training Objective

Masked  
Language  
Model + Next  
Sentence  
prediction

Model: Multi-layer self-attention



# BERT Input Representation

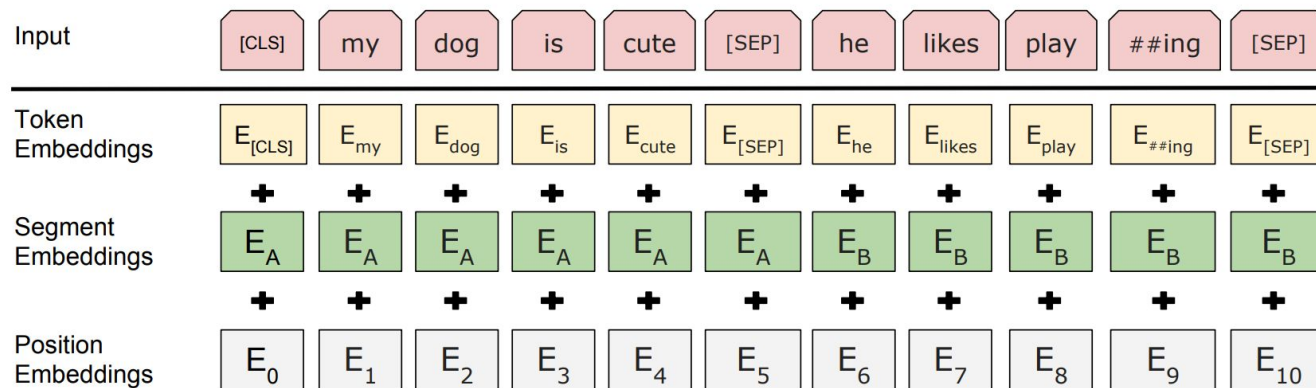


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

# BERT - Masked Language Model Objective

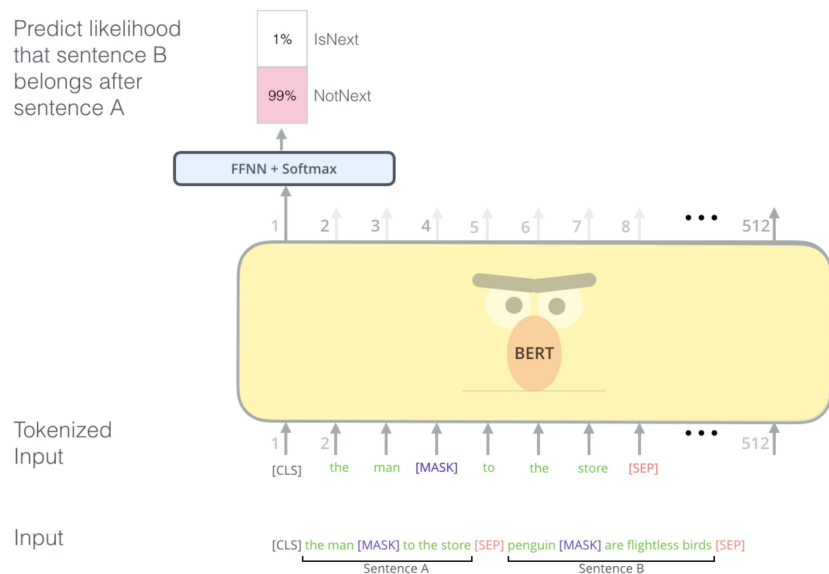
- Randomly select 15% of the tokens
  - For 80% of the time:
    - Replace the word with the [MASK] token
  - For 10% of the time:
    - Replace the word with a random word
  - For 10% of the time:
    - Keep the word unchanged
- **Downside:** we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning. To mitigate this, we do not always replace “masked” words with the actual [MASK] token.

Acknowledge to figure from: <http://jalammar.github.io/illustrated-bert/>

# BERT - Next sentence prediction objective

## Next sentence prediction (Binary classification)

- Randomly select a split over sentences:
  - Store the segment A
  - For 50% of the time:
    - Sample random sentence split from another document as segment B.
  - For 50% of the time:
    - Use the actual sentences as segment B.



