Transformer

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

Topics

- Transformer: what & why
- Background concepts
  - Recap: RNN, LSTM
  - Encoder-decoder architecture
  - Attention
- Transformer Model
  - Transformer Attention & Q, K, V
  - Self-Attention
  - Multi-Head Attention
  - Positional Encoding
  - Masked Attention
- Language Model

Transformer Model

- Original paper:
  - “Attention is All You Need” by Vaswani *et al.,* NEURIPS, 2017
    - based on “Attention” idea from Bahdanau *et al*., ICLR, 2015
  - “Self-Attention” mechanism

- Why Transformer?
  - Big improvement from RNN, LSTM
  - Recent SOTA NLP models are all transformer-based
    - BERT >> ALBERT, RoBERTA, WangchanBERTa >> GPT
  - Also applicable to vision tasks (ViT)

Main Contributions of Transformer Paper

- More parallelizable than RNN
- Much fewer # of operations than CNN-based solutions
- SOTA machine translation
  - Fast training time
  - SOTA BLEU scores on Eng-Ger, Eng-Fr

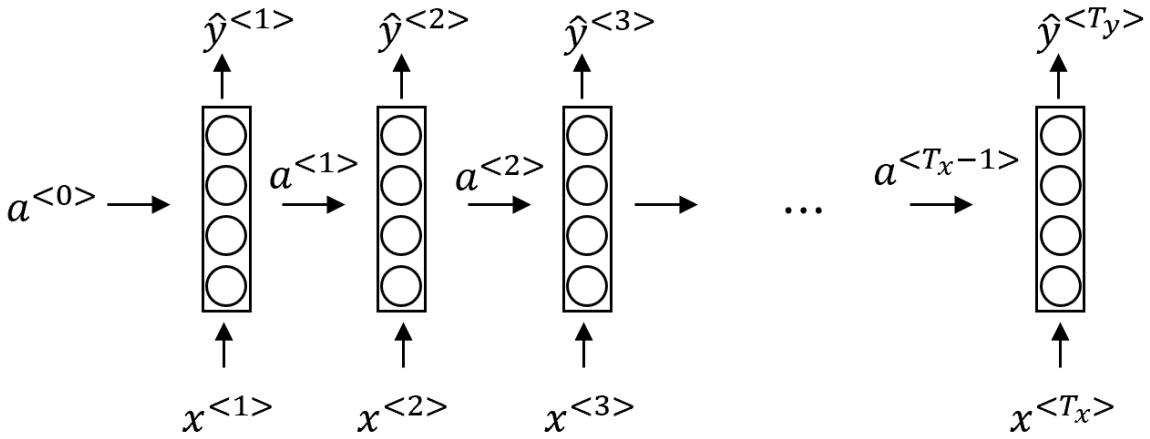
Background Concepts

- RNN, LSTM
- Encoder-Decoder Model
  - Sequence to Sequence Learning with Neural Networks (Cho et al., 2014)
    - https://arxiv.org/abs/1409.3215
  - Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (Sutskever et al., 2014)
    - https://arxiv.org/abs/1406.1078
- Attention Mechanism
  - Neural Machine Translation by Jointly Learning to Align and Translate (Bahdanau et al., 2015)
    - https://arxiv.org/abs/1409.0473

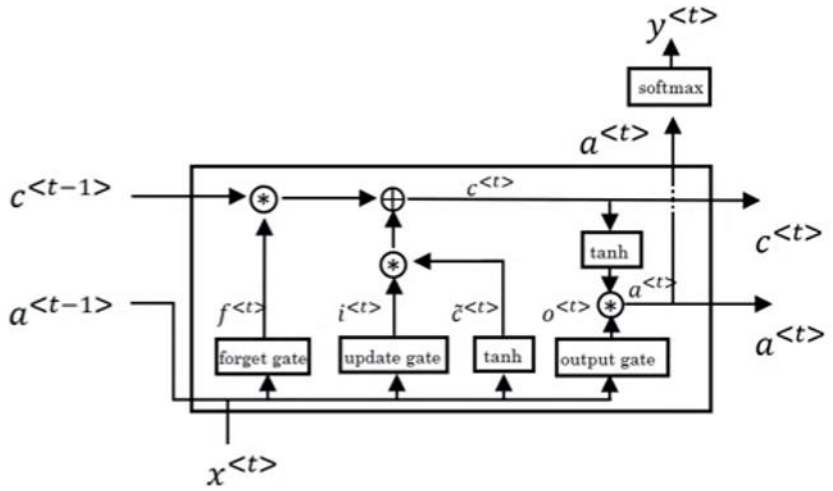
https://www.coursera.org/learn/nlp-sequence-models/

Quick Recap

RNN

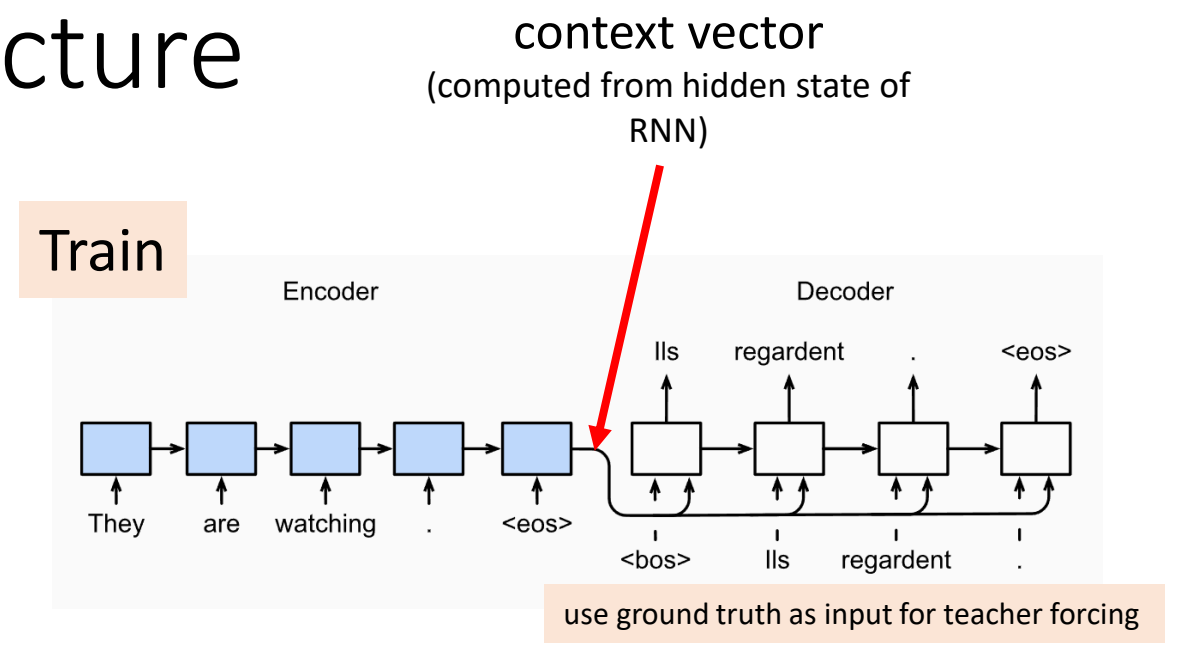


LSTM



Encoder-Decoder Architecture

- “seq2seq” model
- Encoder
  - Encodes the entire input sequence into "context vector" (representation of input sequence)
- Decoder
  - Generates output based on the context vector
- Train both parts at once (End-to-End)

