TBSI 清华-伯克利深圳学院 Tsinghua-Berkeley Shenzhen Institute

BERT: a pre-trained language model for Transformer

Bert: Pre-training of deep bidirectional transformers for language understanding

Presenter: Qiqi Chen

Content

1.1 Architecture of Transformer

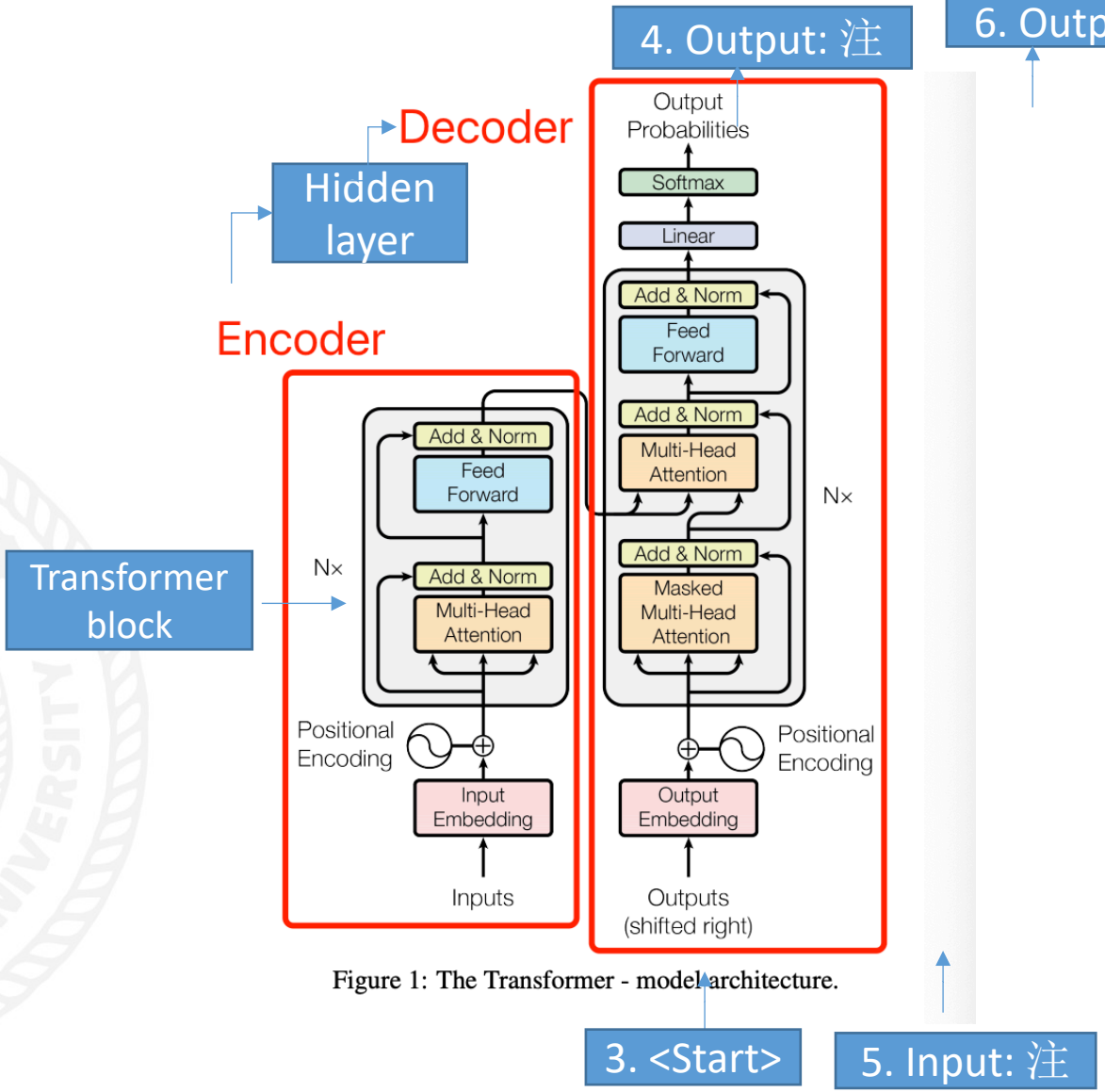


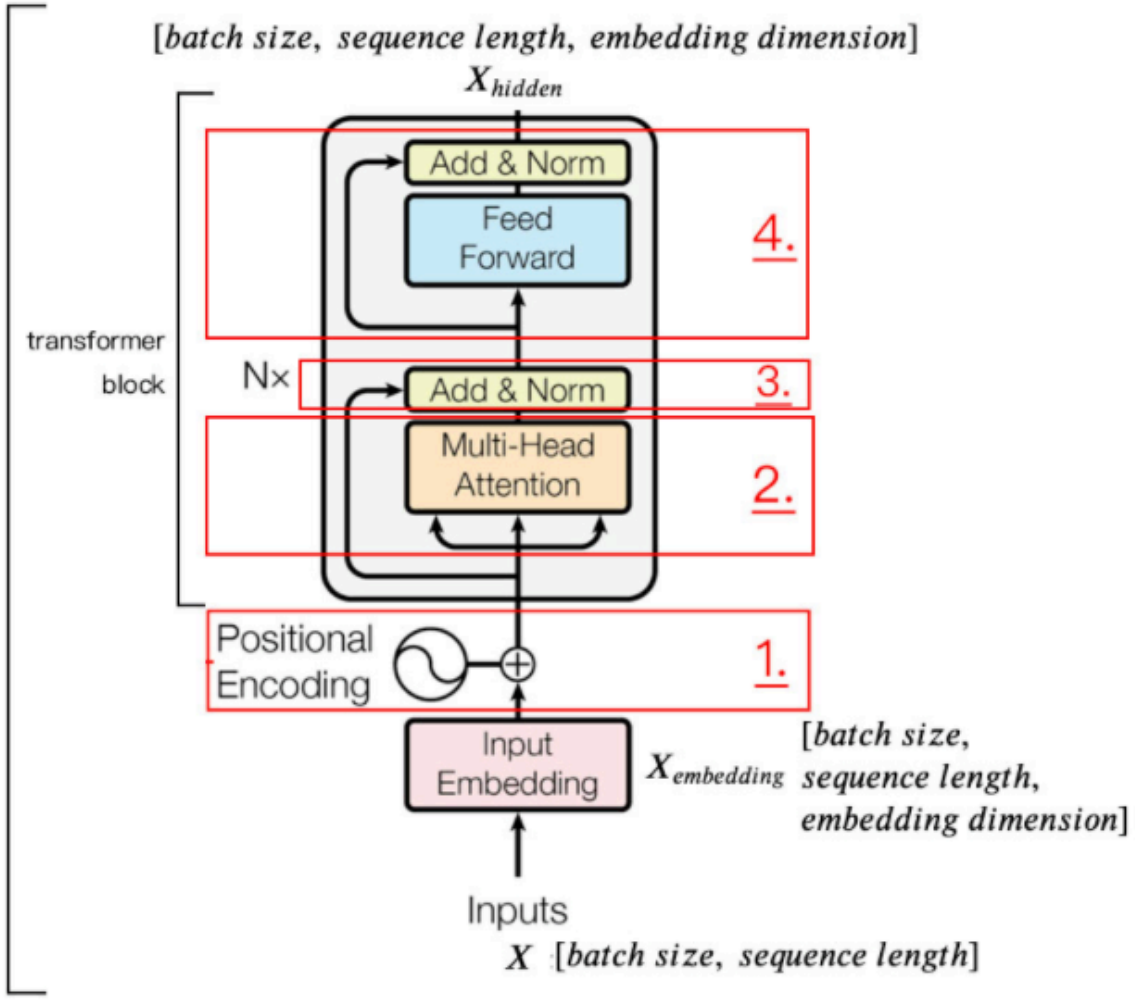
Figure 1: The Transformer - model architecture.

Eg. Translator

1. Input: Attention is all you need
2. Encoder outputs the hidden layer, then as the input to the decoder;
3. Input <start> token into the decoder
4. Output the first word “注”;
5. Input “注” into the decoder;
6. Output the second word “意”
7. Repeat 5-6 until the decoder outputs <end> token.

Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. arXiv preprint arXiv:1706.03762, 2017.

