NLP Basics: Part 3

Colin Magdamo

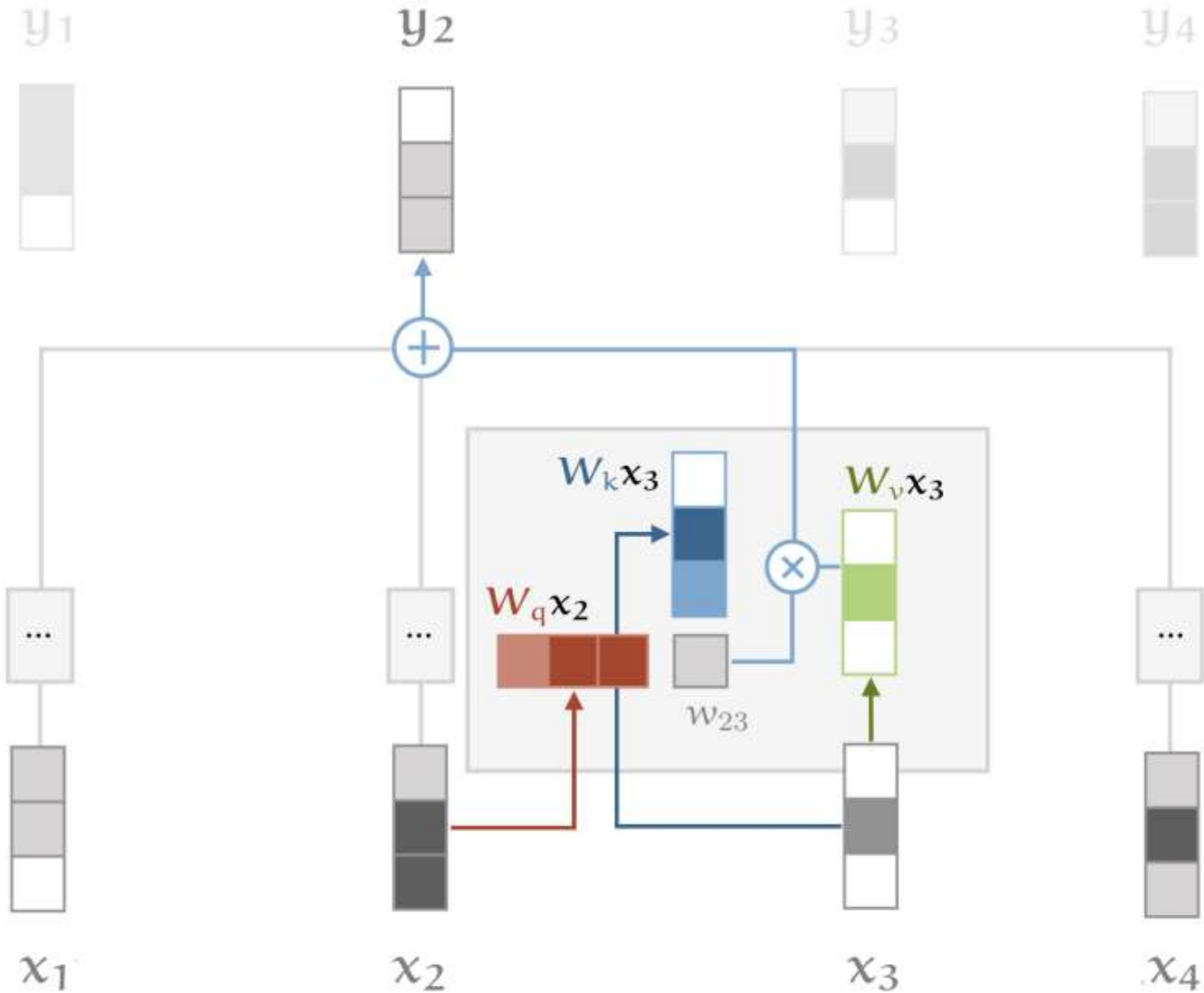
