Transformers for NLP and IR

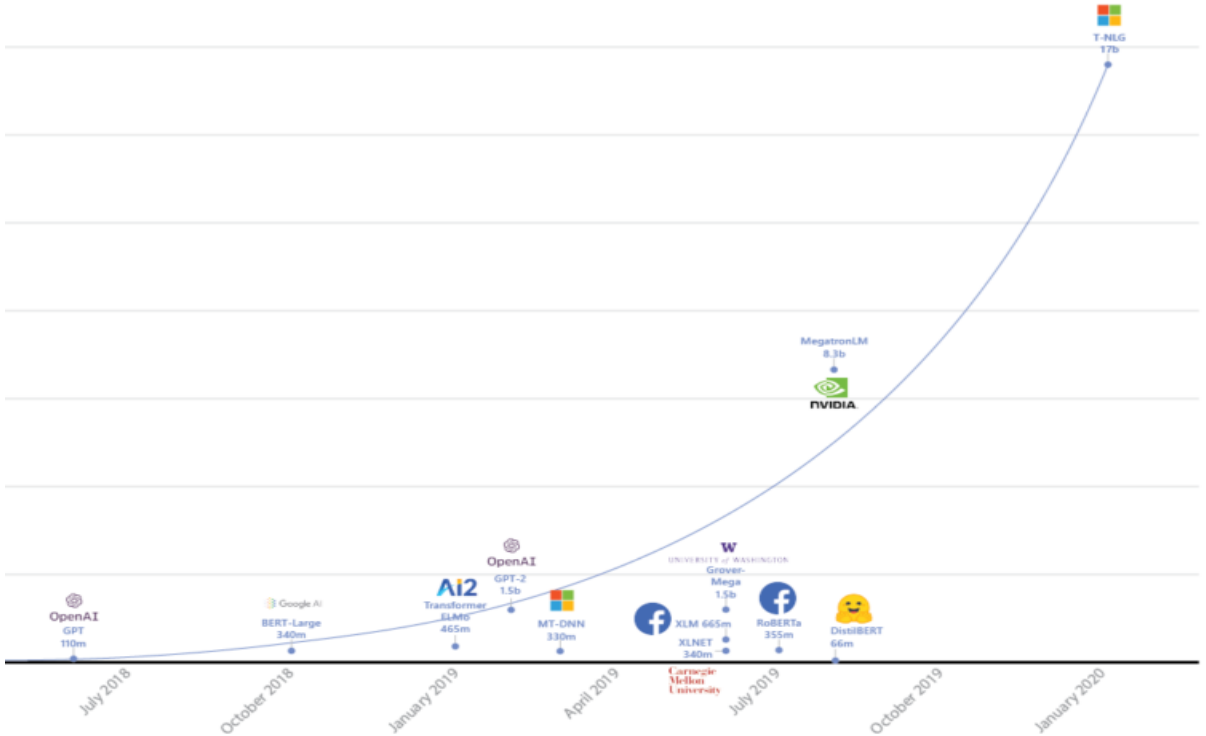
Prepared by Diana Inkpen at the University of Ottawa, 2021-2020

Based on Vaswani et al., NIPS 2017, “Attention is all you need”
Some slides by Scott Parkinson

Based on Devlin et al., ACL 2018, “ BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”
and on material by Rani Horev https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Transformer

- Significant modern deep learning architecture.
- Has led to a revolution in NLP.
- Trained models are increasingly massive.
- Self Supervised.

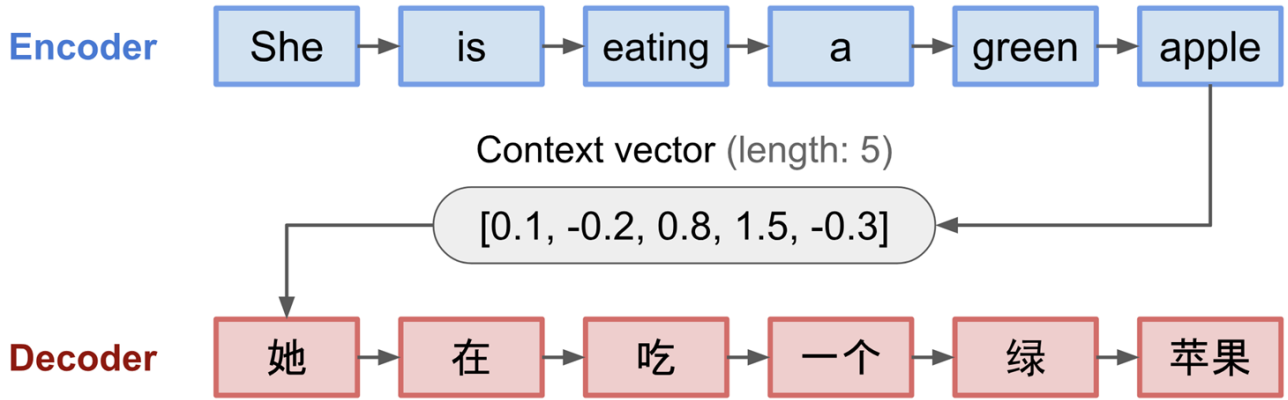


https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

The Problem Space

Sequence to Sequence (seq2seq), input/output is no longer a fixed size
examples: language translation, sentiment analysis, speech encoding

Trained models build a strong semantic representation of language.



