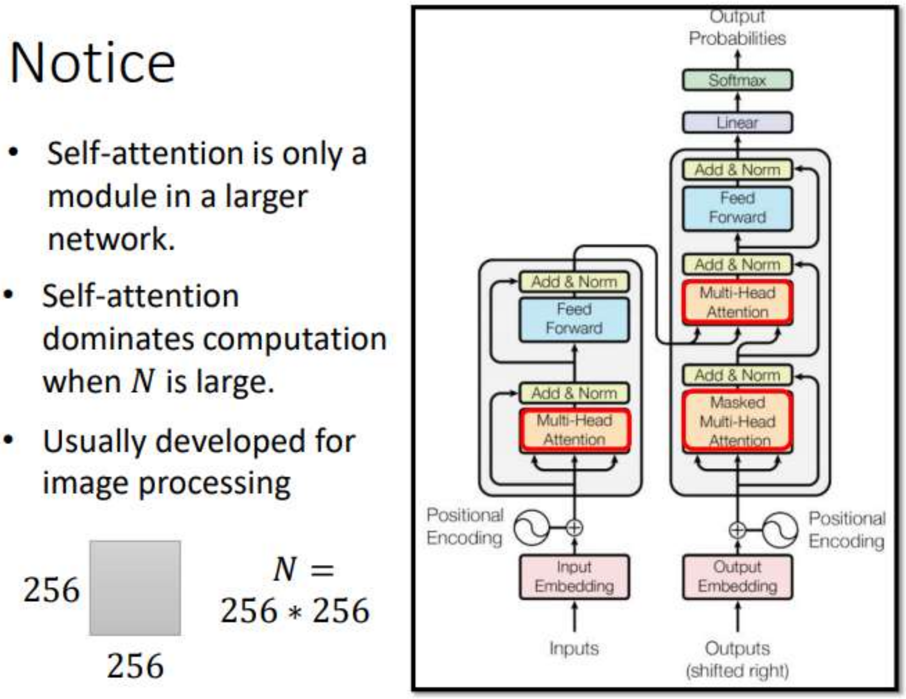
Lecture 5 : Sequence to sequence

Create at 2022/06/01

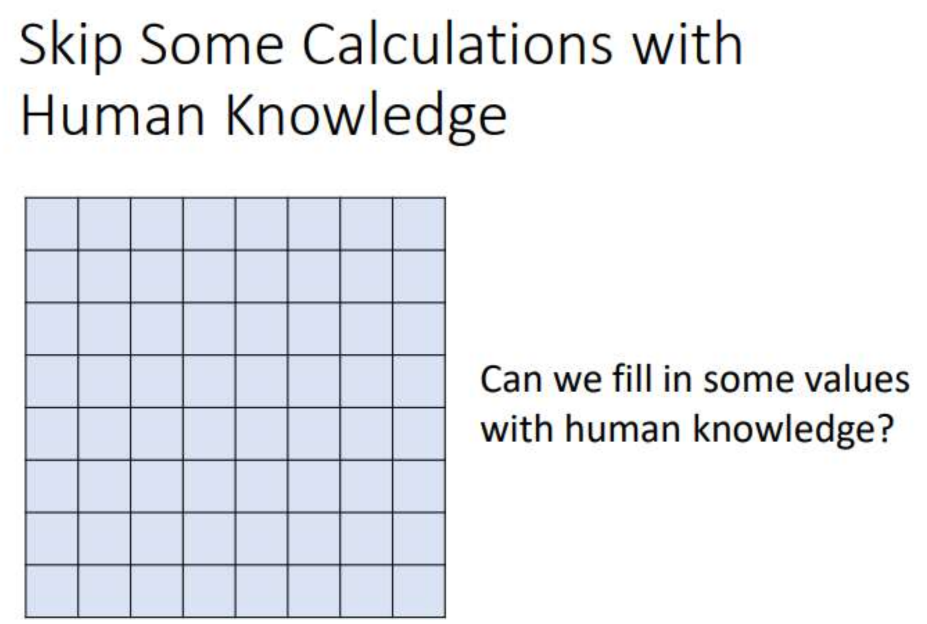
- Lecture 5 : Sequence to sequence
    - 各式各樣的 self-attention
        - Local attention / Truncated attention
        - Stride attention
        - Global attention
        - Clustering
        - Sinkhorn Sorting Network
        - Linformer

- 上課資源：
    1. 各式各樣神奇的自注意力機制 (Self-attention) 變型 (https://www.youtube.com/watch?v=yHoAq1IT_og)
- 預習教材：
    - 【機器學習2021】自注意力機制 (Self-attention) (上) (https://www.youtube.com/watch?v=hYdO9CscNes)
    - 【機器學習2021】自注意力機制 (Self-attention) (下) (https://www.youtube.com/watch?v=gmsMY5kc-zw)

