Transformer

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

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 NLP models are transformer-based
    - BERT >> ALBERT, RoBERTA, WangchanBERTa >> GPT
  - Also applicable to vision tasks (ViT)

Main Contributions of Transformer Paper

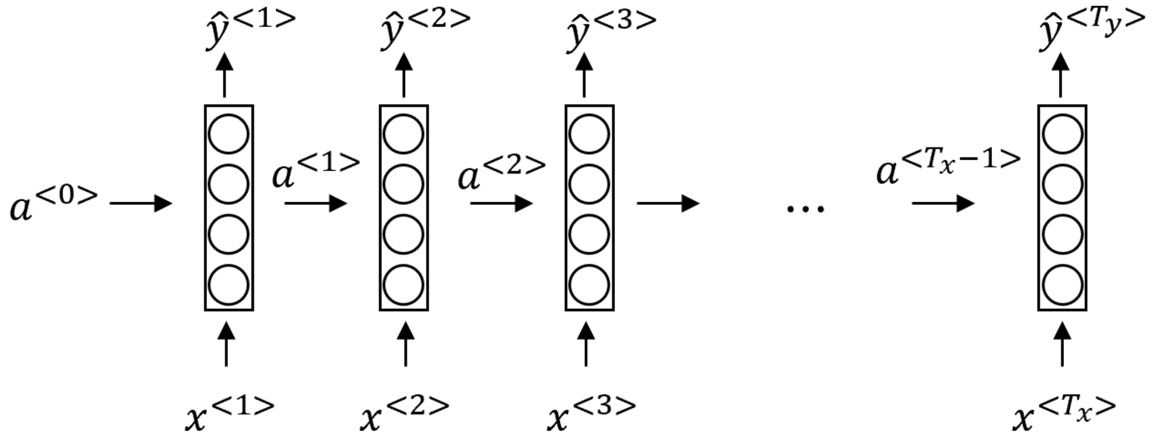
- More parallelizable than RNN
- Much fewer # of operations than CNN-based solutions
- SOTA machine translation
  - Fast training time
  - SOTA BLEU scores on Eng-Ger, Eng-Fr

Background Concepts

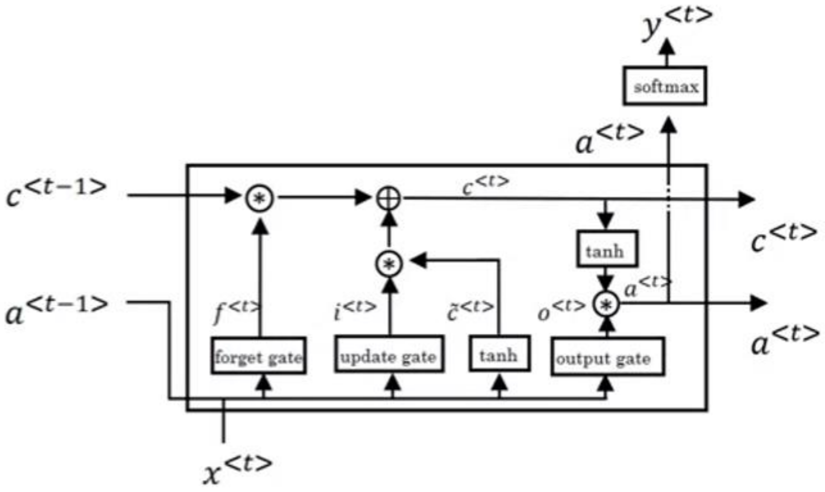
- RNN, LSTM
- Encoder-Decoder Model
  - Sequence to Sequence Learning with Neural Networks (Cho et al., 2014)
    - https://arxiv.org/abs/1409.3215
  - Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (Sutskever et al., 2014)
    - https://arxiv.org/abs/1406.1078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

Quick Recap

RNN



LSTM

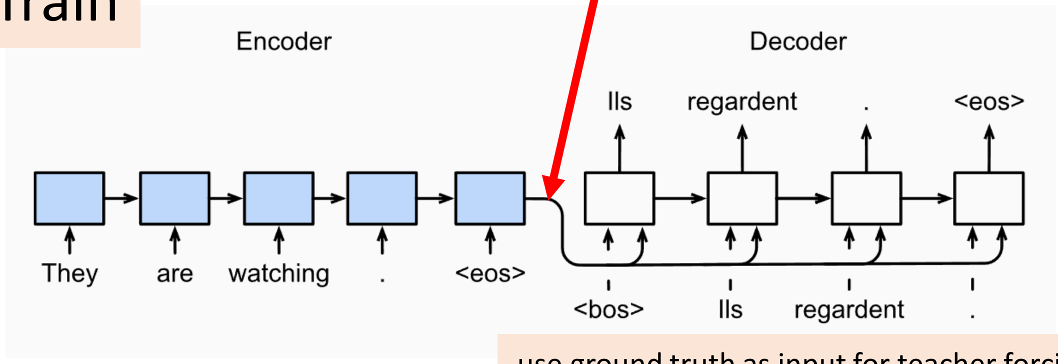


Encoder-Decoder Architecture

- “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)