Contextual Embedding

1. I broke the vase.
2. I am broke.
3. I broke the record.

What is the meaning of the word broke?

Transformer

Attention is all you need. Vaswani et al. (2017)

Sequence to sequence while handling long term dependencies using attention.

Benefits: attention, parallelization, addresses vanishing gradient issue of RNNs

Cons: complexity, difficulty and cost to train, size, resources

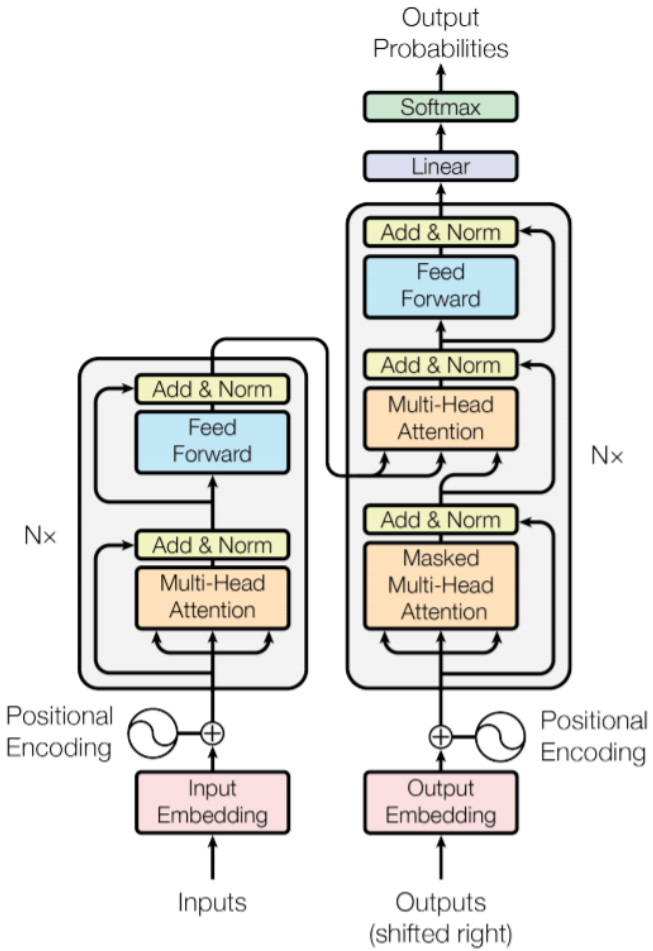


Figure 1: The Transformer - model architecture.

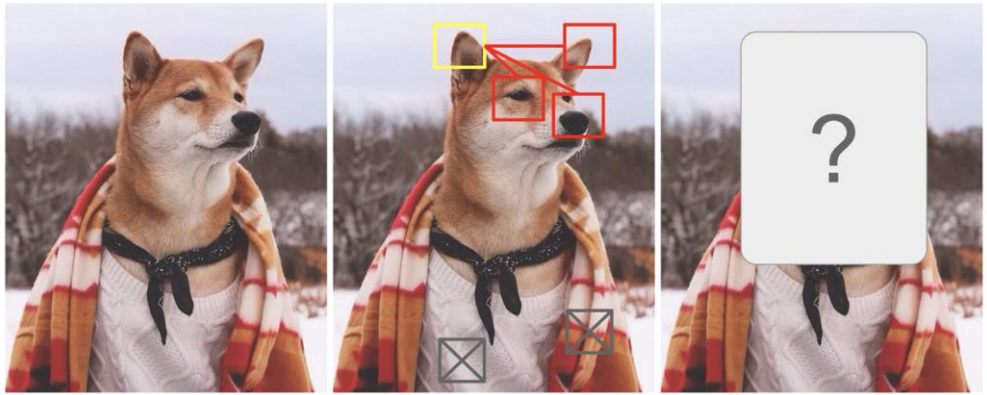
The Model

Important: Attention; works on entire input (not just single word)/multi hidden state propagation

Characteristics: Encoder/Decoder design, Self Supervised (next token or masked prediction)

Important DL techniques: Stacking to make deep/add (residual connection) and layer normalization, label smoothing, Adam optimizer (learning rate scheduler), very hard to train right on your own