1.2 Transformer encoder



Encoder:

The process by which natural language sequences are computed into a hidden layer

Batch size:

# of sequences(sentences)

1.2.1 Positional encoding

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

Dimension:

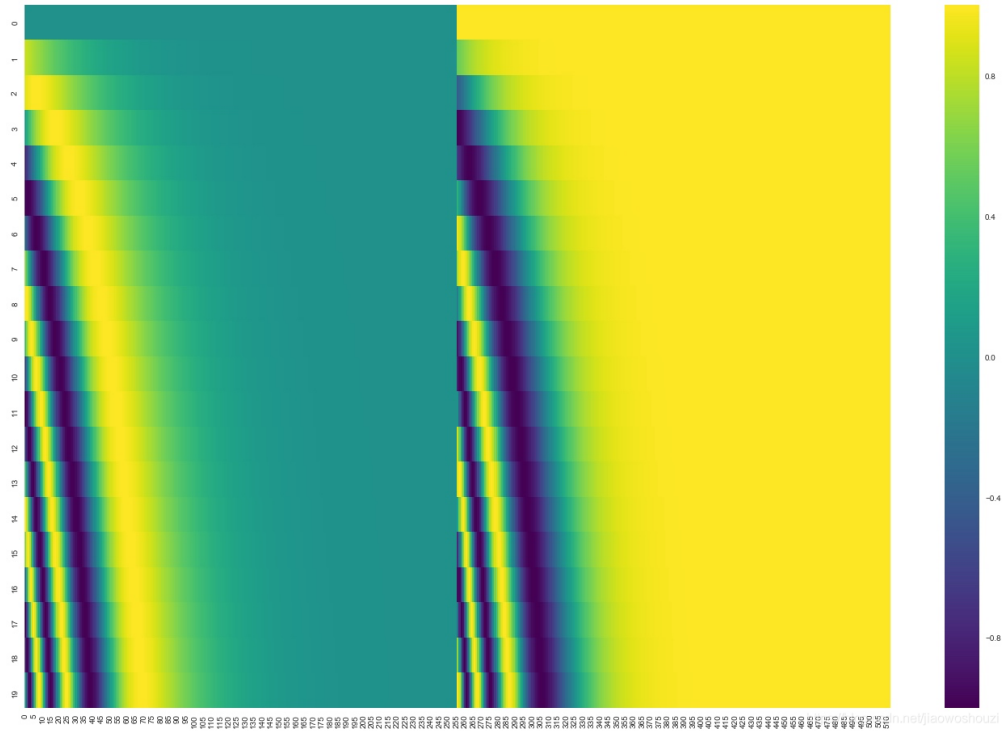
[max sequence length, embedding dimension]

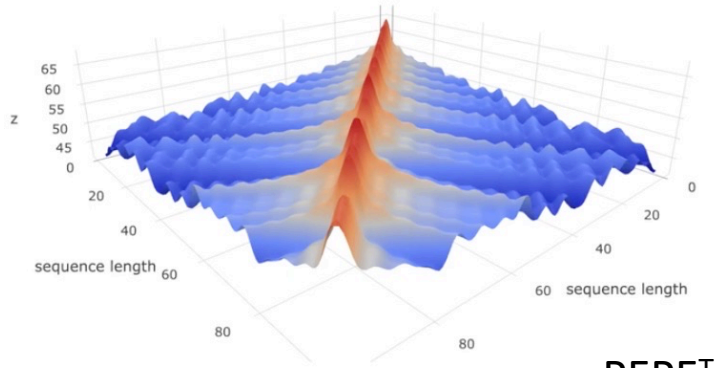
pos: the position index of the word

[0, max sequence length)