The second task BERT is pre-trained on is a two-sentence classification task. The tokenization is oversimplified in this graphic as BERT actually uses WordPieces as tokens rather than words --- so some words are broken down into smaller chunks.

Acknowledge to figure from: <http://jalammar.github.io/illustrated-bert/>

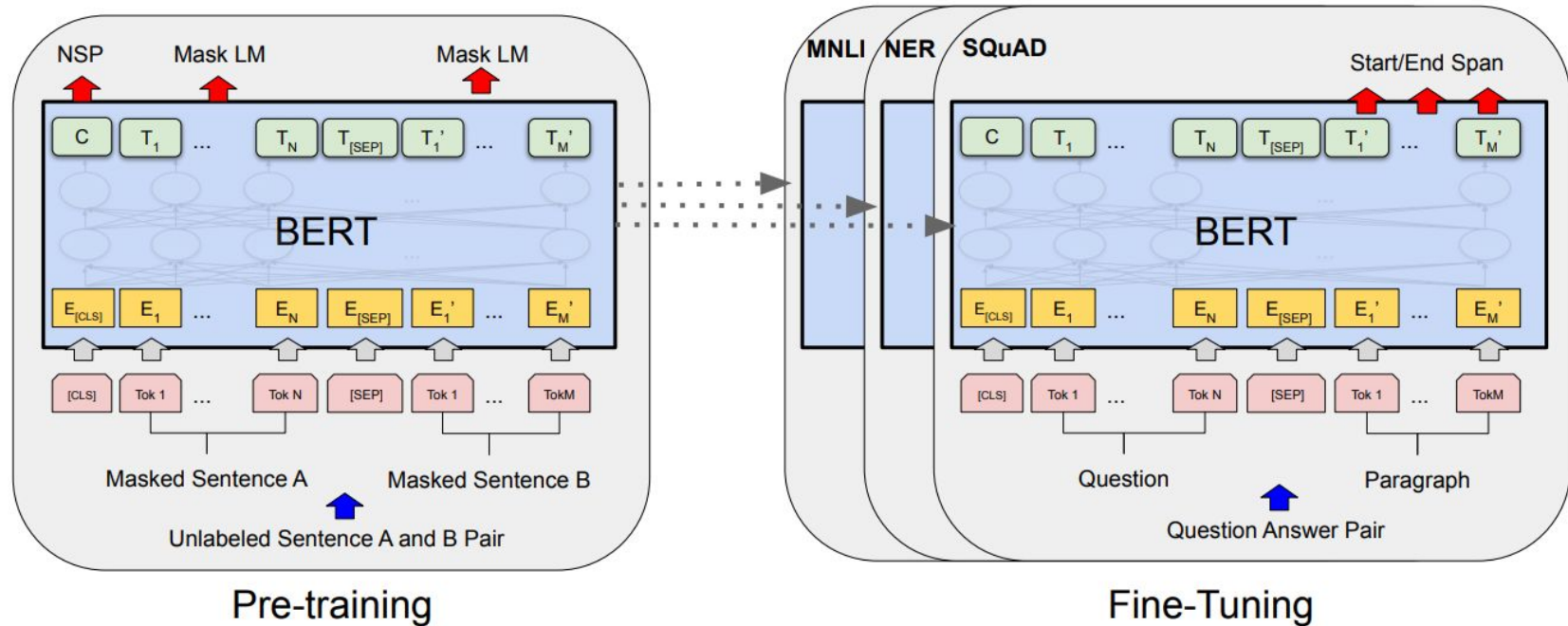
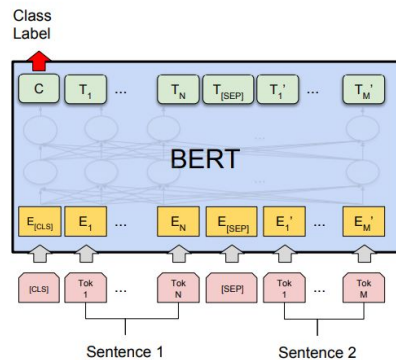


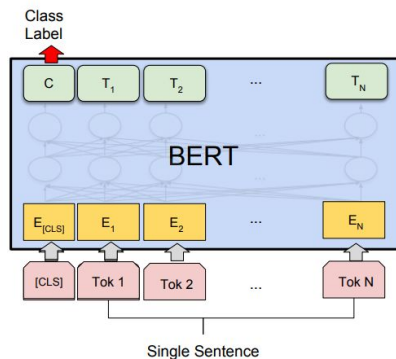
Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).