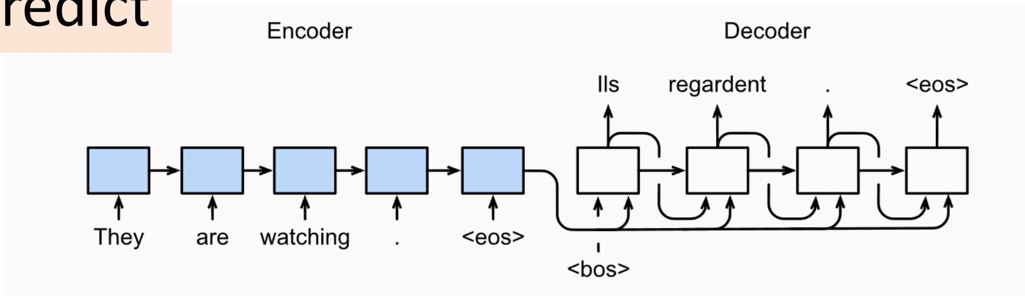
context vector
(computed from hidden state of RNN)

Train



use ground truth as input for teacher forcing

Predict

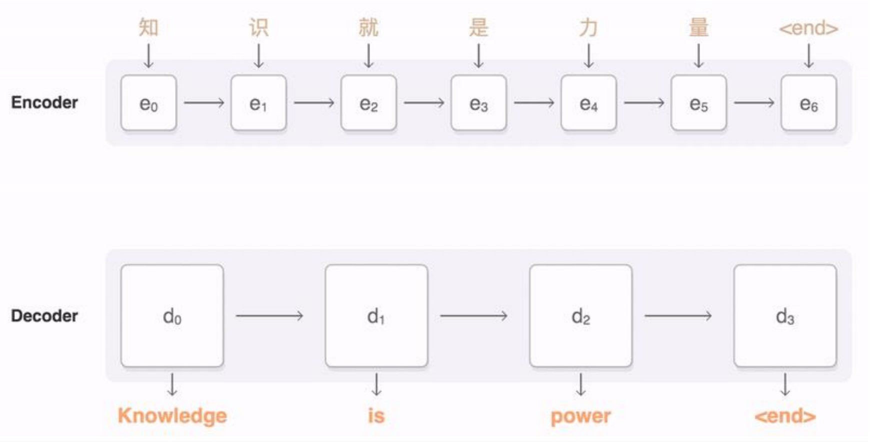


https://d2l.ai/chapter_recurrent-modern/seq2seq.html

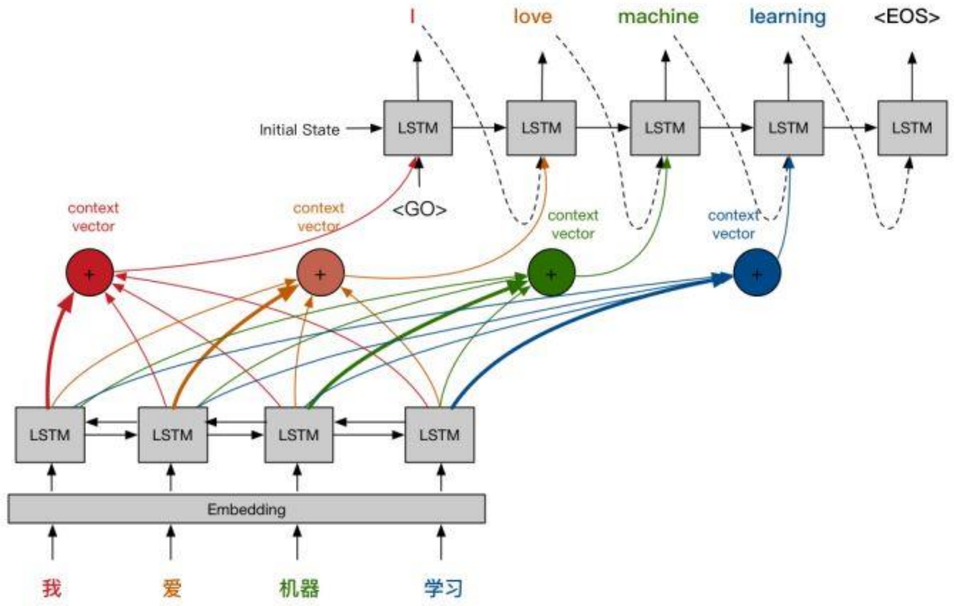
Attention

- 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



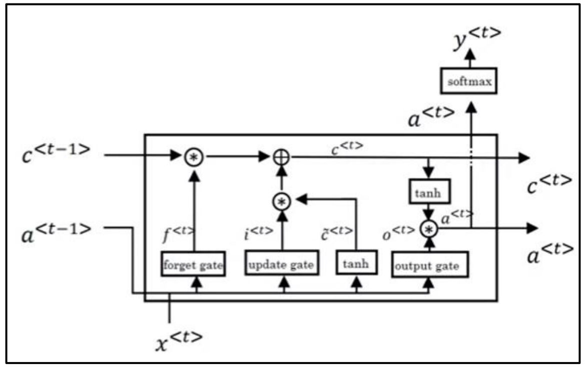
https://zhuanlan.zhihu.com/p/37290775