i: the dimension index of the word embeddings

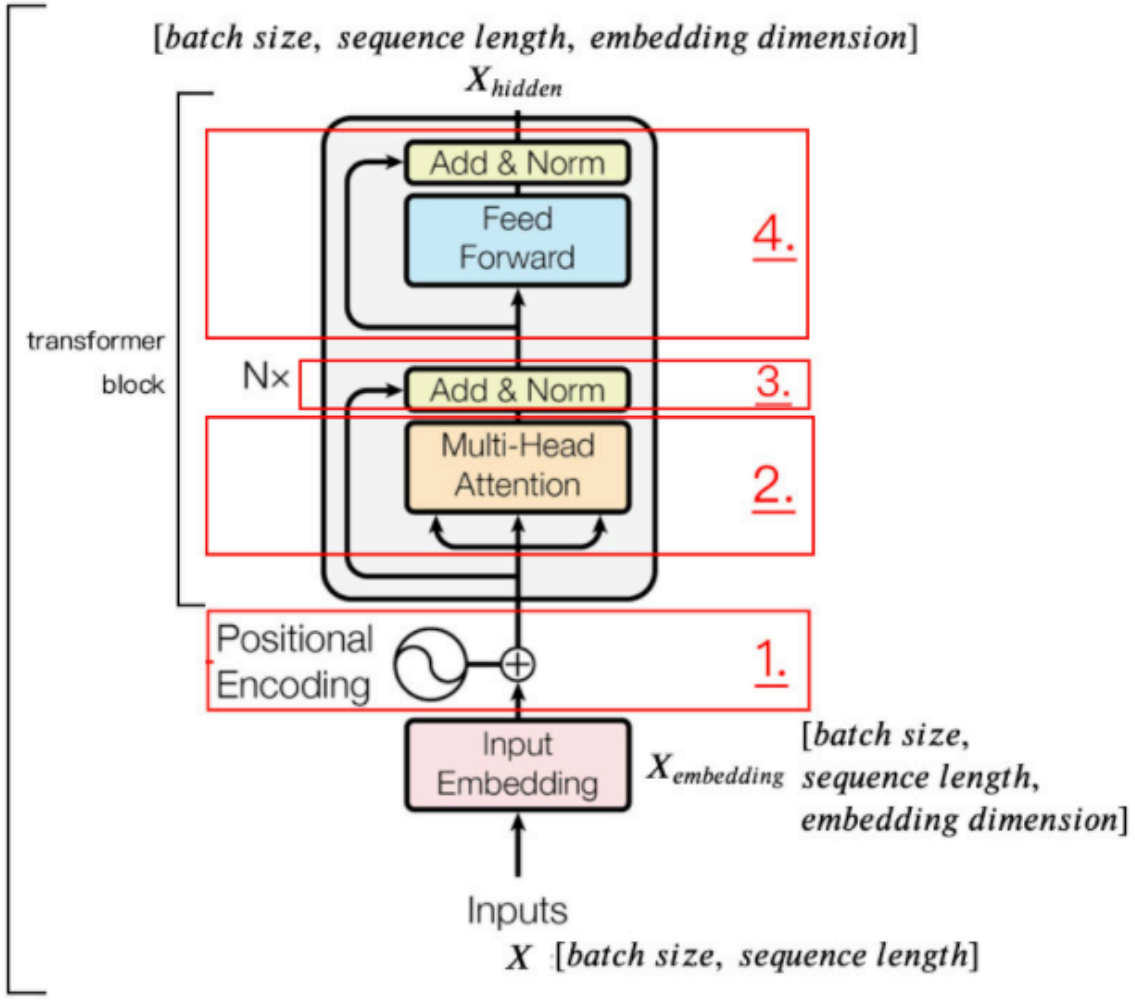
[0, embedding dimension)





PEPE^T

1.2 Transformer encoder



Encoder:

The process by which natural language sequences are computed into a hidden layer

Batch size:

# of sequences(sentences)

1.2.2 Self attention mechanism

Input:

$$X \in \mathbb{R}^{batch\ size * seq.\ len.}$$

$$X_{embedding} = Embedding Lookup(X) + Positional Encoding$$

$$X_{embedding} \in \mathbb{R}^{batch\ size * seq.\ len. * embed.\ dim.} \quad \text{(eq.2)}$$

Step 1: assign 3 weights W_Q, W_K, W_V for Linear mapping

Each word has three different vectors:

$$Q = Linear(X_{embedding}) = X_{embedding} W_Q$$
$$K = Linear(X_{embedding}) = X_{embedding} W_K$$
$$V = Linear(X_{embedding}) = X_{embedding} W_V$$

Query vector (Q), Key vector (K), and Value vector (V)