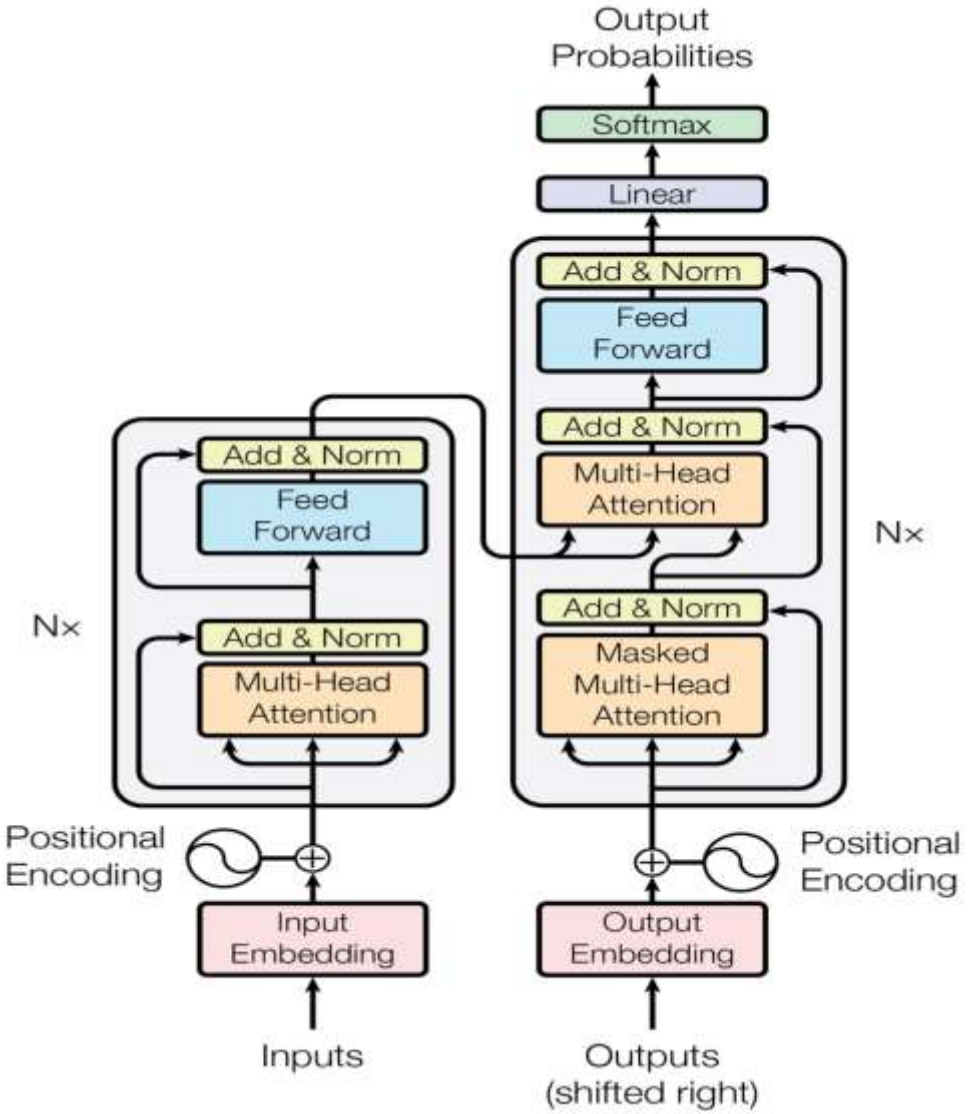