How to compute attention?

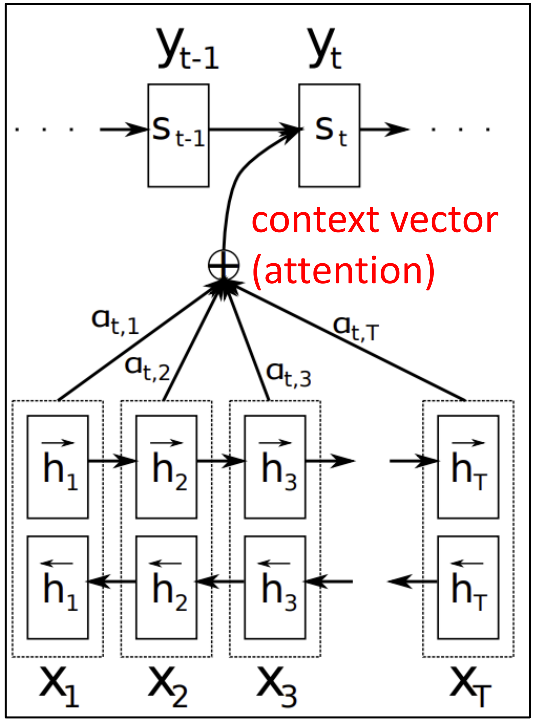
(Bahdanau, 2015)

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

- Encoder-decoder notations:
  - ในส่วน encoder ใช้ Bidirectional RNN
    - ซึ่งในแต่ละ step t' มี forward $\overrightarrow{h}^{<t'>}$ 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

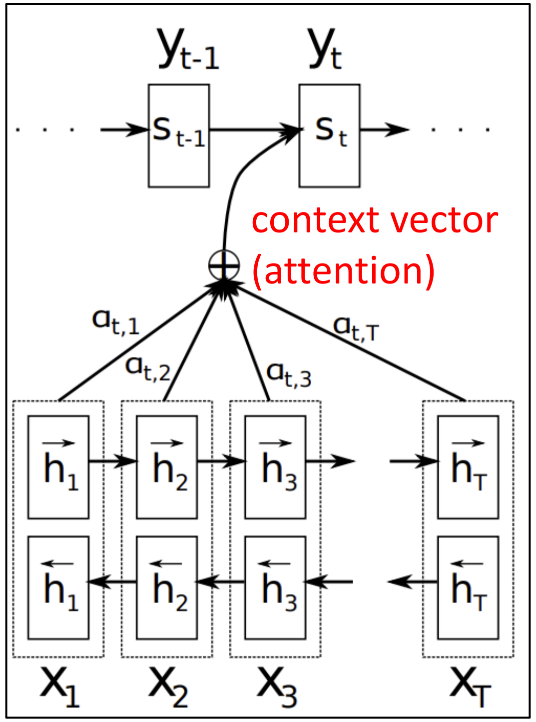


LSTM cell



https://arxiv.org/abs/1409.0473

How to compute attention?
(Bahdanau, 2015)