Attention: Intuition

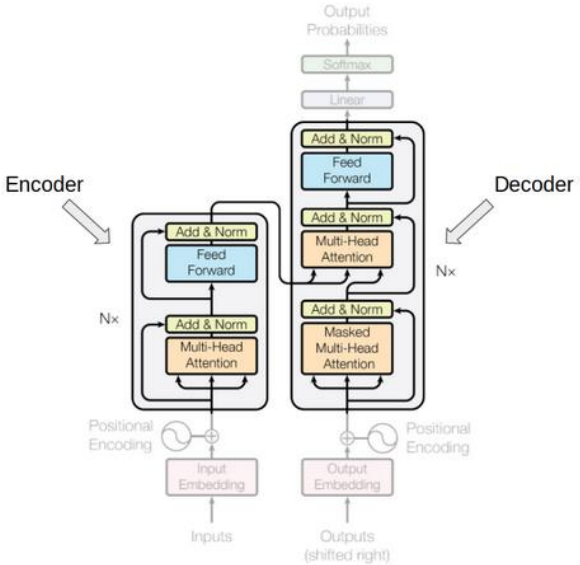


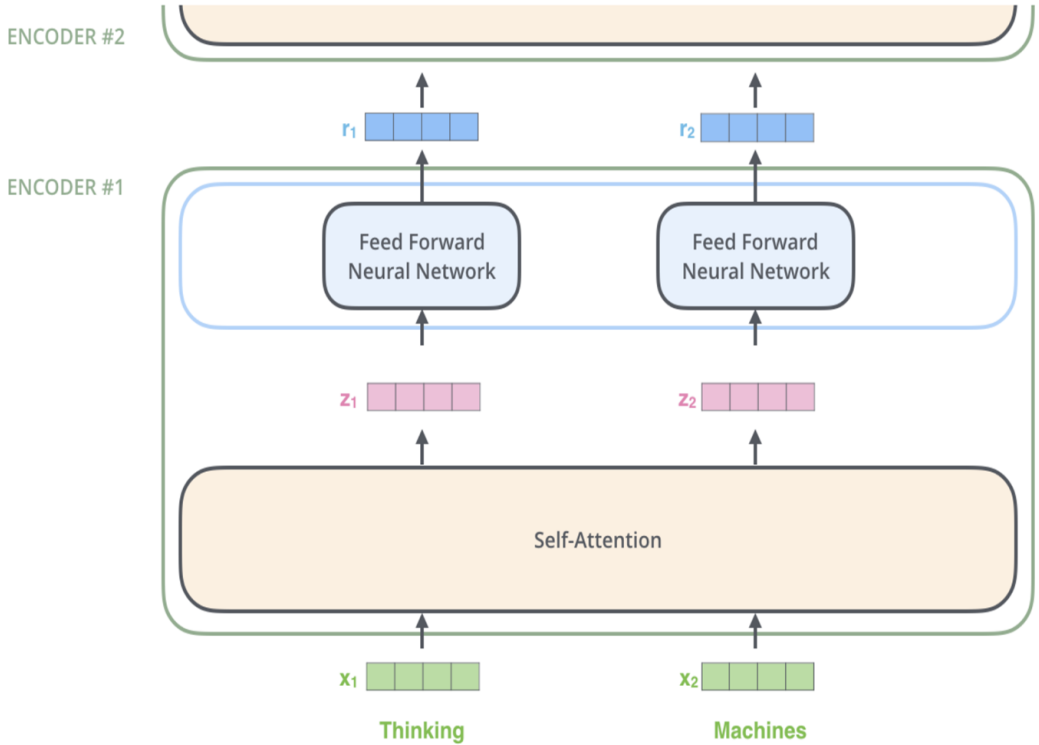
high attention

low attention

She is eating a green apple.