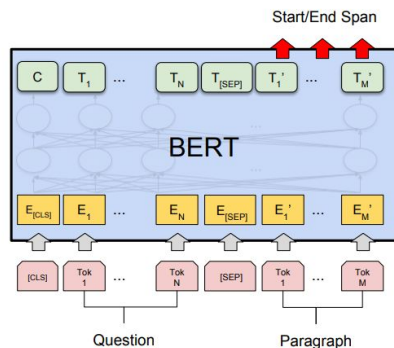
# The illustration of fine-tuning BERT on different tasks



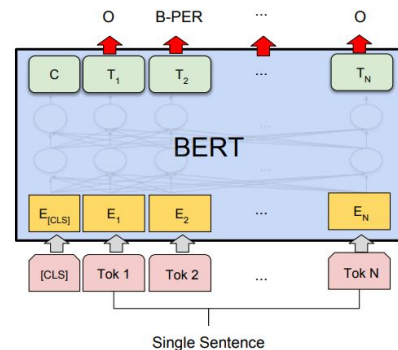
(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG



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



(c) Question Answering Tasks:  
SQuAD v1.1



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

Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

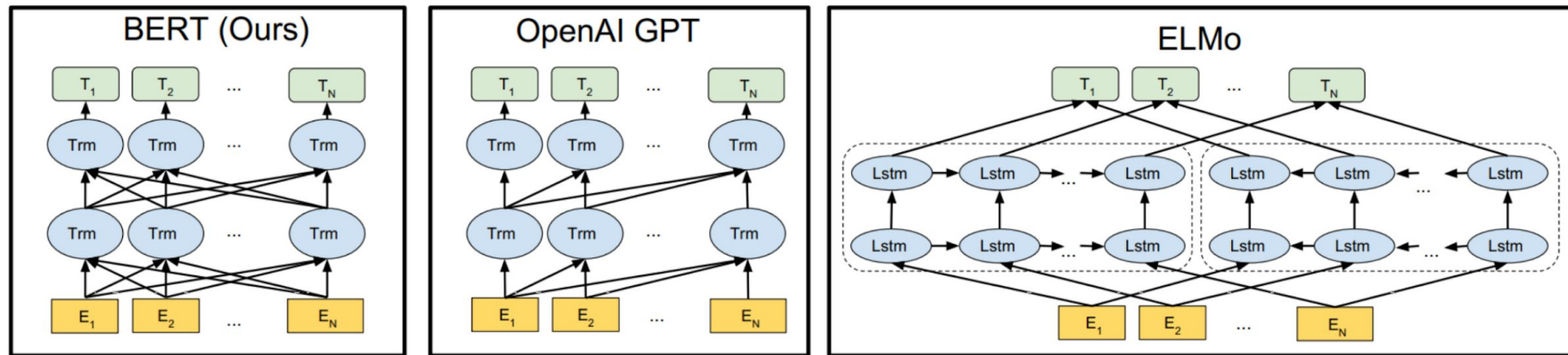
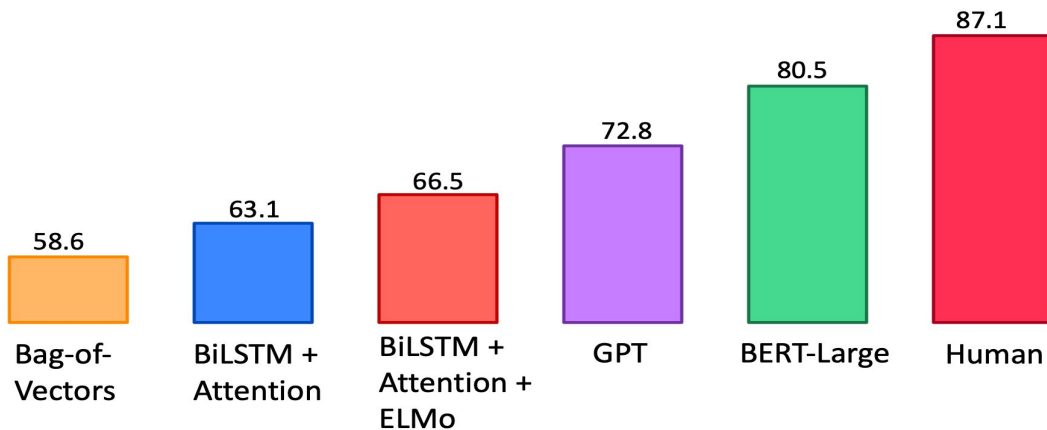