Attention Mechanism

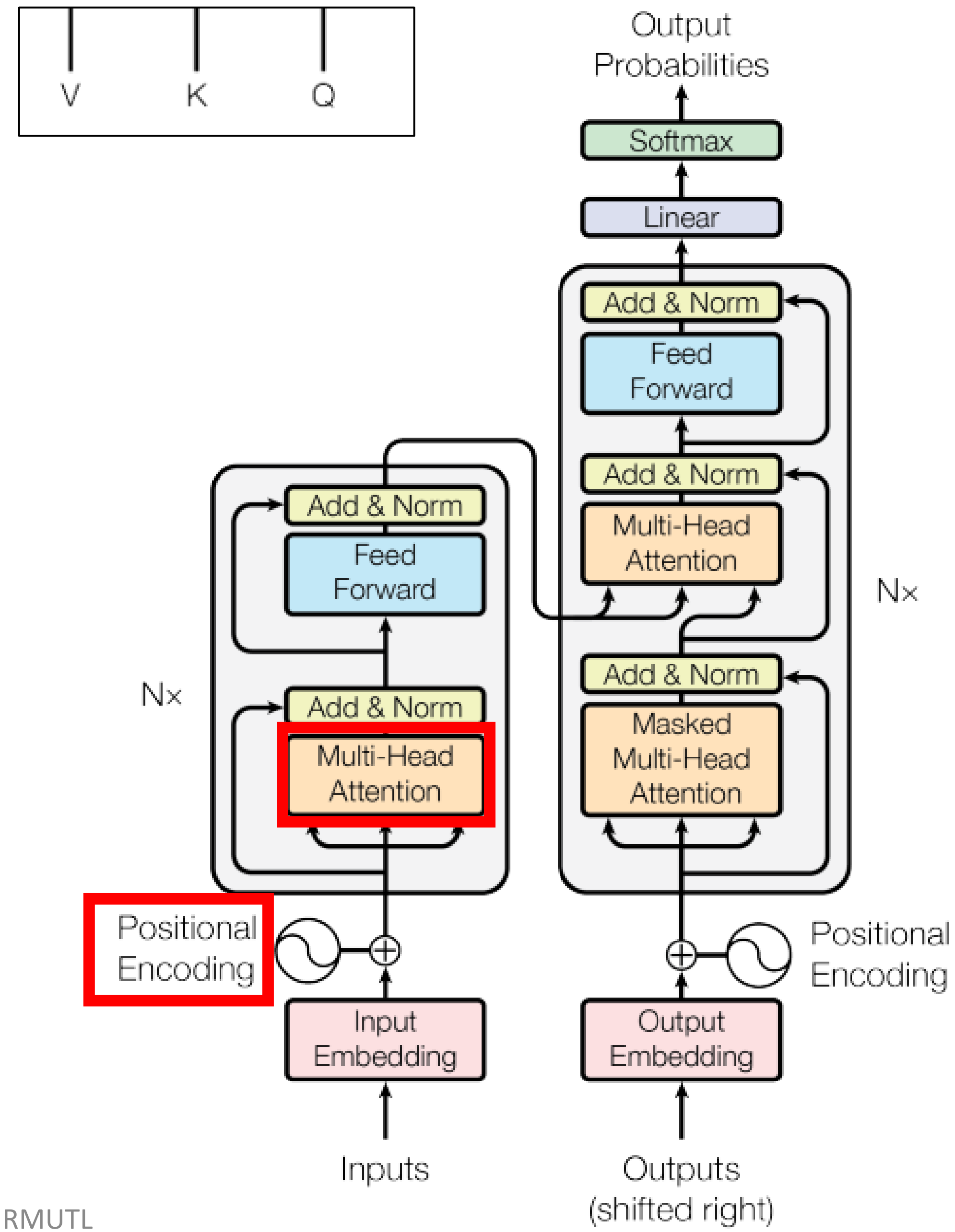
- การคำนวณ context vector $c^{<t>}$
  - $c^{<t>} = \sum_{t'=1}^{T_x} \alpha^{<t,t'>} [\overrightarrow{h}^{<t'>}, \overleftarrow{h}^{<t'>}]$
  - เป็นการสกัดข้อมูลที่เกี่ยวข้องมาจาก $\overrightarrow{h}^{<t'>}, \overleftarrow{h}^{<t'>}$ ของทั้ง encoder sequence

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

https://arxiv.org/abs/1409.0473

Transformer Model

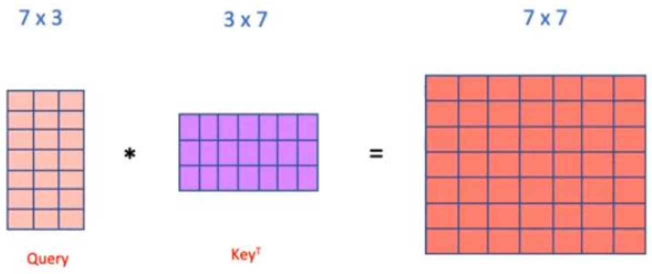
- No recurrence/convolution
- Self-Attention
- Multi-Head Attention
- Positional Encoding
  - To maintain word ordering information
- Residual Connections