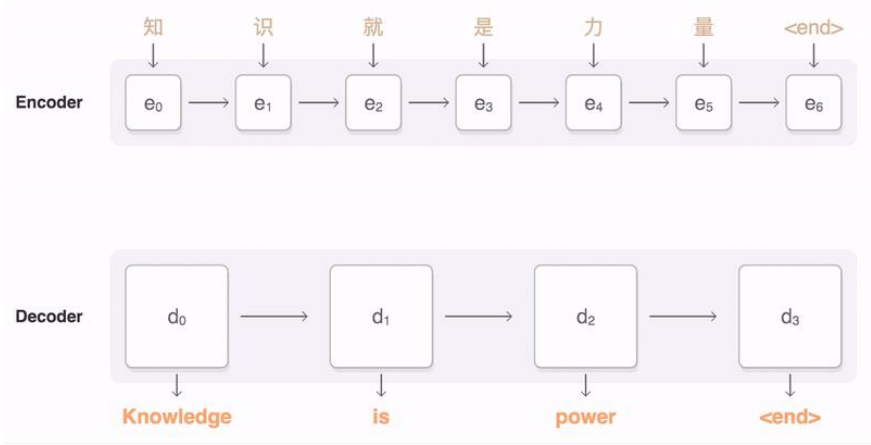


https://d2l.ai/chapter_recurrent-modern/seq2seq.html

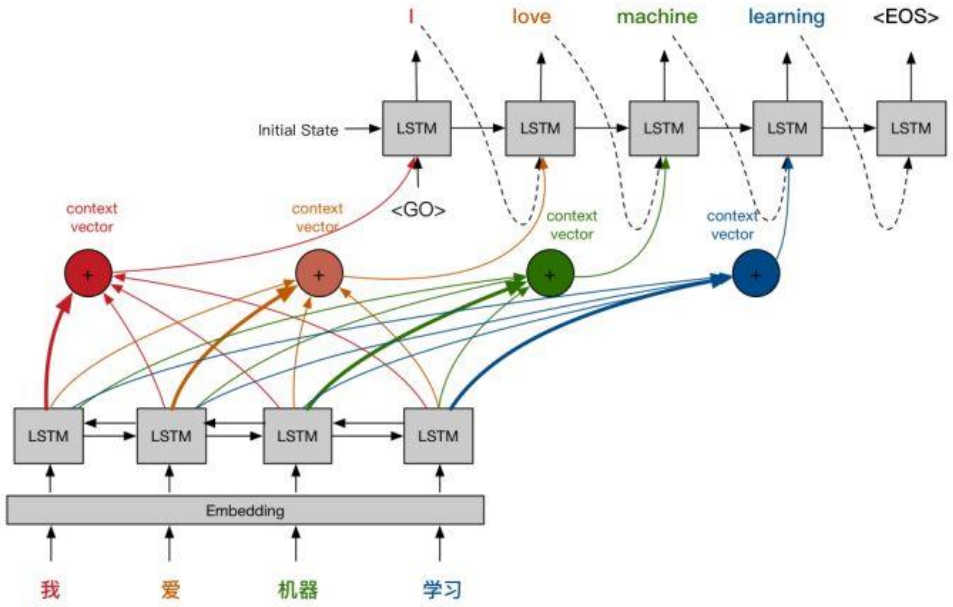
Attention

- Limitation of Encoder-Decoder
  - Context vector C is fixed-length
    - Does not work well with long input sentences
  - Alignment problem in MT:
    - Which parts of input sentence should the decoder concentrates on?

- Attention (Bahdanau et al., 2015)
  - Use weighted/combined context vectors from many timesteps of the encoder
  - Weight = attention given to that input word



https://github.com/google/seq2seq

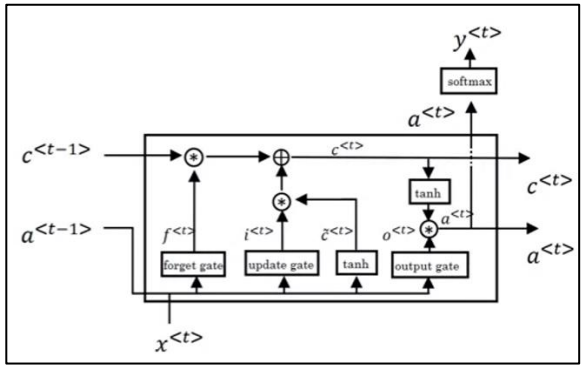


https://zhuanlan.zhihu.com/p/37290775

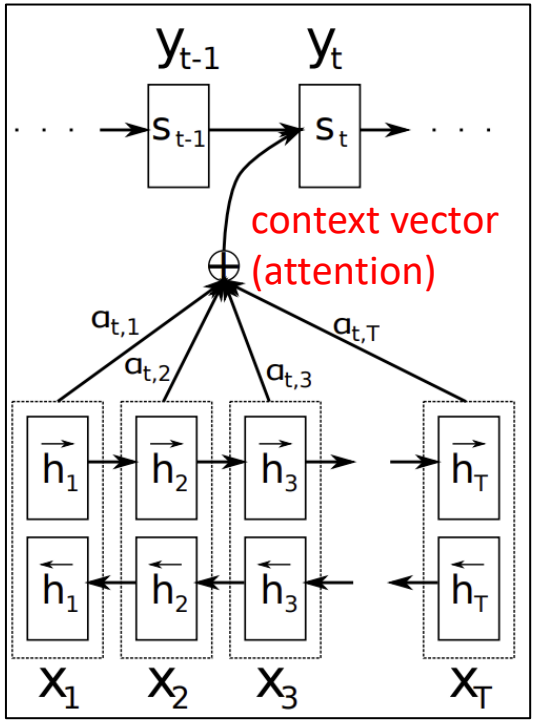
How to compute attention?
(Bahdanau, 2015)