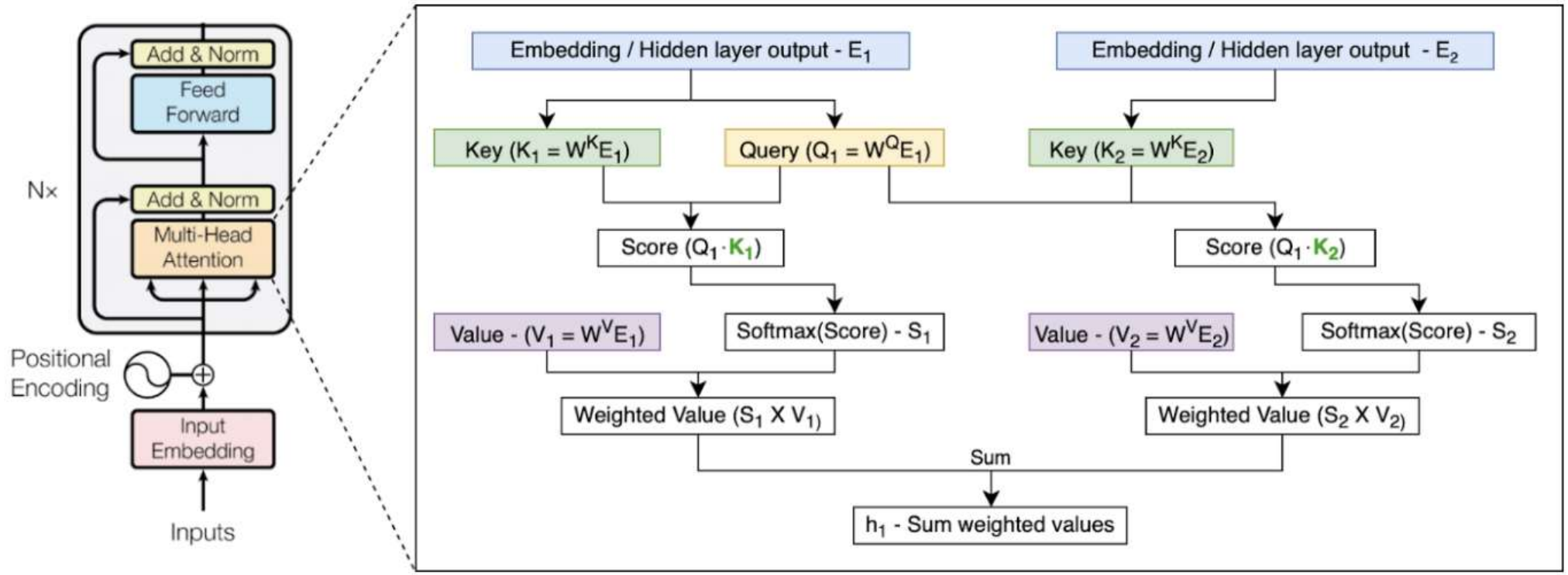
Illustration of the self-attention with key, query and value transformations.