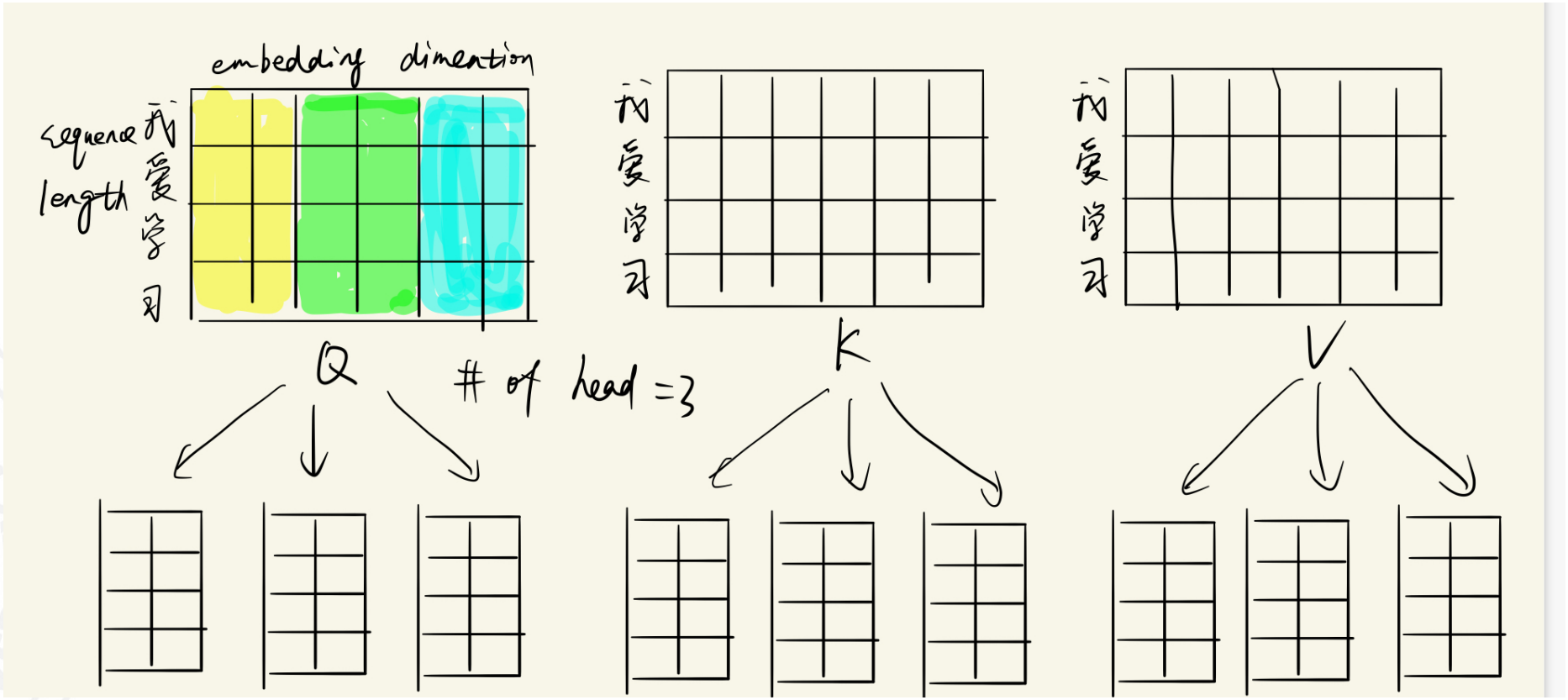
Step 2: $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

Score each vector

To stabilize the gradient, Transformer uses normalization, i.e., dividing by $\sqrt{d_k}$

[sequence length, embedding dimention]