- ทบทวน RNN notation:
  - $a^{<t>}$ คือ activation (= hidden state $h^{<t>}$) ของ LSTM, GRU

- Encoder-decoder notations:
  - ในส่วน encoder ใช้ Bidirectional RNN
    - ซึ่งในแต่ละ step t' มี forward $\vec{h}^{<t'>}$ และ backward $\overleftarrow{h}^{<t'>}$
  - ในส่วน decoder ใช้ RNN
    - ซึ่งในแต่ละ step t จะ generate คำตอบ $y^{<t>}$
      โดยนำ context vector $c^{<t>}$ จาก encoder มาร่วมคิดด้วย



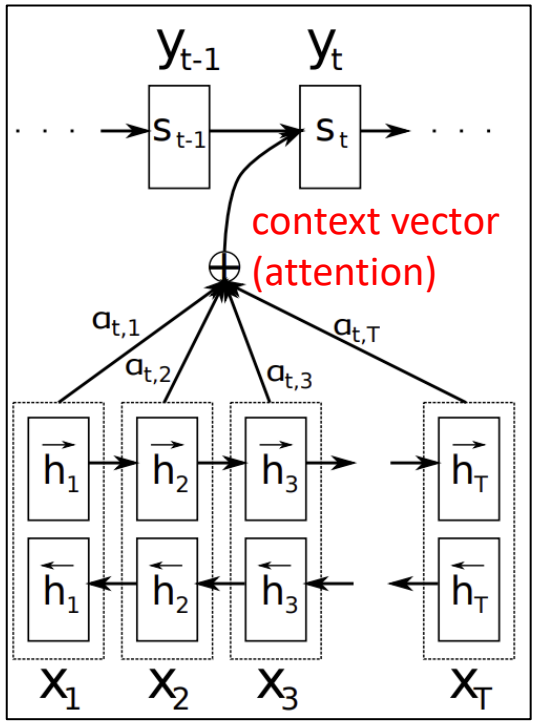
LSTM cell



https://arxiv.org/abs/1409.0473

How to compute attention?

(Bahdanau, 2015)



Attention Mechanism

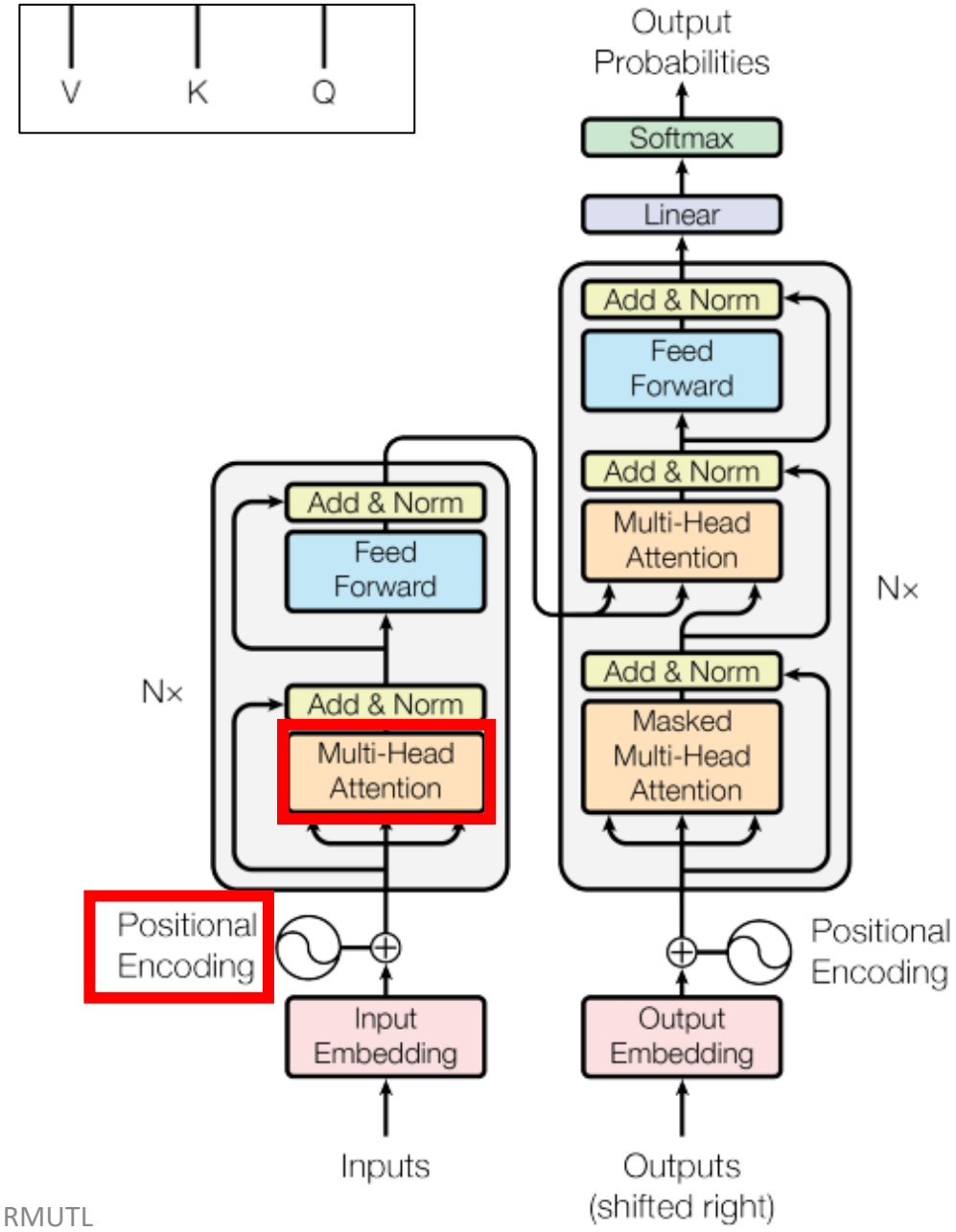
- การคำนวณ context vector $c^{<t>}$
  - $c^{<t>} = \sum_{t'=1}^{T_x} \alpha^{<t,t'>} [\vec{h}^{<t'>}, \overleftarrow{h}^{<t'>}]$
  - เป็นการสกัดข้อมูลที่เกี่ยวข้องมาจาก $\vec{h}^{<t'>}, \overleftarrow{h}^{<t'>}$ของทั้ง encoder sequence