Output from ALL previous steps of decoder?

https://stats.stackexchange.com/questions/421935/what-exactly-are-keys-queries-and-values-in-attention-mechanisms?rq=1

Multi-Head Attention

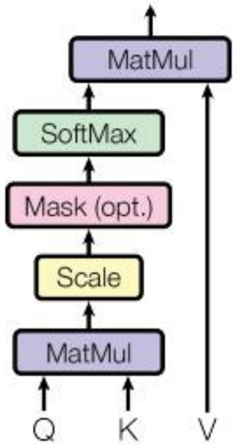


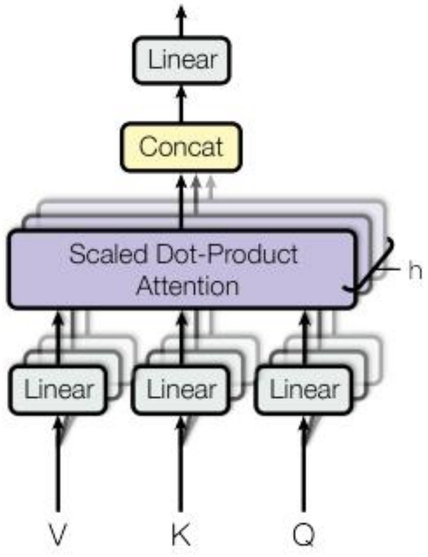
Attention Filter

Concept from Retrieval System

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

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





- QK^T (attention filter):
  - look up keys that are closest to query
- times V:
  - get V that corresponds to that key

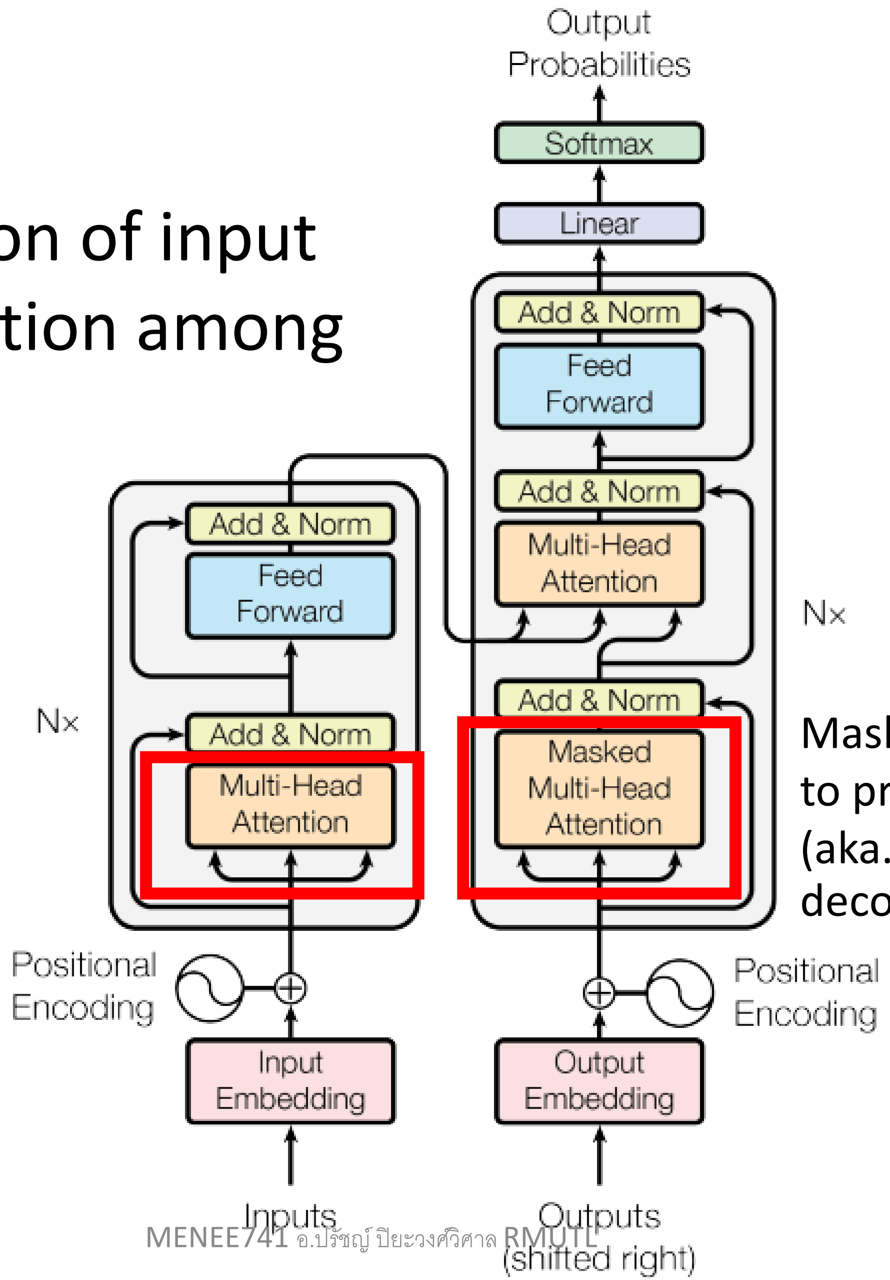
Multi-Head =>
Each head detects different feature of the language