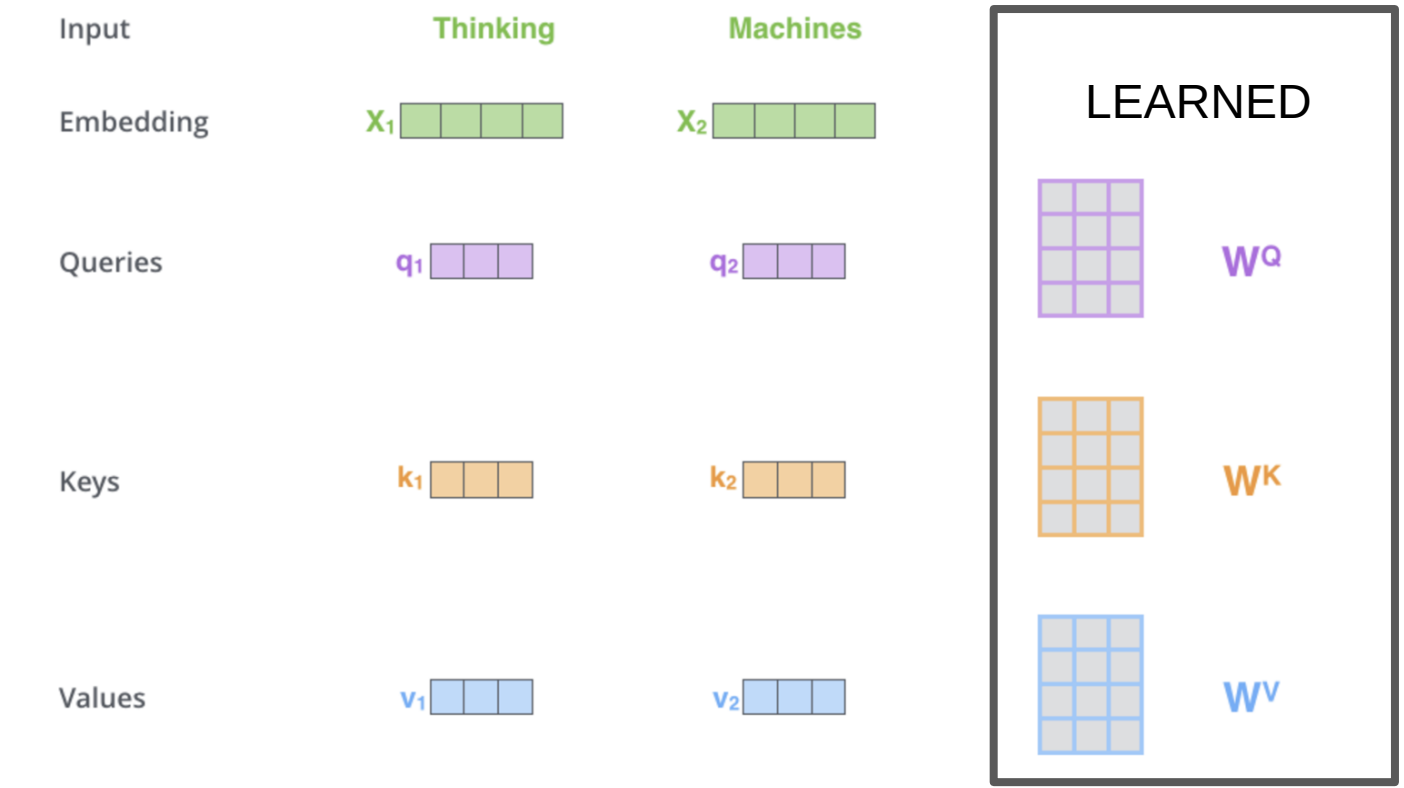
https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html





The word at each position passes through a self-attention process. Then, they each pass through a feed-forward neural network -- the exact same network with each vector flowing through it separately.

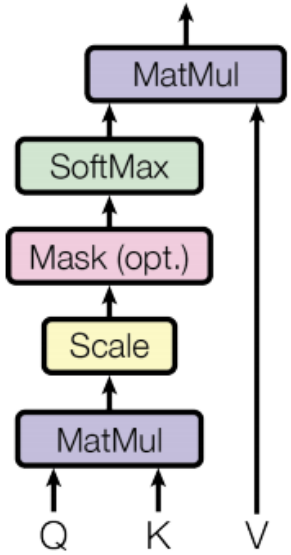
http://jalammar.github.io/illustrated-transformer/



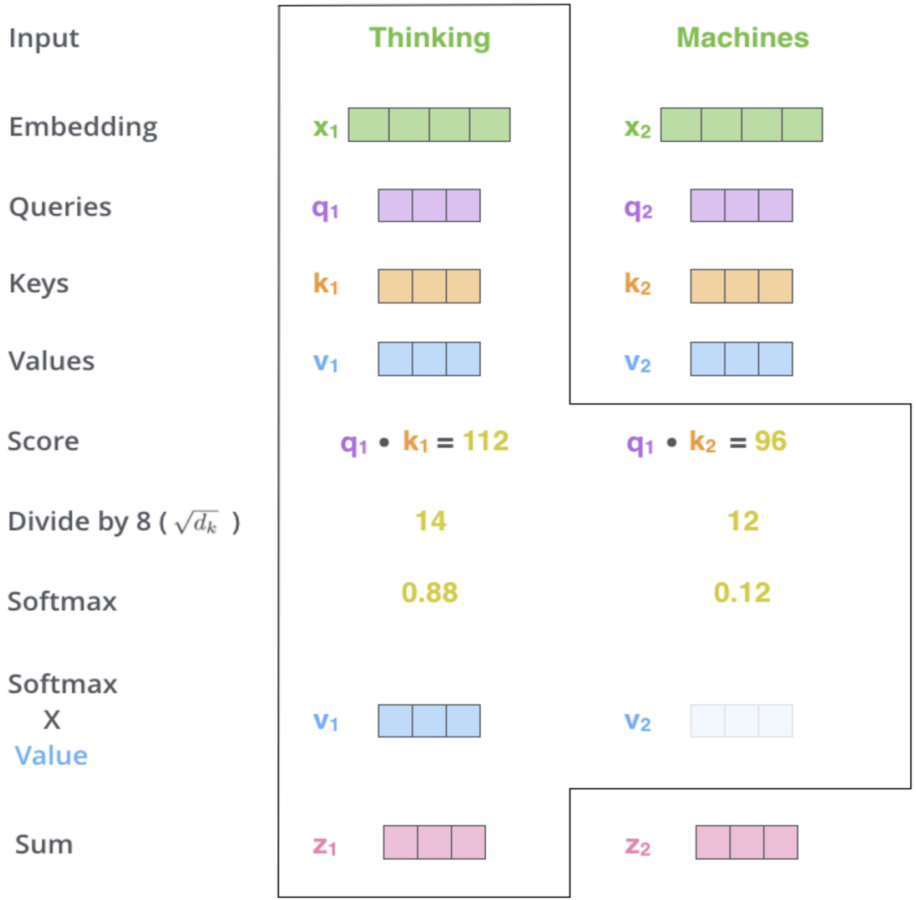
Multiplying x1 by the WQ weight matrix produces q1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