- โดยที่ $\alpha^{<t,t'>}$ คือ attention score (decoder step t ควรให้ความสนใจกับ encoder step t' มาก/น้อย?)
  - $\alpha^{<t,t'>} = \text{softmax}(e^{<t,t'>})$ เพื่อ normalize ค่าให้อยู่ในช่วง 0-1
  - $e^{<t,t'>} = v^T \tanh(W_e[s^{<t-1>}, h^{<t>}])$ = 1-layer NN

https://arxiv.org/abs/1409.0473

Transformer Model
(Vaswani, 2017)

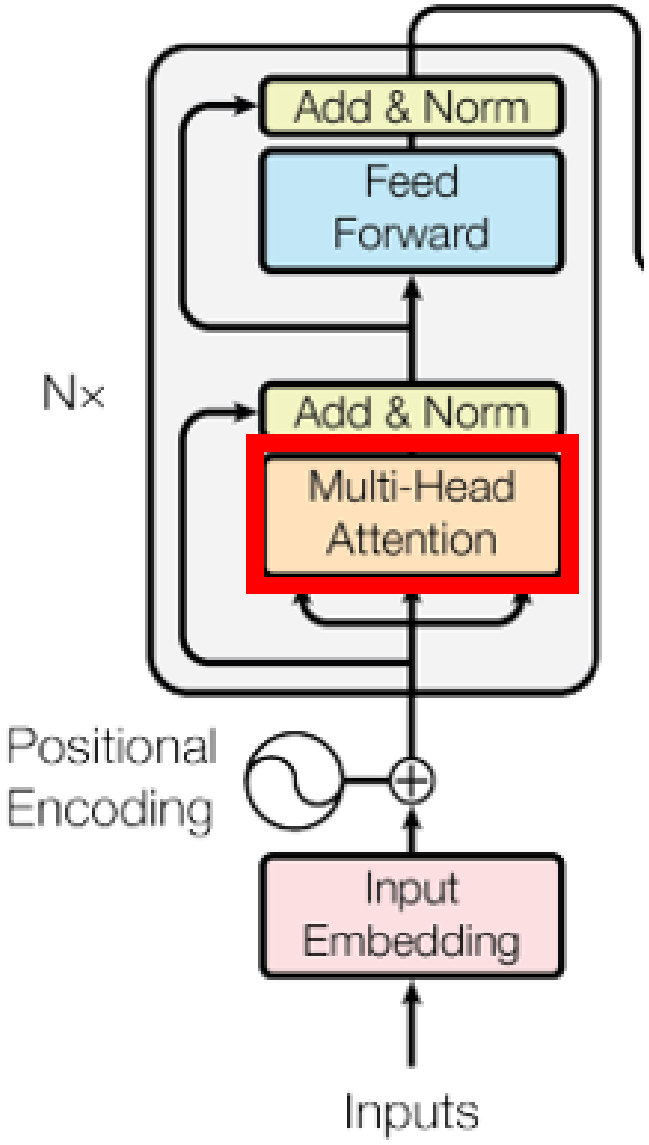
- No recurrence/convolution
- Main components:
  - Self-Attention => Multi-Head Attention
    - Allow model to attend to different parts of the input sequence
  - Positional Encoding
    - To maintain word ordering information
  - Residual Connections
    - To prevent vanishing gradient



Output from ALL previous steps of decoder?

The Core of Transformer

- To understand this block: we must first understand:
  - What are Q, K, V?
  - Transformer Attention $A(Q, K, V)$
  - Self-Attention
- Then, Multi-Head Attention is just h copies of Self-Attention



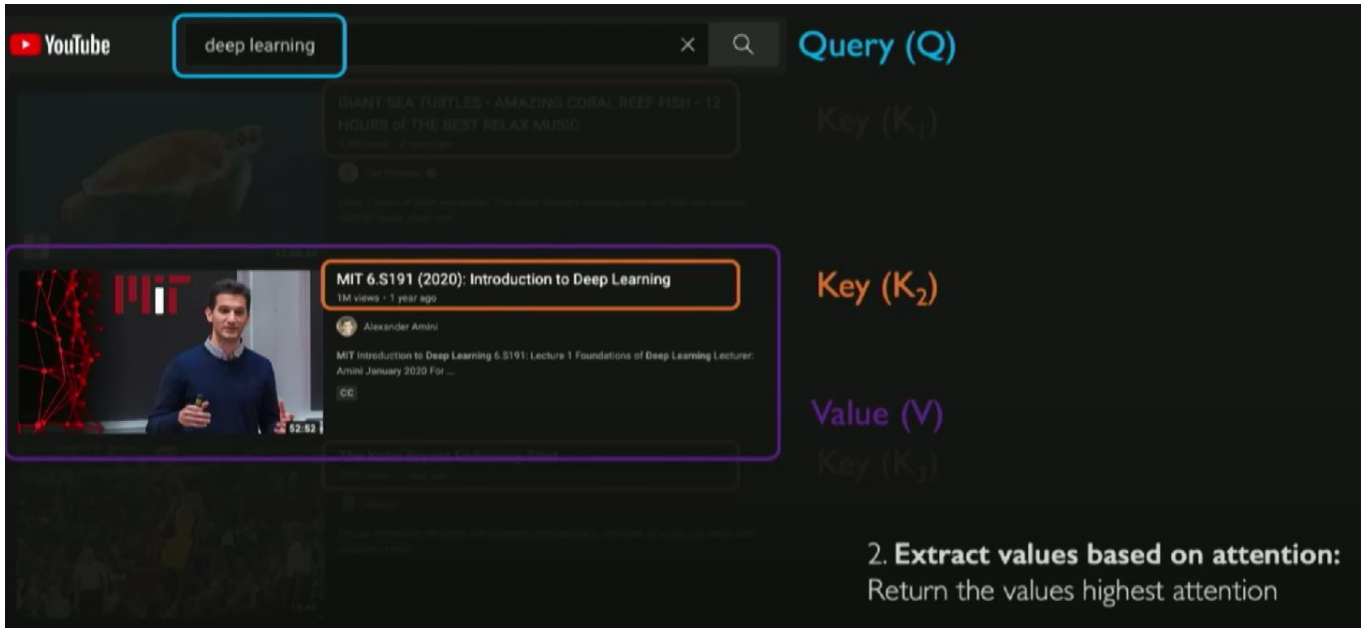
What are Q, K, V in Transformer?