1.2.3 Multi head Attention

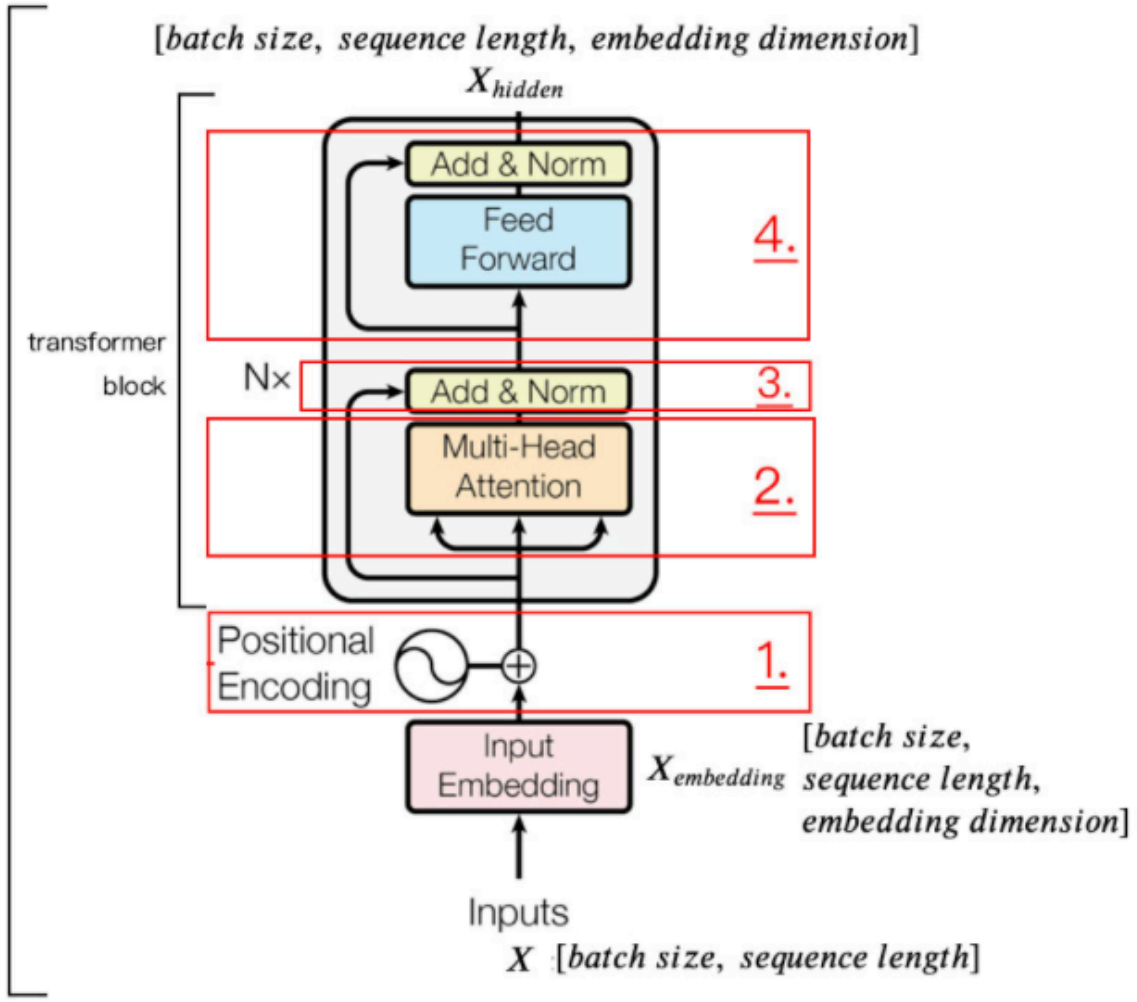


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

hyper-parameter:
# of heads

[batch size, sequence length, embedding dimension/h]

1.2.4 Add&Norm and Feed forward



Residual connection(Add)

$$X_{embedding} + Attention(Q, K, V)$$

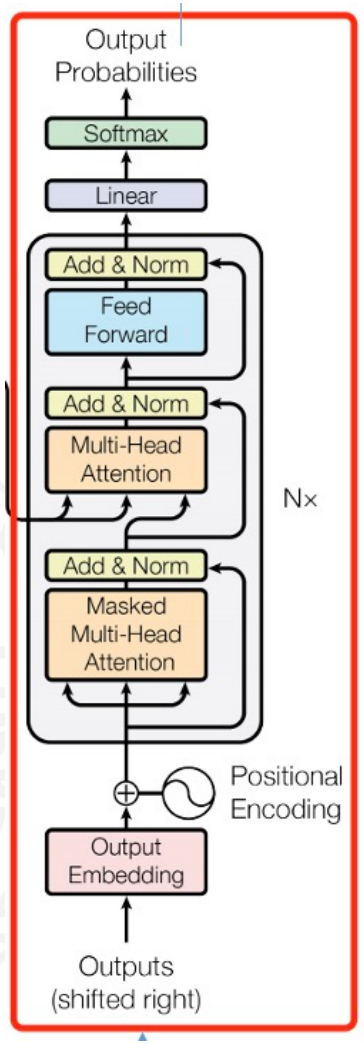
Layer normalization:
- Normalized to the standard normal distribution
- Accelerate the training speed and convergence