Self-Attention

Self-Attention: representation of input sequence that captures relation among input words

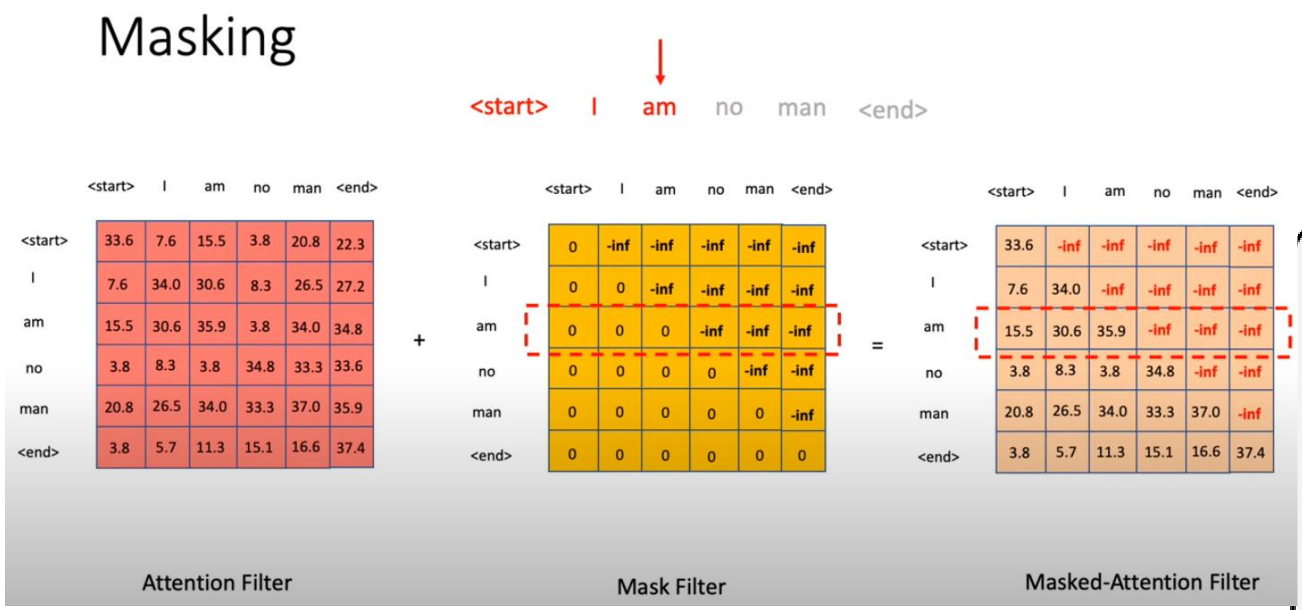
	When	you	play	the	game	of	thrones
When	89	20	41	10	55	10	59
you	20	90	81	22	70	15	72
play	41	81	95	10	90	30	92
the	10	22	10	92	88	40	89
game	55	70	90	88	98	44	87
of	10	15	30	40	44	85	59
thrones	59	72	92	90	95	59	99

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



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

Masking



	<start>	I	am	no	man	<end>
<start>	33.6	7.6	15.5	3.8	20.8	22.3
I	7.6	34.0	30.6	8.3	26.5	27.2
am	15.5	30.6	35.9	3.8	34.0	34.8
no	3.8	8.3	3.8	34.8	33.3	33.6
man	20.8	26.5	34.0	33.3	37.0	35.9
<end>	3.8	5.7	11.3	15.1	16.6	37.4