In transformer, attention A is a function of Q, K, V

- $A(Q, K, V)$

Concept from Retrieval System

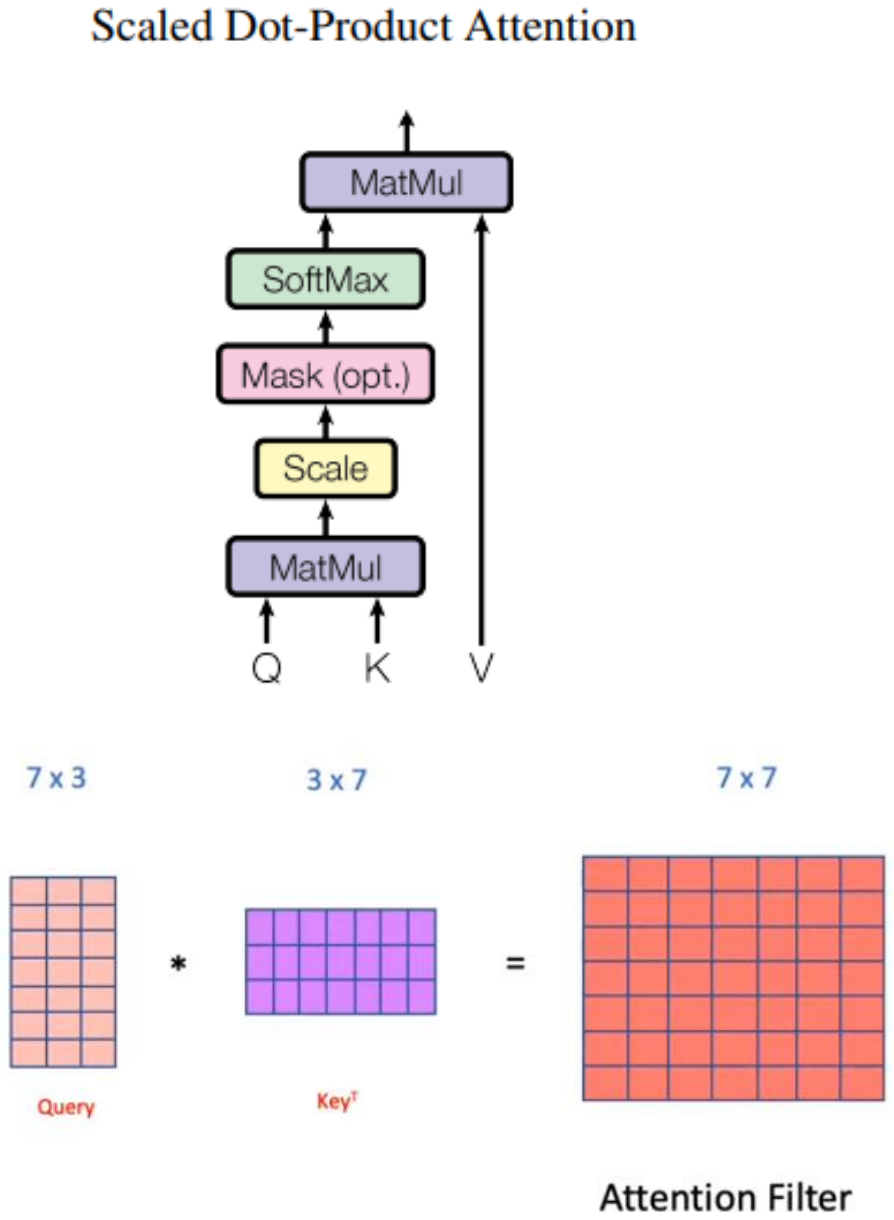
- Q (Query)
- K (Key)
- V (Value)



https://www.youtube.com/watch?v=ySEx_Bqxvvo

Transformer Attention

- Recap: RNN Attention
  - $\alpha^{<t,t'>} = \text{softmax}(e^{<t,t'>})$
- Transformer Attention
  - $A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
    - d_k is dimension of K
  - also called “scaled dot-product attention”
- Explanation
  - QK^T (attention filter):
    - look up keys that are closest to query
  - times V:
    - get V that corresponds to that key