各式各樣的 self-attention

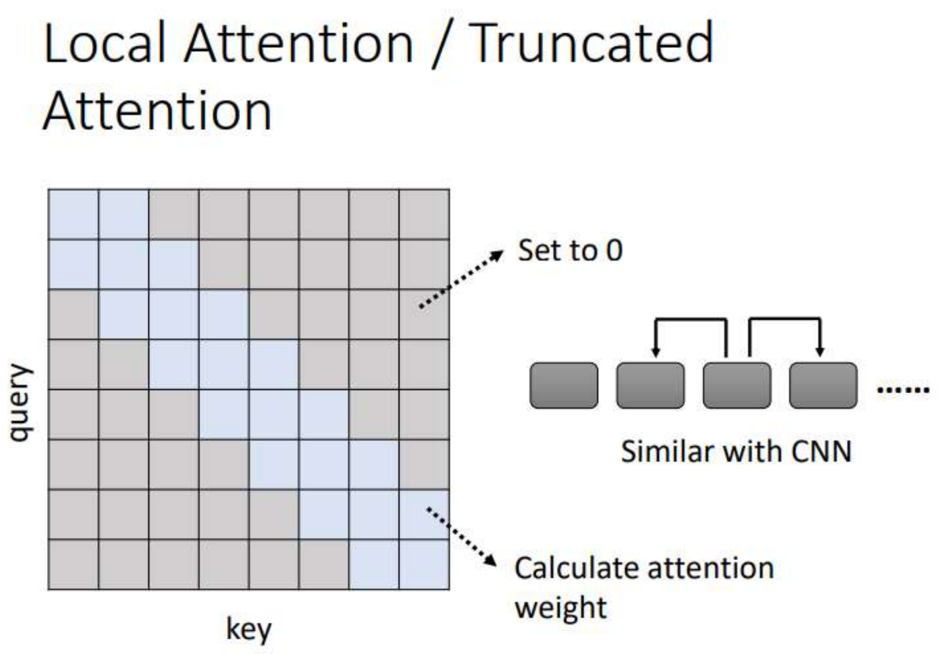


- self-attention 只是一個很大的 network 裡面其中一小部分
- 當 sequence 很長的時候，加快 self-attention 才有可能發揮真正的效果



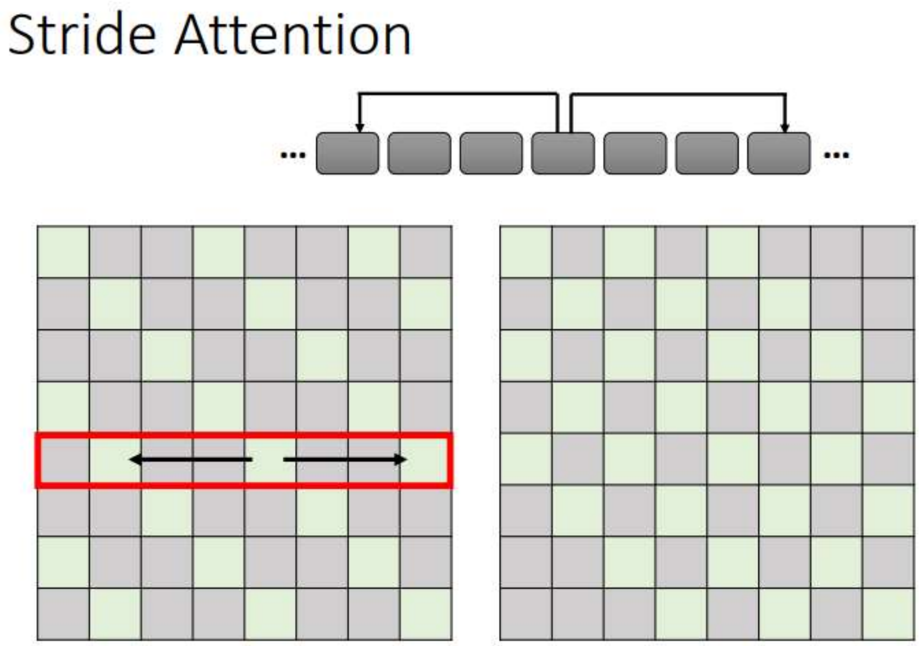
透過人類的知識直接填入正確資訊，減少運算

Local attention / Truncated attention



- 有些問題不需要看整個 sequence，只需要看左右鄰居就可以得到一個位置有什麼樣的資訊
- 可以加快運算速度，只計算藍色的部分
- Local attention 是可以加快 self-attention 的方法