Feed forward Neural Network :

Two linear mapping layers + activation function like ReLU

$$X_{hidden} = Activate(Linear(Linear(X_{attention})))$$

1.3 Transformer decoder



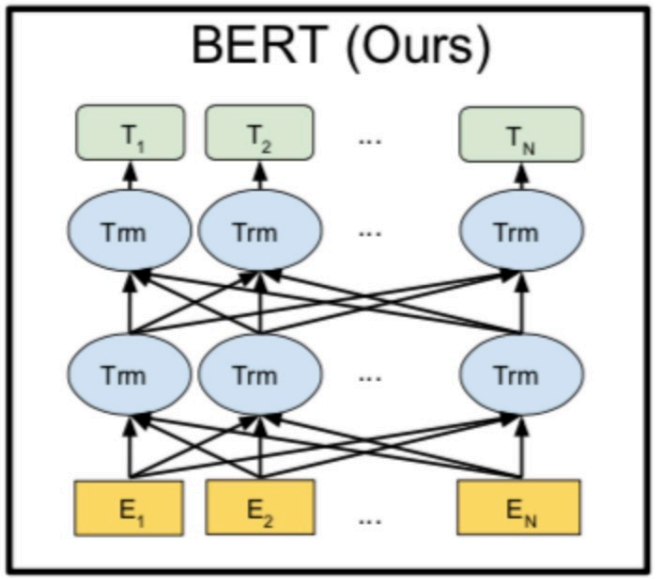
- **Similar to encoder**
- Padding mask

  Filling –inf (because softmax(-inf)≈0) after short sequence
- Sequence mask

  At time t, decoding output can only be output before time t

  Hiding the information after t.

2 BERT



$$BERT_{BASE} : L = 12, H = 768, A = 12, TotalParameters = 110M$$

$$BERT_{LARGE} : L = 24, H = 1024, A = 16, TotalParameters = 340M$$

L is the number of layers in Transformer

H is the output dimensions

A is the number of mutil-head attention

E_N is a single word
T_N is the final calculated hidden layer

The operation of the attention matrix and the attention weighting:
Each word in the sequence contains information before and information after the word

2.1 The pre-training process of BERT

Task 1: **Masked Language Model --** like 完形填空 in Chinese：)

- Step 1: Randomly mask 15% of the words in each sentence,

1) 80%: replaced by [mask]
   My dog is hairy -> My dog is [mask]
2) 10 %: replaced by any other token
   my dog is hairy -> my dog is apple
3) 10 %: no change.
   my dog is hairy -> my dog is hairy

- Step 2: Predict the content of the masked part

The weight of the mapping layer W_{vocab}

$X_{hidden}W_{vocab}$ = [batch_size, seq_len, embedding_dim] · [embedding_dim, vocab_size]
= [[batch_size, seq_len, vocab_size]

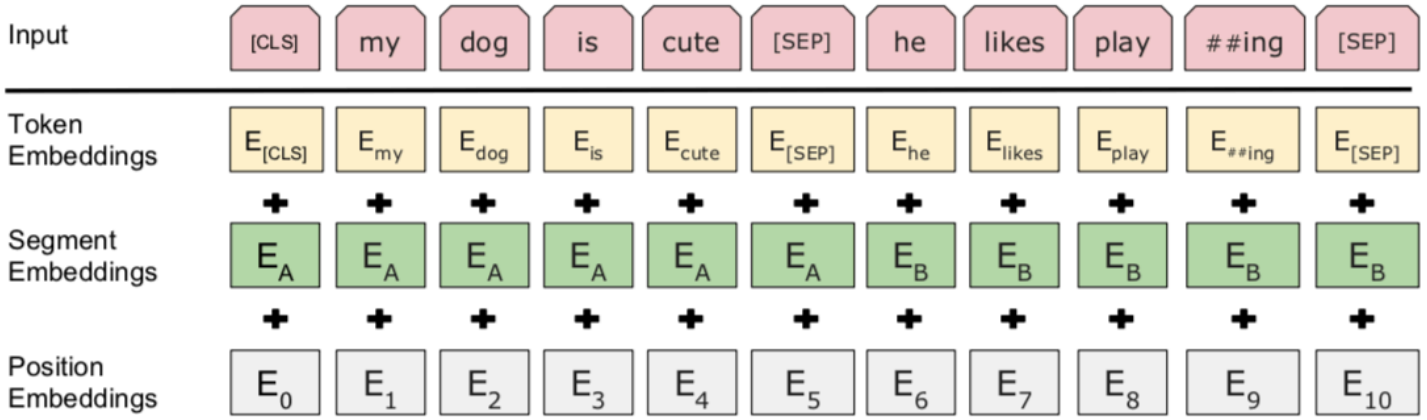
Through softmax normalization, the sum of each word corresponding to VOCAB_SIZE is 1
The prediction results of the model are obtained by the word with the highest probability in VOCAB_SIZE