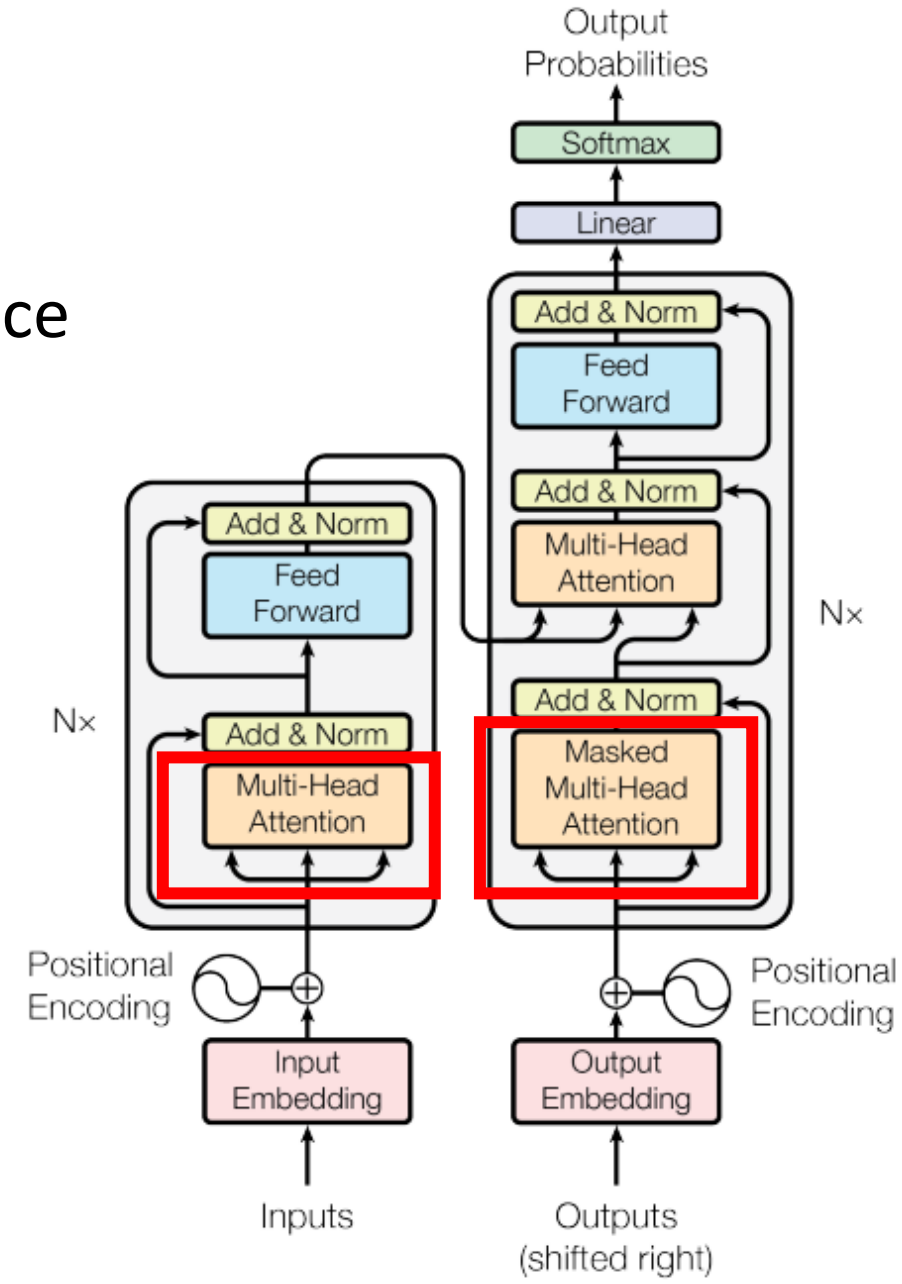
Self-Attention

Basic idea: to build representation of input sequence that captures relation among input words

Example:

He went to the bank and learned of his empty account, after which he went to a river bank and cried.

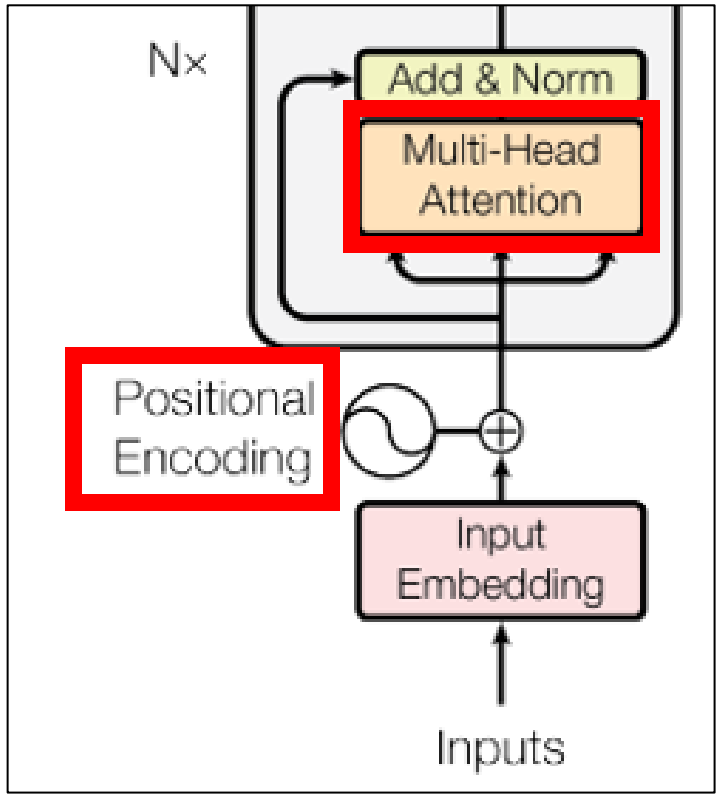
	When	you	play	the	game	of	thrones
When	89	20	41	10	55	10	59
you	20	90	81	22	70	15	72
play	41	81	95	10	90	30	92
the	10	22	10	92	88	40	89
game	55	70	90	88	98	44	87
of	10	15	30	40	44	85	59
thrones	59	72	92	90	95	59	99



https://www.youtube.com/watch?v=mMa2PmYJlCo

Self-Attention

- To compute self-attention of an input sequence:
  - First, input sequence is converted into word embeddings
    - e.g. word2vec
  - Add positional encoding to the embeddings
    - so that the attention block knows the word ordering
    - use clever sine/cosine function
  - Apply linear projection W^Q, W^K, W^V to the embeddings X
    - $Q = XW^Q$
    - $K = XW^K$
    - $V = XW^V$
  - Compute attention $A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$



positional encoding

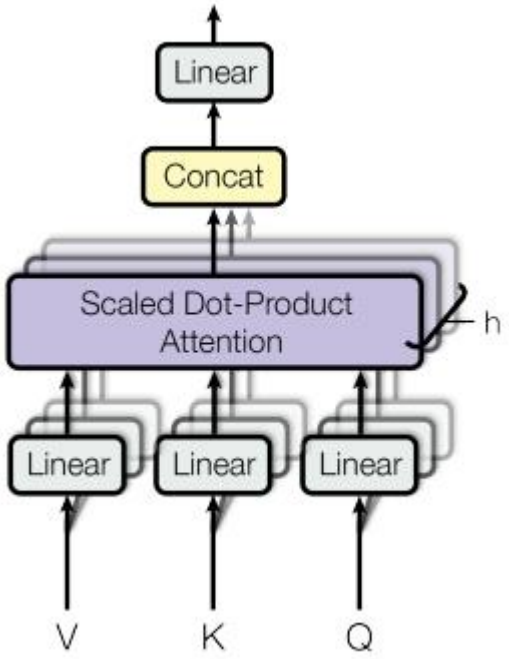
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Multi-Head Attention

Each head can detect different feature of the language

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^o$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$