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

Scaled Dot-Product Attention

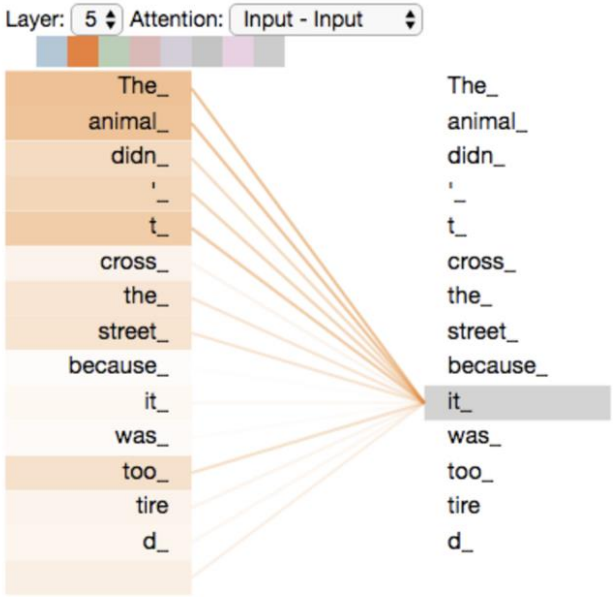


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



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

Attention Example



As we are encoding the word "it" in encoder #5 (the top encoder in the stack), part of the attention mechanism was focusing on "The Animal", and baked a part of its representation into the encoding of "it".

Summary

Transformer models can be too big to train from scratch.

Can use transfer learning on pre-trained models.

There are smaller models with only small loss of performance.

Semi Supervised learning.

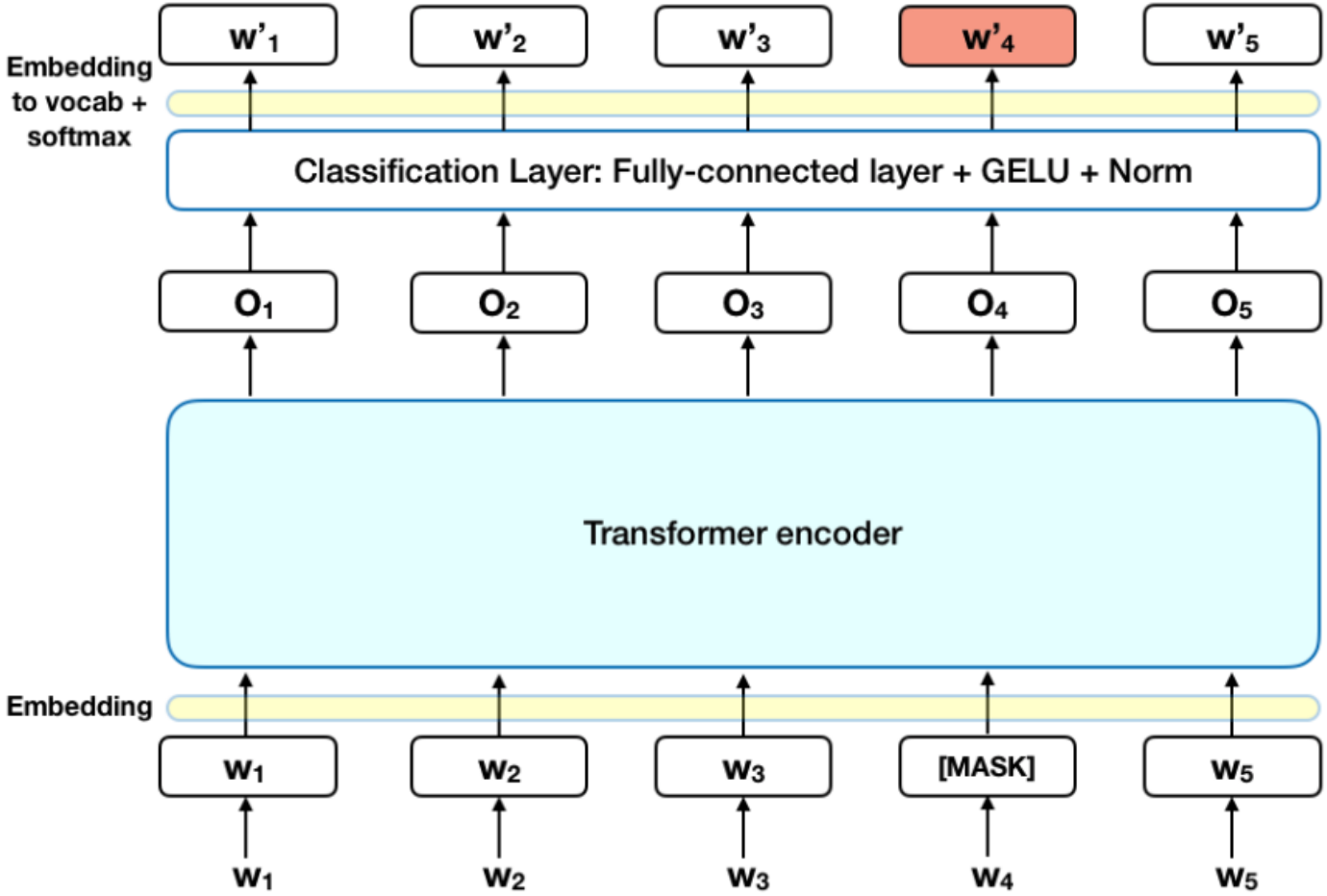
Attention is an important concept.