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

The Original Transformer Model



https://arxiv.org/abs/1706.03762



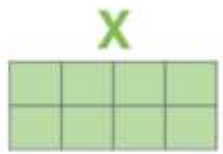
An example of a single Attention Head on a single token (E1). Its output is calculated using its Query vector, and the Key and Value vectors of all tokens (In the chart we show only one additional token E2) — The Query and the Key define the weight of each token, and the output is the weighted sum of all Value vectors.

https://towardsdatascience.com/transformer-xl-explained-combining-transformers-and-rnns-into-a-state-of-the-art-language-model-c0cfe9e5a924

1) This is our input sentence*

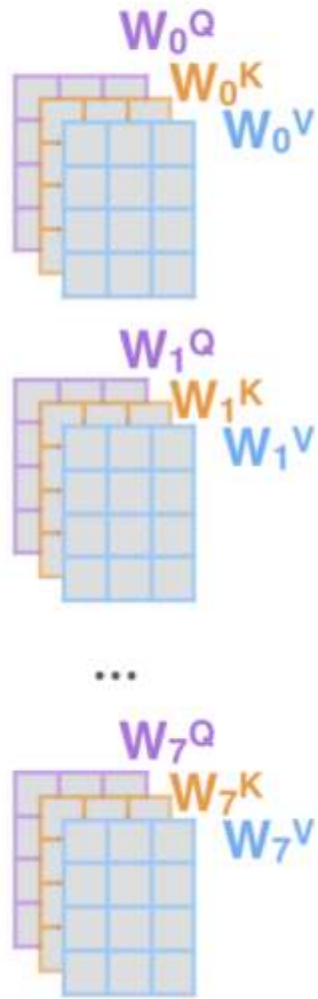
Thinking
Machines

2) We embed each word*

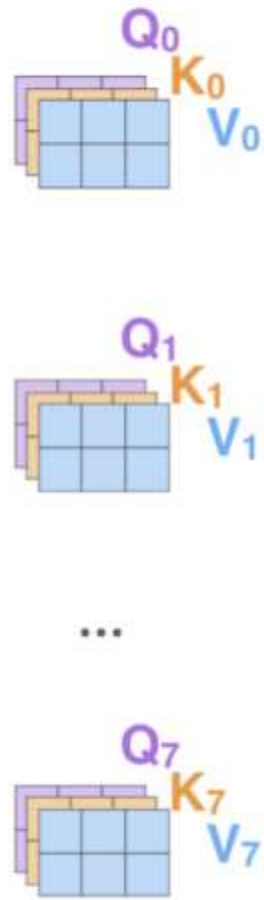


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

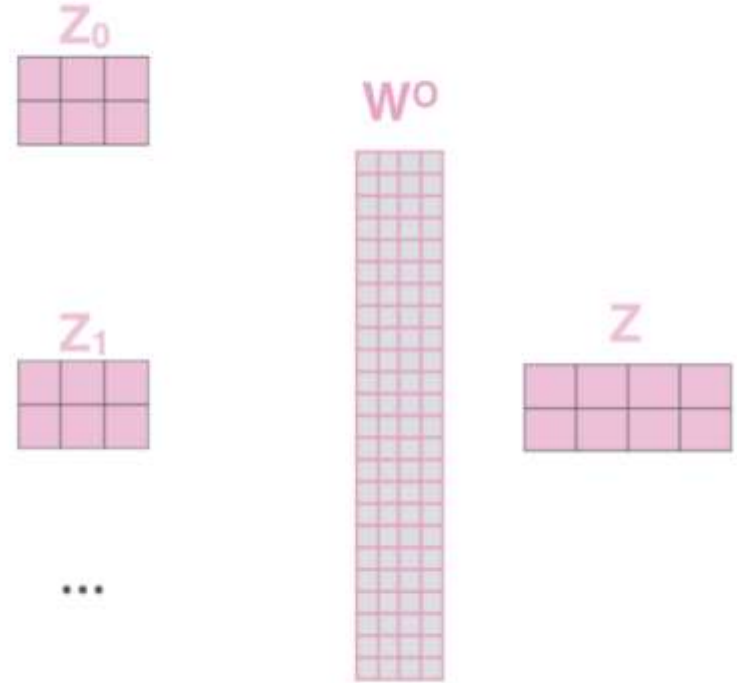
3) Split into 8 heads. We multiply X or R with weight matrices



4) Calculate attention using the resulting Q/K/V matrices



5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



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

Multihead Attention: 13 tokens, 512 Dimensional Embeddings, 8 Heads

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

Scale so input to softmax doesn't explode

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

13×512 | 13×64 | h = 8 | 512×512

13×512

https://medium.com/@jonathan_hui/nlp-bert-transformer-7f0ac397f524

Core Attention Model

- Key: word embedding representing a token
  - Depressed represents an emotional state, has a negative connotation
- Query:
  - What matches the key? What is the key looking for? In this case depressed might look for ‘subjects that could be in emotional states’ or ‘other words that reflect negative sentiments’ or ‘behavioral consequences of being depressed’