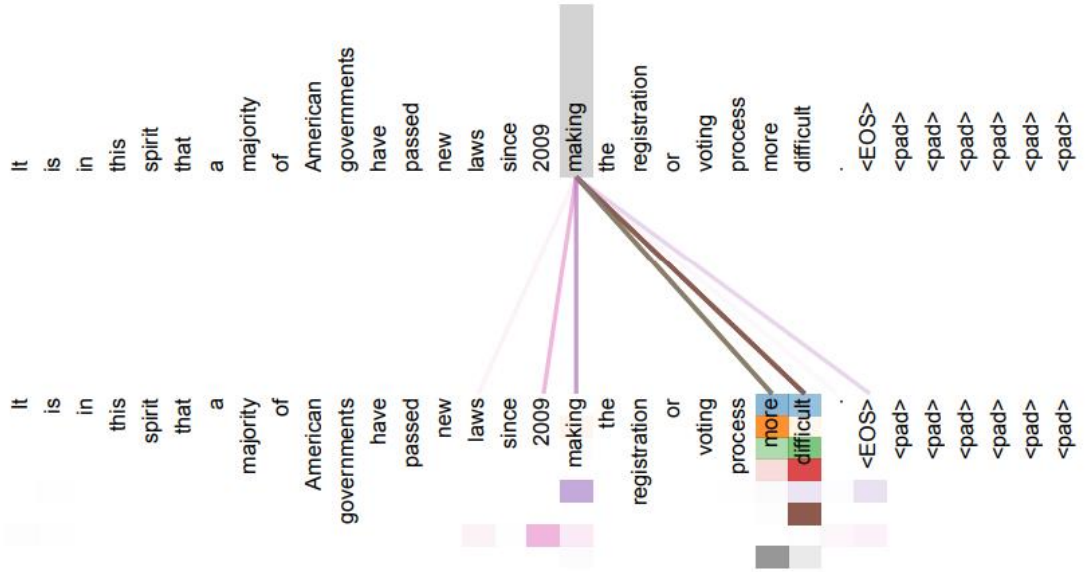
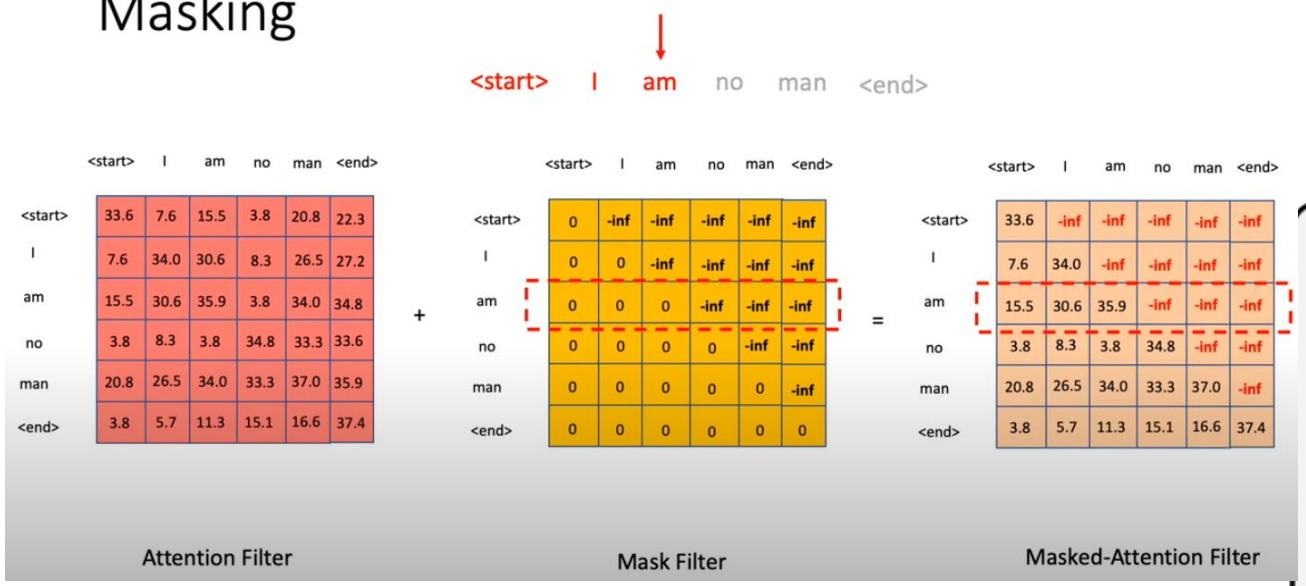