2.1 The pre-training process of BERT

Task 2: Next Sentence Prediction (input is sentence pair)

Step 1: token embedding
Step 2:
[cls]My dog is cute [sep] bird can fly [sep]

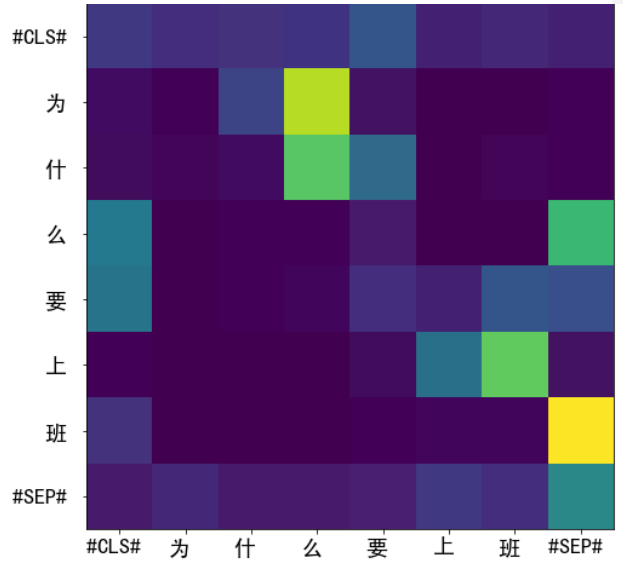
Step 3: segment embedding
[cls]My dog is cute [sep] bird can fly [sep]
0 0 0 0 0 1 1 1 1 1



Step 4: Attention mechanism
X_{hidden} = [batch_size, seq_len, embedding_dim]

All information in this sentence is expected to be summarized into a vector corresponding to the [CLS]token
cls_vector = X_{hidden}[:, 0, :] ([batch_size, embedding_dim])

y = sigmoid(Linear(cls_ vector))
y∈(0, 1)



2.2 The application results of BERT

The author fine-tuning the 11 NLP tasks, and achieved the performance of state-of-the-art

- NER

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

Table 3: CoNLL-2003 Named Entity Recognition results. The hyperparameters were selected using the Dev set, and the reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

- The Standford Question Answering Dataset (SQuAD)

System	Dev EM	Dev F1	Test EM	Test F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

3 Conclusion

- A deep bi-directional Transformer language model is adopted
- The training depth bi-directional pre-training model was achieved by Mask LM
- Compared with the unidirectional language model, the training results are more accurate and semantic understanding is more accurate

- Performance is not good when multiple words are masked and there is a relationship between the words

XLNet VS BERT

- **Autoencoder LM(BERT)**

  MASK problem but bidirection
- **Autoregressive LM(ELMO)**

  Unidirection but no need MASK
- **XLNet**

  By arranging and combining words in sentences

  Put some of the words after Ti in the place before Ti

Reference

Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. arXiv preprint arXiv:1706.03762, 2017.

Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

Question: how does it related to other recent language models, e.g. GPT-3?

- **Architecture:**

  GPT-3 is built using transformer decoder blocks and BERT uses transformer encode blocks

  BERT uses Self Attention and GPT3 uses Masked Self Attention.
- **Parameters**: BERT largest model has 340 Million parameters and GPT 3 is 175 billion Training
- **Learning approach**: BERT has pre-trained models for different downstream. NLP tasks which are further fine-tuned on custom data

  GPT3 has a single model for all downstream tasks and does not require fine-tuning

  GPT 3 Learns from examples through zero shot, one shot or few shot approach