Many online resources to dig deeper.

BERT

(Bidirectional Encoder Representations from Transformers)

- BERT’s key technical innovation is applying the bidirectional training of Transformer to language modelling.
- Previous language models (LM) looked at a text sequence from left to right.
- LM predict the next word in a sequence: “The child came home from ___”
- Attention mechanism learns contextual relations between words (or sub-words) in the text.
- Used only the encoder part of the transformer. No decoder needed. The output is a language model


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Masked LM

- Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token.
- The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.
- BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words.

- Steps:

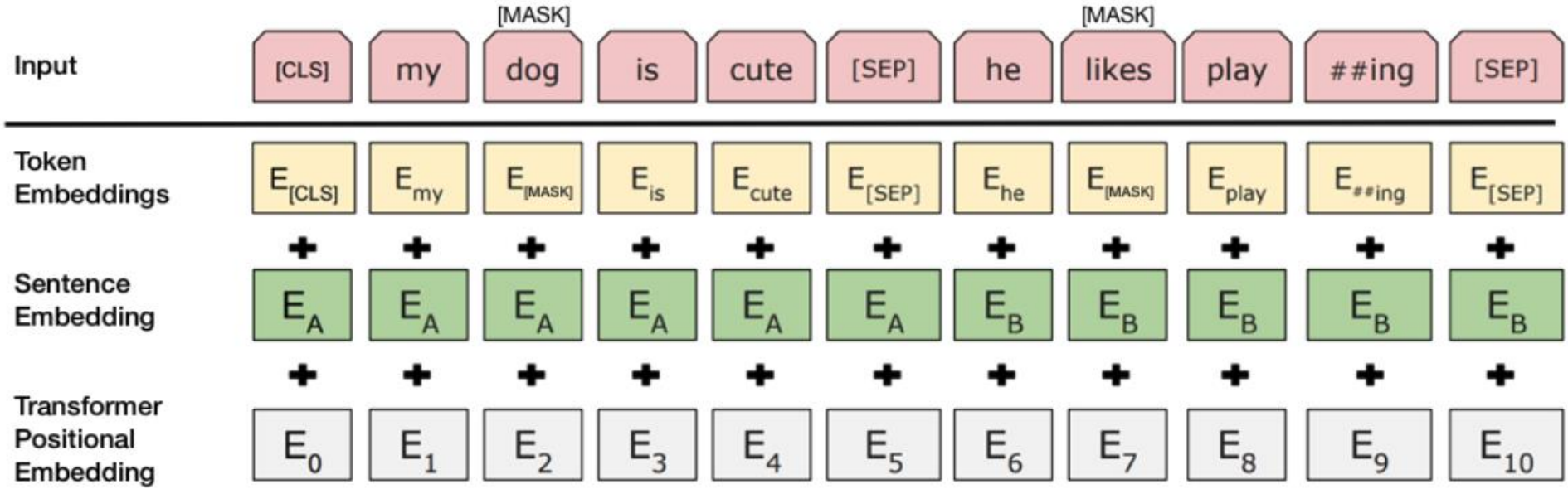
1. Add a classification layer on top of the encoder output.
2. Multiply the output vectors by the embedding matrix, transforming them into the vocabulary dimension.
3. Calculate the probability of each word in the vocabulary with softmax.

Next Sentence Prediction (NSP)

- The model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document.
- 50% positive examples: 50% pairs of consecutive sentences
- 50% pairs in which the second sentence is a random sentence from the corpus.

- Steps:

1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. A sentence embedding indicating Sentence A or Sentence B is added to each token. A positional embedding is added to each token to indicate its position in the sequence.