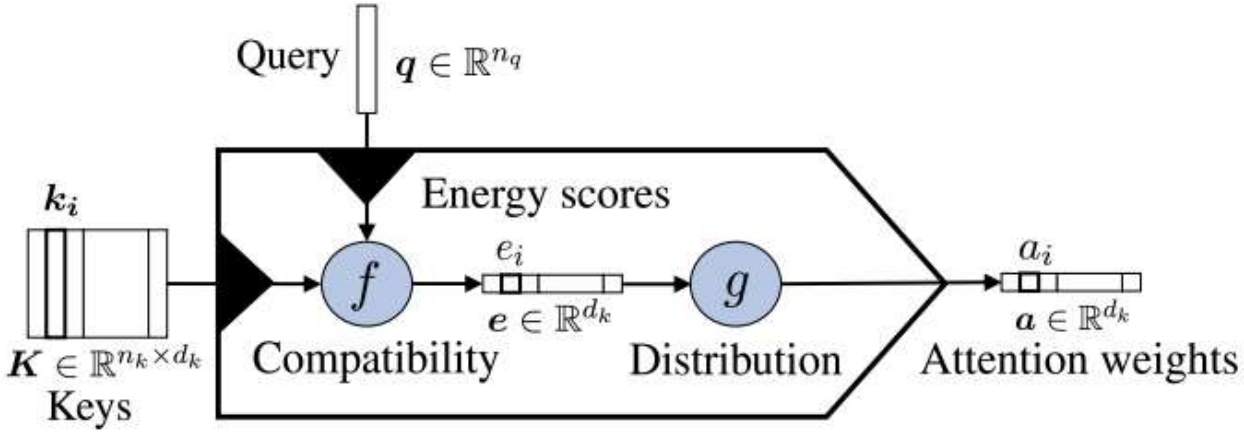


Fig. 3. Core attention model.

BERT: Bidirectional Encoding Representations from Transformers

- Why BERT?
  - Pre-trained, unsupervised, deeply bidirectional
  - Previous training methods for transformers required unidirectionality in their training objective (masking tokens)
  - Shallow bidirectionality is training two networks in opposing directions and then combining output in a final step, so context on both sides is never learned jointly
    - i hear voices all the time but now that im on a good combo of 800mg quetiapine and 20mg olanzapine, im doing better…the voices arent as bad as they can be
      - Is hearing voices a bad thing in this context? <- backward read
      - Is this patient on a high dose of quetiapine? -> forward read
      - Understanding the effect of this med? Need both directions
  - Deep bidirectionality is training a neural net to leverage context on both sides of a target token from the very bottom

How is BERT Trained?

- Masking
  - *Mask* 15 percent of input tokens, run through the encoder stack of a transformer, and attempt to predict masked words
    - Various tricks within this masking procedure; 80 percent get a MASK token, 10 percent get random word, 10 percent get original word
      - If we always used MASK, then model might not learn other tokens

```
Input: the man went to the [MASK1] . he bought a [MASK2] of milk.
Labels: [MASK1] = store; [MASK2] = gallon
```

- Next Sentence Predictions
  - Given a pair of sentences, does sentence B follow sentence A?
  - Useful for NLP inference (Q&A), not that useful for paraphrase detection or sentiment analysis.

```
Sentence A: the man went to the store .
Sentence B: he bought a gallon of milk .
Label: IsNextSentence
```