Attention Filter

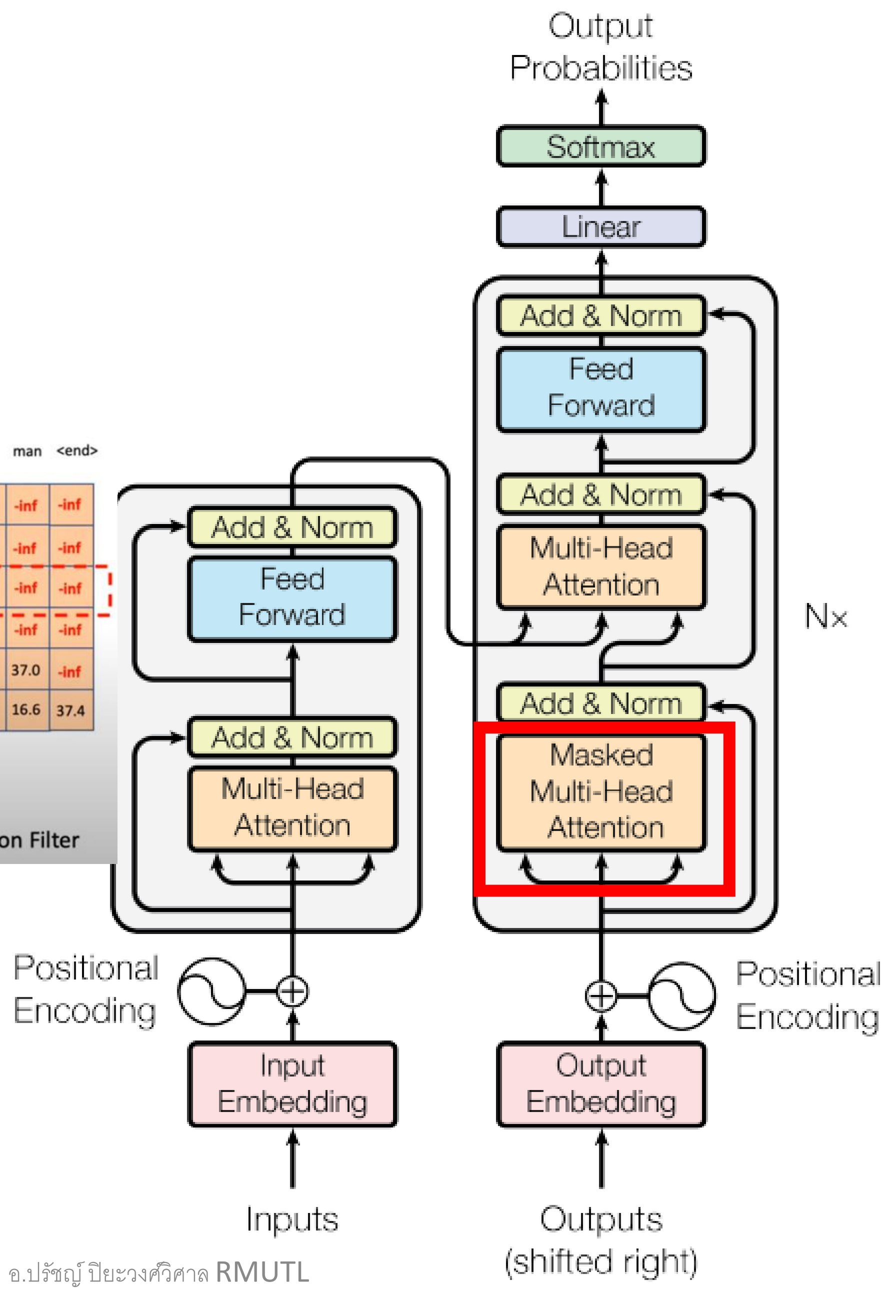
	<start>	I	am	no	man	<end>
<start>	0	-inf	-inf	-inf	-inf	-inf
I	0	0	-inf	-inf	-inf	-inf
am	0	0	0	-inf	-inf	-inf
no	0	0	0	0	-inf	-inf
man	0	0	0	0	0	-inf
<end>	0	0	0	0	0	0