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

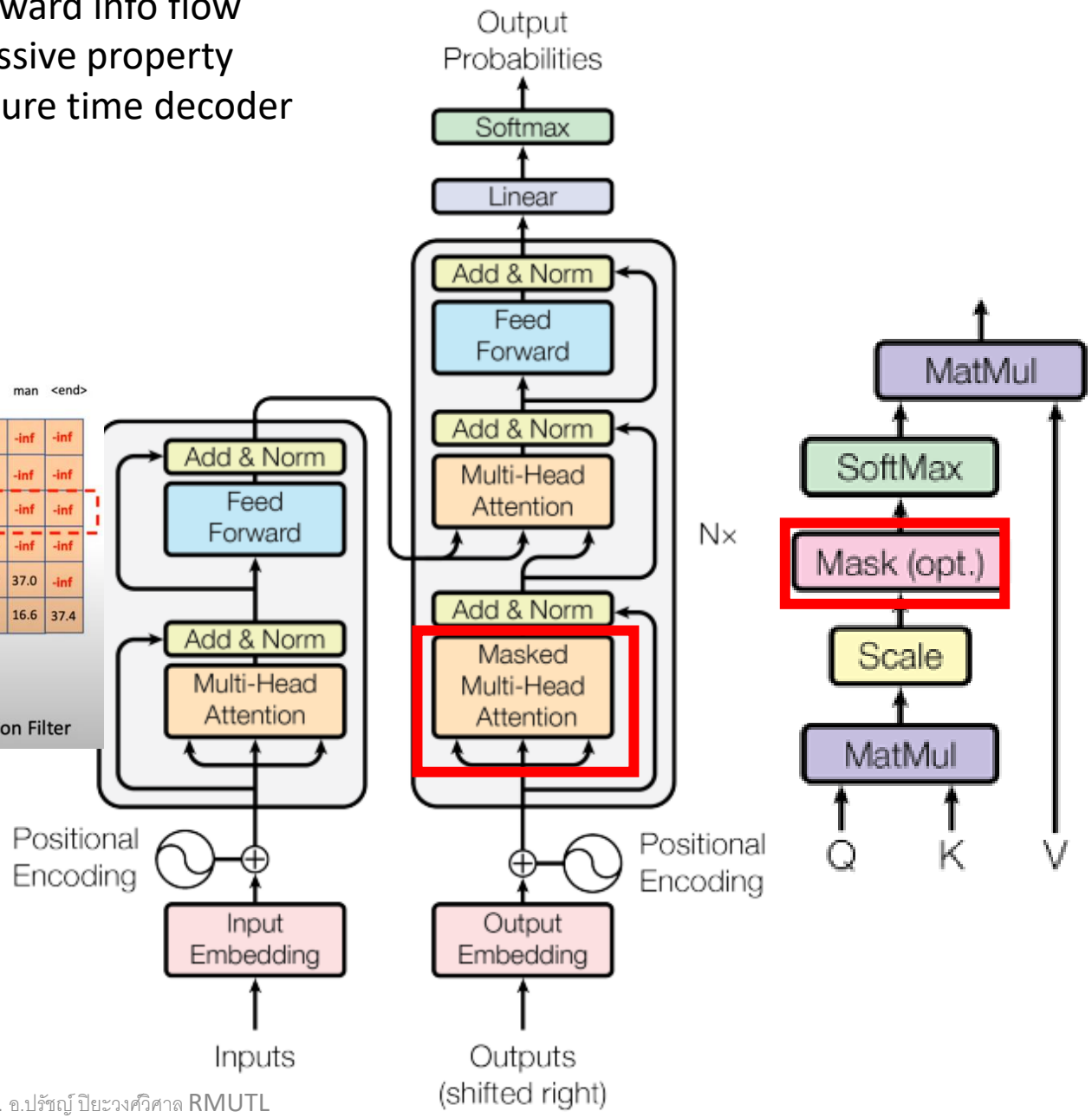
Vaswani, 2017

Masking

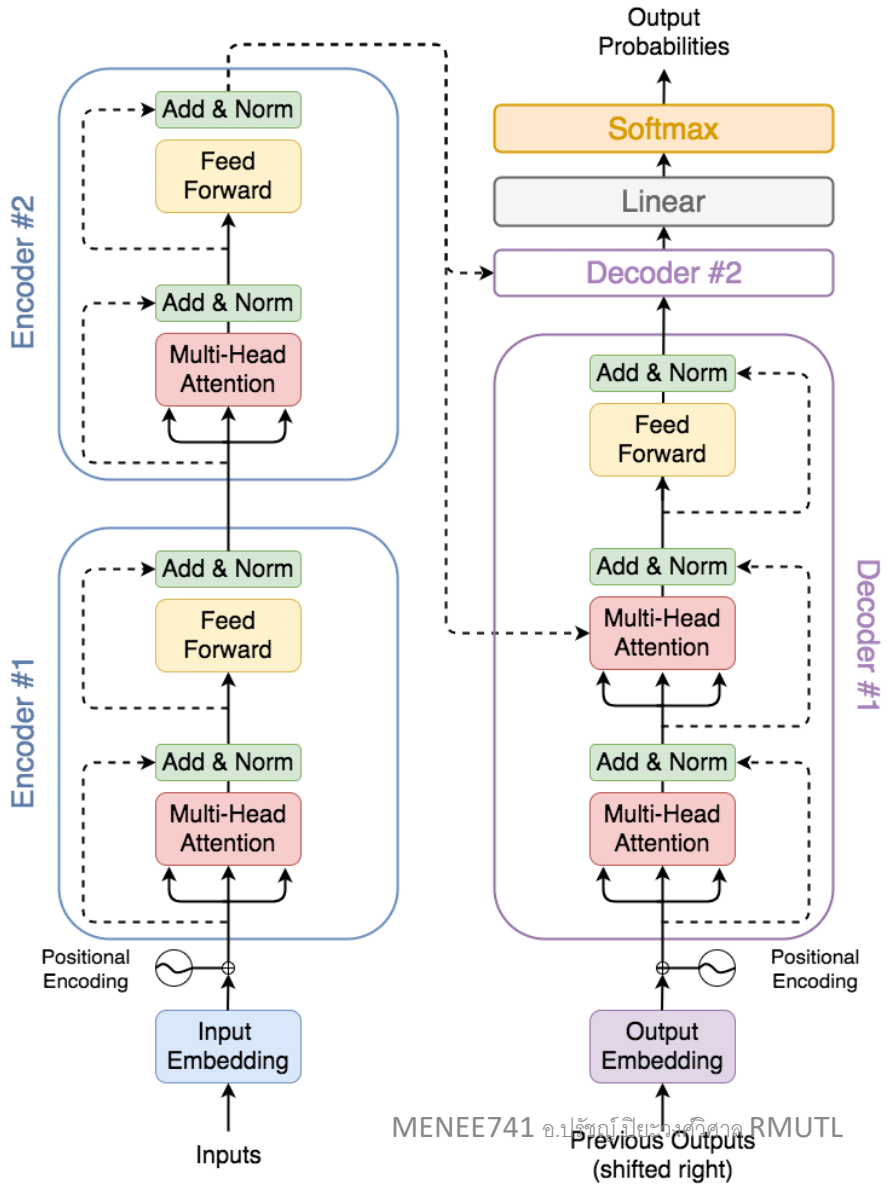
Masked to prevent leftward info flow to preserve auto-regressive property (aka. disallow using future time decoder output)



https://www.youtube.com/watch?v=gJ9kaJsE78k



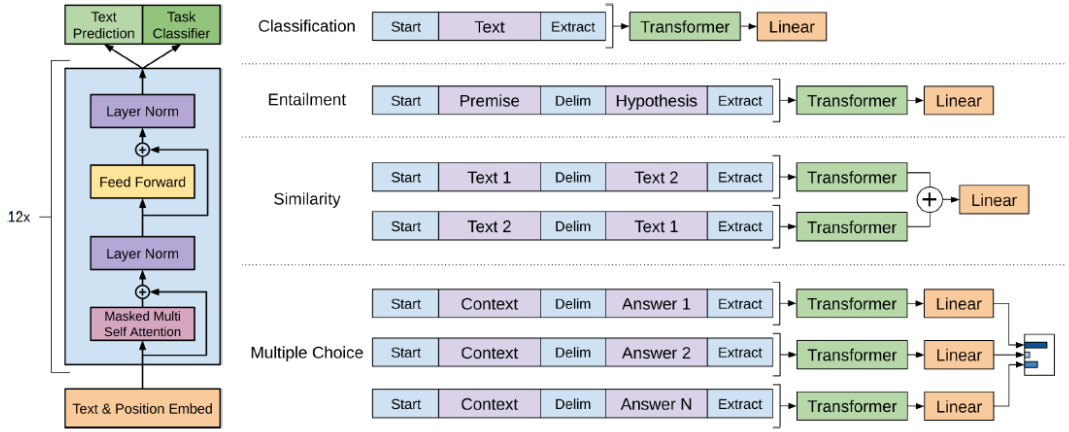
Example with 2 Encoder/Decoder Layers



Transformer >>> Language Model

- Masked LM/Autoencoding LM = predict missing words
  - "encoder" part of transformer
  - good at sentence/token classification
  - e.g. BERT, ALBERT, RoBERTA

- Autoregressive LM = predict next word
  - "decoder" part of transformer
  - good at generating text
  - e.g. GPT, LLaMa, Reformer



Radford et al., OPENAI, 2018

https://huggingface.co/transformers/v3.1.0/model_summary.html