Stride attention



- 看三格之前的資訊與三格之後的資訊，可以看到比較大範圍的資訊
- 只計算綠色的部分

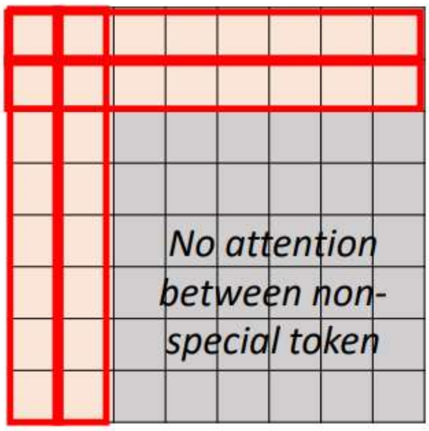
Global attention

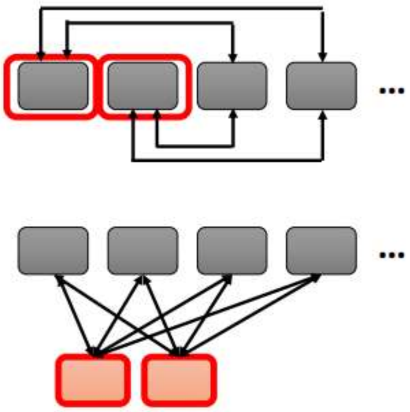
Global Attention

special token = “token中的里長伯“

Add special token into original sequence

- Attend to every token → collect global information
- Attended by every token → it knows global information



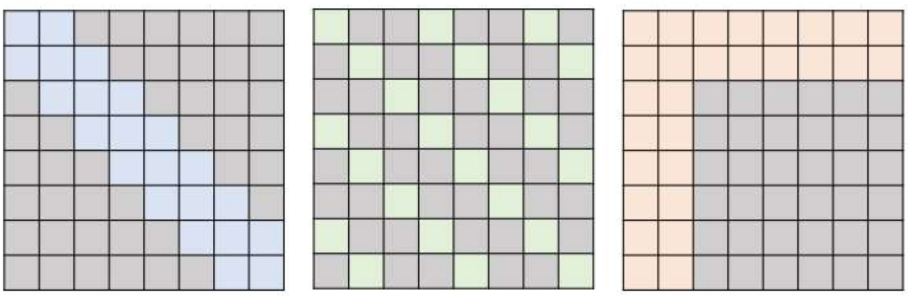


- 如果要看一整個 sequence 發生甚麼事，可以用 global attention
- 在原本的 sequence 裡面加上特殊的 token，代表這個位置需要做 global attention

Global attention 會做甚麼事呢？

- attend 到 sequence 裡的每一個 token，到 sequence 裡的每一個 token 收集資訊
- 法 1 : 在原來的 sequence 裡面直接 assign 一些 token 做為 special token
- 法 2 : 用外加 token，special token 會 attend 到所有其他的 token

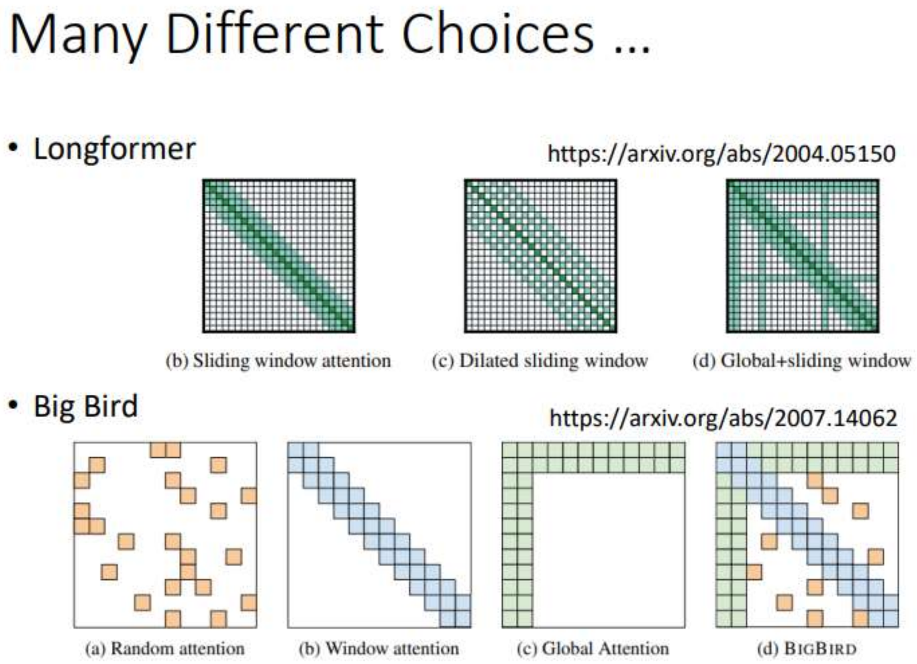
Many Different Choices ...