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

# General Language Understanding Evaluation (GLUE) benchmark:

BERT was integrated in Google search, improving an estimated 10% of queries ([link](#)).

## GLUE Benchmark Results



Acknowledge to figure from: <https://gluebenchmark.com/leaderboard>  
[Kevin Clark. Future of NLP + Deep Learning. Mar 2019. CS224n. Stanford University.](#)

# How to improve?

- Better results:
  - Better models
  - More data
  - Larger model
  - Train longer
- More efficient models:
  - Optimal trade-off between performance and efficiency
  - Lighter model (fewer parameters)
  - Faster
  - Lower training costs

# RoBERTa (Liu et. al, 2019)

**Problem:** BERT was significantly undertrained. Modifications are:

- Training the model longer, with bigger batches, over more data
- Removing the next sentence prediction objective
- Training on longer sequences
- Dynamically changing the masking pattern applied to the training data.

RoBERTa has the same architecture as BERT, but uses a byte-level BPE as a tokenizer (same as GPT-2).

**Idea:** training the model longer on a bigger dataset.

**Source of data**  
63M News  
English Article

**Training Objective**  
MLM

# ALBERT (Lan et. al, 2020)

## Modifications:

- Splitting the embedding matrix into two smaller matrices
- Using repeating layers split among groups

ALBERT has 18x fewer parameters  
Can be trained about 1.7x faster  
~9% improvement on GLUE

**Idea:** Presenting two parameter-reduction techniques to lower memory consumption and increase the training speed of BERT.

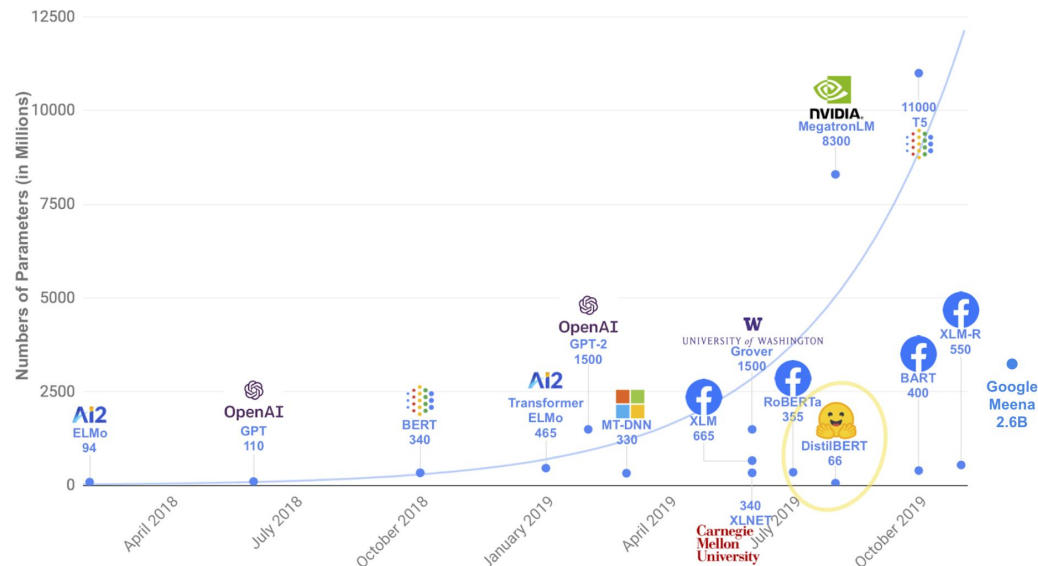
### Source of data

As in (Devlin et al., 2019)