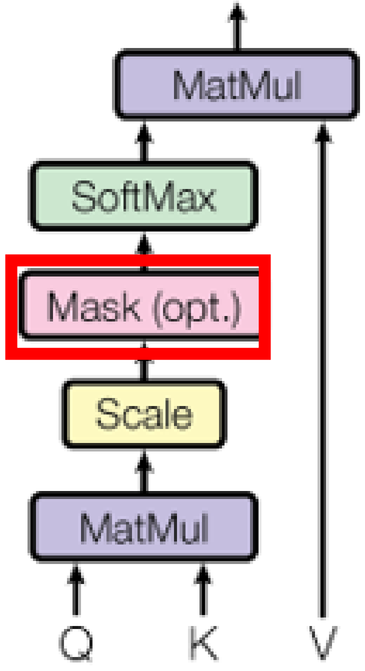
Mask Filter

	<start>	I	am	no	man	<end>
<start>	33.6	-inf	-inf	-inf	-inf	-inf
I	7.6	34.0	-inf	-inf	-inf	-inf
am	15.5	30.6	35.9	-inf	-inf	-inf
no	3.8	8.3	3.8	34.8	-inf	-inf
man	20.8	26.5	34.0	33.3	37.0	-inf
<end>	3.8	5.7	11.3	15.1	16.6	37.4

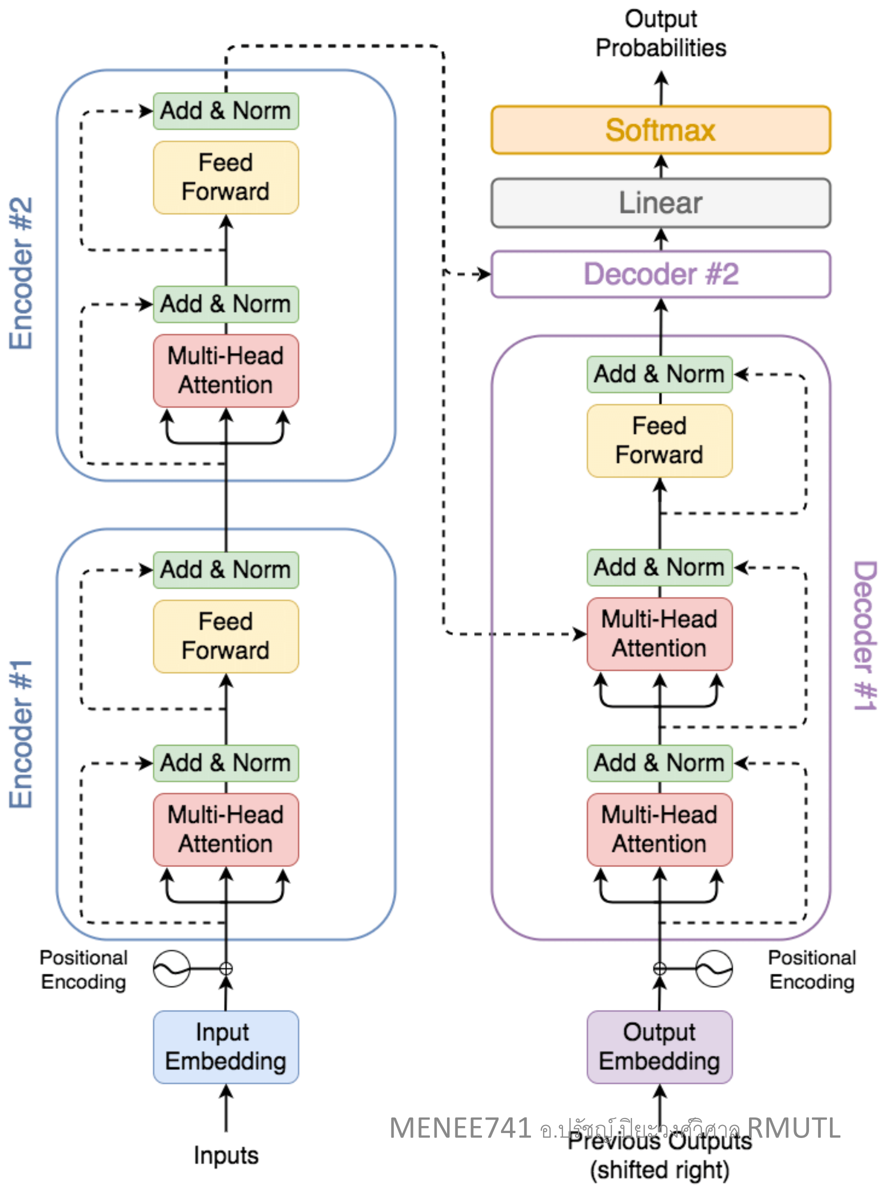
Masked-Attention Filter

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





Example with 2 Encoder/Decoder Layers



Confusing Part

Why do we need K & V

Not just K ?

To read:
https://stats.stackexchange.com/questions/421935/what-exactly-are-keys-queries-and-values-in-attention-mechanisms?rq=1

https://medium.com/@b.terryjack/deep-learning-the-transformer-9ae5e9c5a190