Input
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
likes
[MASK]
play
##ing
[SEP]
Token Embeddings
E[CLS]
Emy
E[MASK]
Eis
Ecute
E[SEP]
Ehe
E[MASK]
Eplay
E##ing
E[SEP]
Sentence Embedding
EA
EB
Transformer Positional Embedding
E0
E1
E2
E3
E4
E5
E6
E7
E8
E9
E10

Training

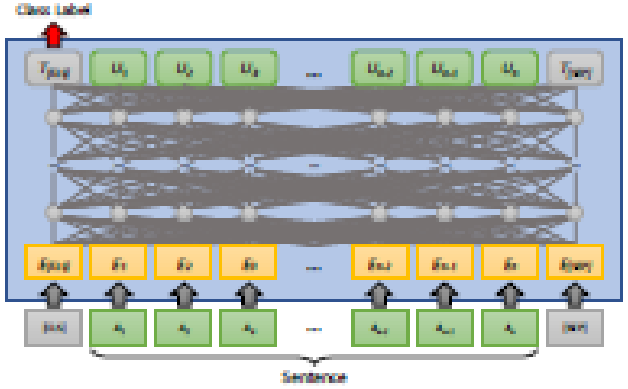
- Steps for predicting if the second sentence is connected to the first, the following steps are performed:

1. The entire input sequence goes through the Transformer model.
2. The output of the [CLS] token is transformed into a 2×1 shaped vector, using a simple classification layer (learned matrices of weights and biases).
3. Calculating the probability of IsNextSequence with softmax.

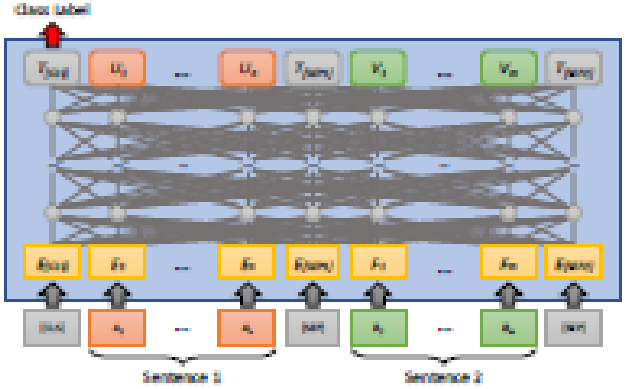
- When training the BERT model, Masked LM and Next Sentence Prediction are trained together, with the goal of minimizing the combined loss function of the two strategies.

How to use BERT for a classification task (fine-tuning): Add a layer to the model

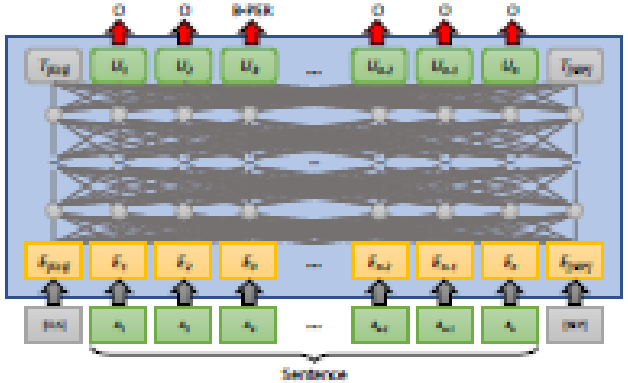
- Sentiment analysis is similar to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token.
- In Question Answering (e.g., SQuAD v1.1), the software receives a question regarding a text sequence and is required to mark the answer in the sequence. The model is trained by learning two extra vectors that mark the beginning and the end of the answer.
- In Named Entity Recognition (NER), the software receives a text sequence and is required to mark the various types of entities (Person, Organization, Date, etc) that appear in the text. The NER model is trained by feeding the output vector of each token into a classification layer that predicts the NER label.



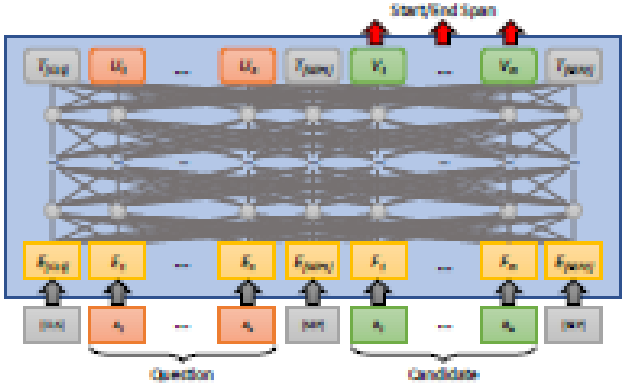
(a) Single-Input Classification Tasks



(b) Two-Input Classification Tasks



(c) Single-Input Sequence Labeling Tasks



(d) Two-Input Sequence Labeling Tasks

Examples:

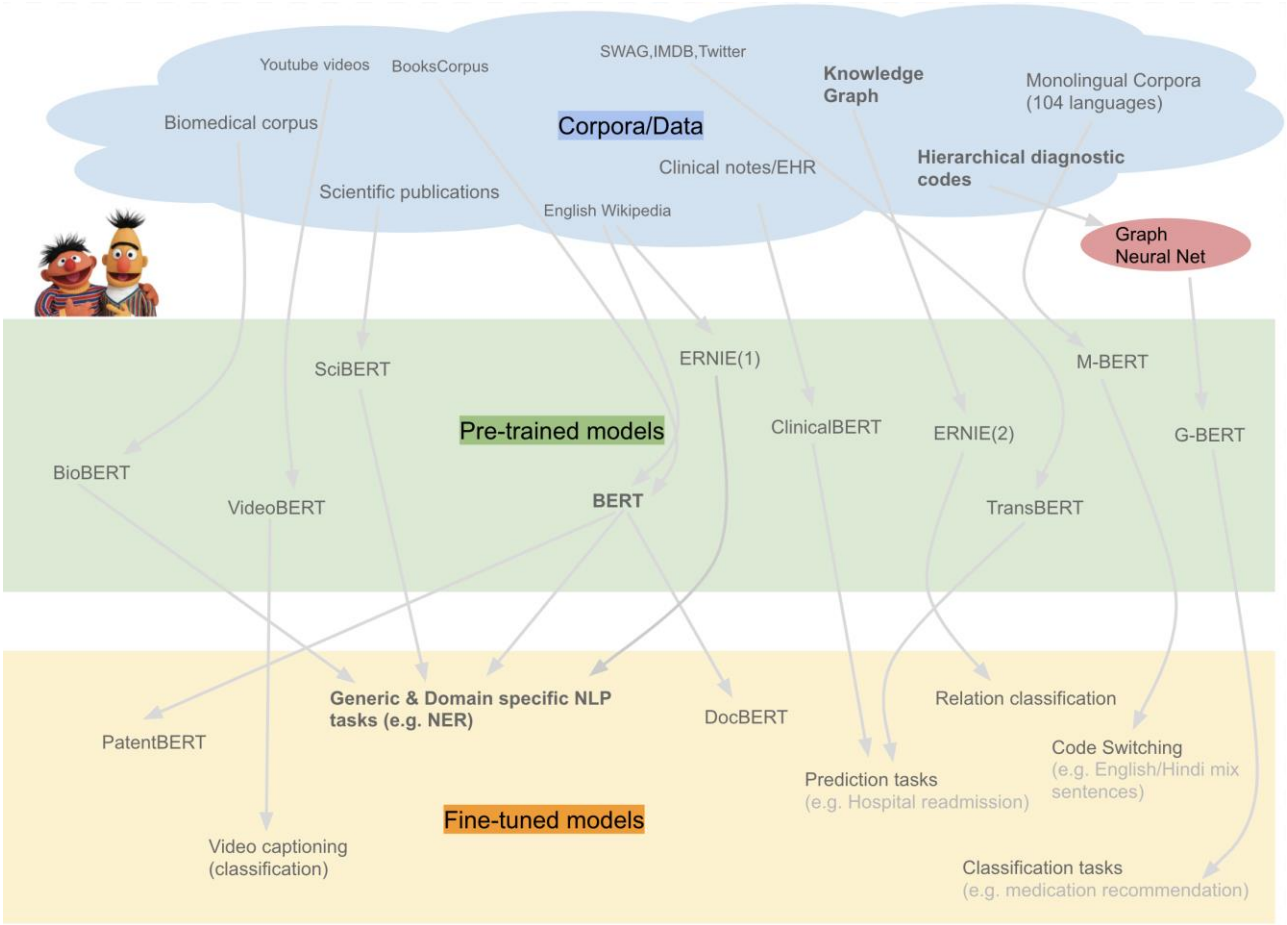
a) sentiment analysis

b) detecting if two sentences are paraphrases of each other

c) named-entity recognition (B, I, O as class labels)

d) question answering

Many versions of BERT

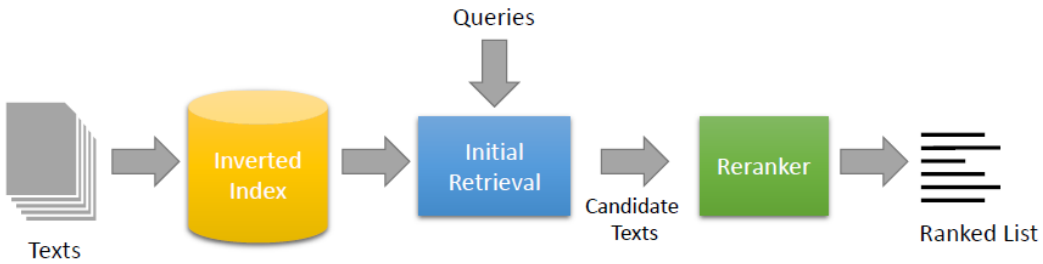


https://towardsdatascience.com/a-review-of-bert-based-models-4ffdc0f15d58
Examples from 2019.
Many others.

How to use BERT for IR

- Represent queries as vectors
- Represent document as vectors (average sentence vectors)
- Compute cosine similarity

For efficiency, use it on re-ranking after boolean retrieval via an index.



Use pre-trained models. Or train more complex models based on training queries.

Resources

https://github.com/sannykim/transformers

http://www.peterbloem.nl/blog/transformers