### Training Objective

MLM,  
sentence-order  
prediction  
(SOP)

# DistilBERT (Sanh et. al, 2019)



**Idea:** by leveraging Knowledge distillation, they reduce the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster.

## Source of data

As in (Devlin et al., 2019)

## Training Objective

language modeling, distillation and cosine-distance losses

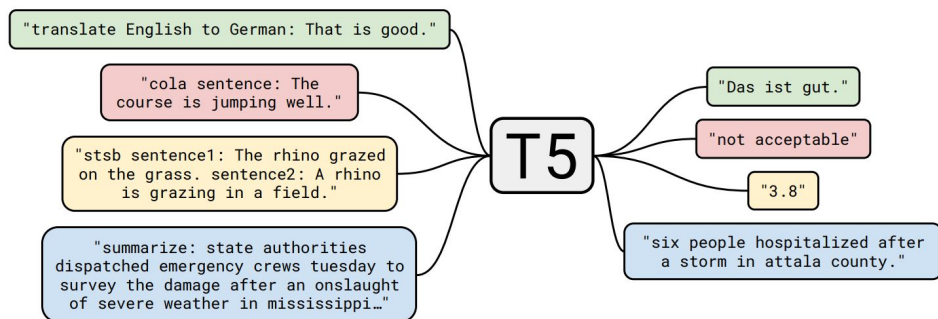
# Which one is better?

	BERT	RoBERTa	DistilBERT
Size (millions)	Base: 110 Large: 340	Base: 110 Large: 340	Base: 66
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.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from 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.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation

Acknowledge to figure from: Suleiman Khan, [BERT, RoBERTa, DistilBERT, XLNet — which one to use?](#)



# T5 (Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer) (Raffel et. al, 2019)



**Figure 1:** A diagram of our text-to-text framework. Every task we consider – including translation, question answering, and classification – is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. “T5” refers to our model, which we dub the “Text-to-Text Transfer Transformer”.

**Idea:** introducing a unified framework (extremely large new neural network model) that converts every language problem into a text-to-text format.

## Source of data

“Colossal Clean Crawled Corpus” (about 750 GB)

**Training Objective**  
maximum likelihood objective

# How to use? (Notebook)



## Transformers

- <https://github.com/huggingface/transformers>
- [https://colab.research.google.com/drive/14e\\_2vNjkAx\\_BdslbO2zUK6G8feRWC\\_Qn](https://colab.research.google.com/drive/14e_2vNjkAx_BdslbO2zUK6G8feRWC_Qn)

# Recommended Learning Resources

- A Primer in BERTology: What we know about how BERT works.  
<https://arxiv.org/pdf/2002.12327.pdf>
- Kevin Clark. Future of NLP + Deep Learning. Mar 2019. CS224n. Stanford University. <http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture20-future.pdf>
- Jacob Devlin. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. <https://nlp.stanford.edu/seminar/details/jdevlin.pdf>
- Jay Alammar. A Visual Guide to Using BERT for the First Time
- Jay Alammar. The Illustrated Transformer.
- Jay Alammar. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)

# References

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need."

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding."

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692* (2019).

Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. "Albert: A lite bert for self-supervised learning of language representations." *arXiv preprint arXiv:1909.11942* (2019).

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer." *arXiv preprint arXiv:1910.10683* (2019).

Jay Alammar. The Illustrated Transformer. <http://jalammar.github.io/illustrated-transformer/>

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *arXiv preprint arXiv:1910.01108* (2019).