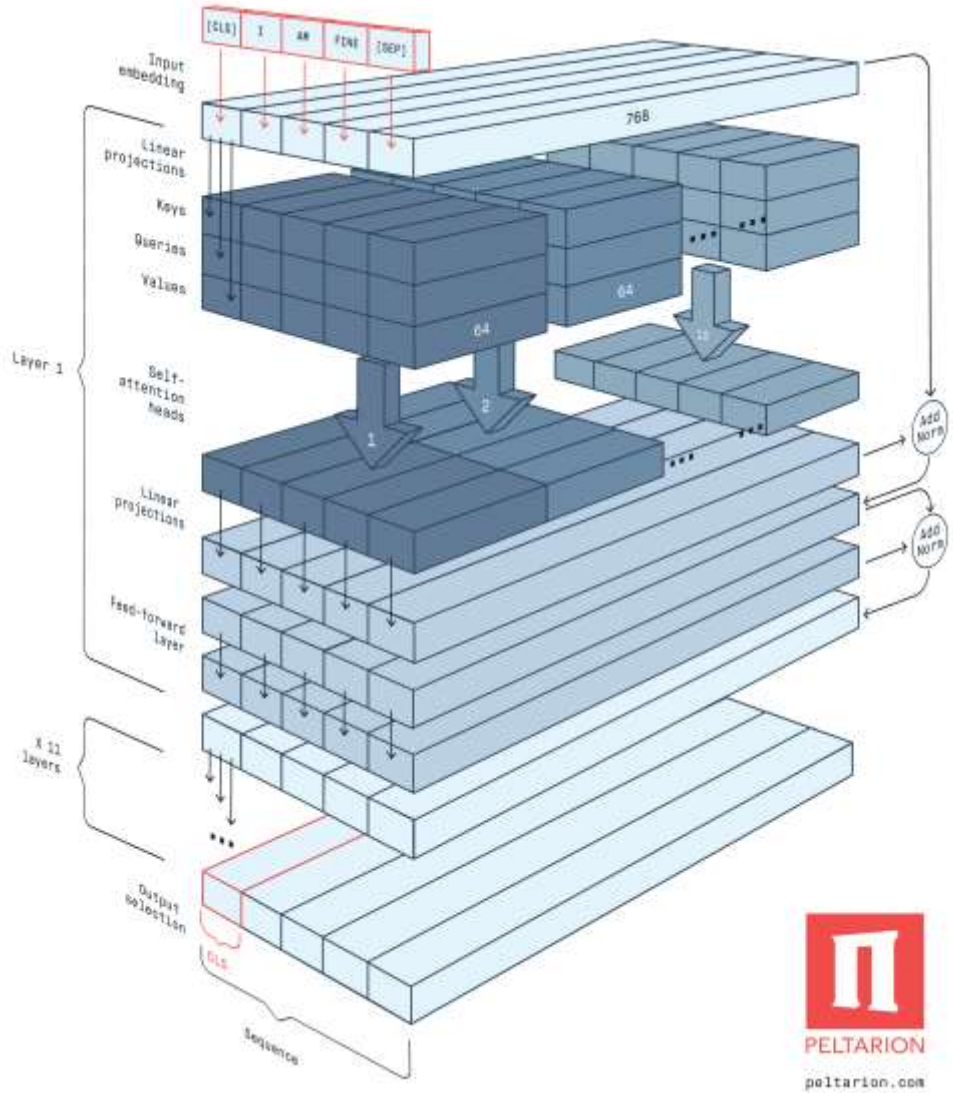
```
Sentence A: the man went to the store .
Sentence B: penguins are flightless .
Label: NotNextSentence
```

https://github.com/google-research/bert

BERT Components

- Embedding Layer
  - Also learns positional encoding
  - Dropout for regularization
- Encoder Layer
  - Multi-Head Self Attention
    - K,Q,V Projections
  - Feed Forward Network
  - Add and Norm
    - Residual Connections
      - Skip non linearities than can create exploding/vanishing gradients
    - LayerNorm
      - Scale of inputs changes as network learns weights, can slow down learning
- Outputs
  - The actual contextually dependent BERT embeddings

BERT


[CLS]
I
AM
FINE
[SEP]
Input embedding
768
Linear projections
Keys
Queries
Values
64
64
64
12
Layer 1
Self-attention heads
1
2
Linear projections
Add Norm
Add Norm
Feed-forward layer
X 11 layers
Output selection
CLS
Sequence
PELTARION
peltarion.com

BERT/NLP Topics for the Future

- Positional Embedding, Dropout, LayerNorm in Depth
- Mathematics of Model Distillation (DistilBERT)
- What are Different BERT Layers Actually Learning?
- Extending BERT to Longer Sequences