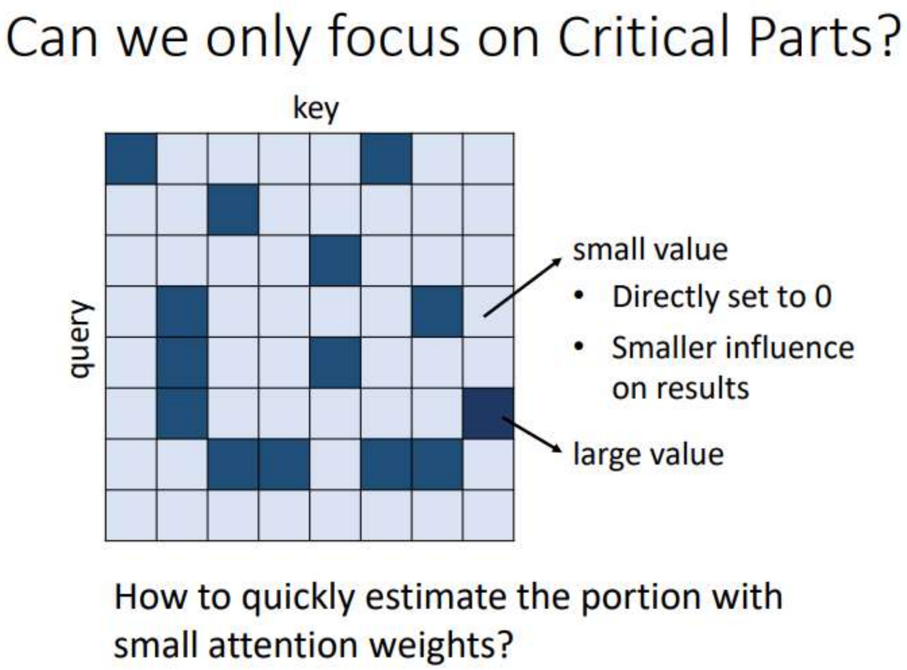
小孩子才做選擇．．．

Different heads use different patterns.

真正好的選擇是...全部都用

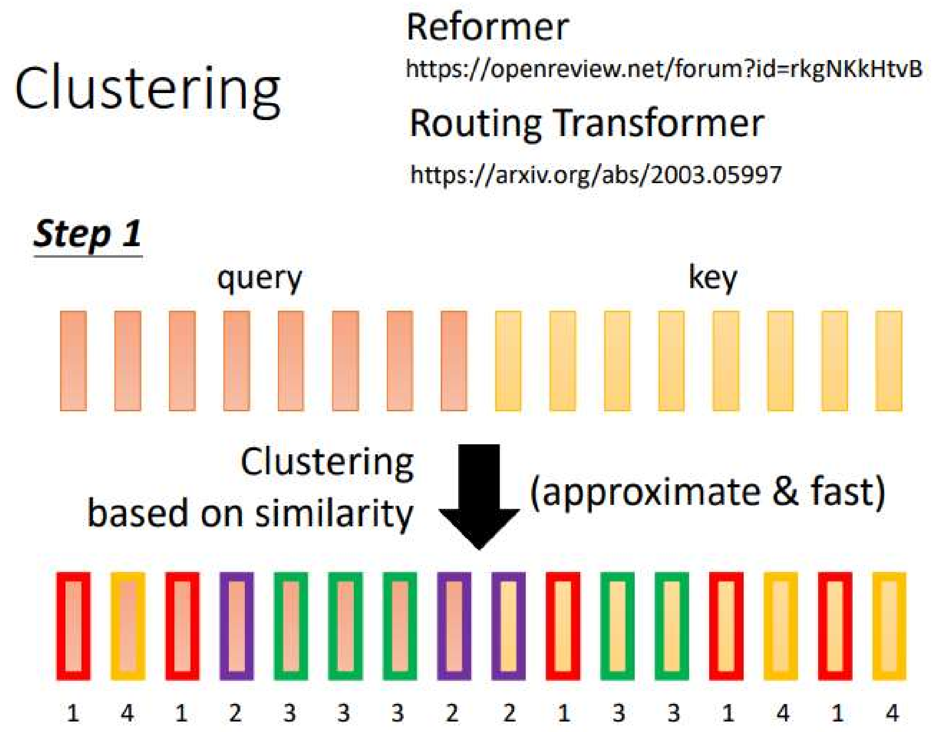


self-attention 的變形

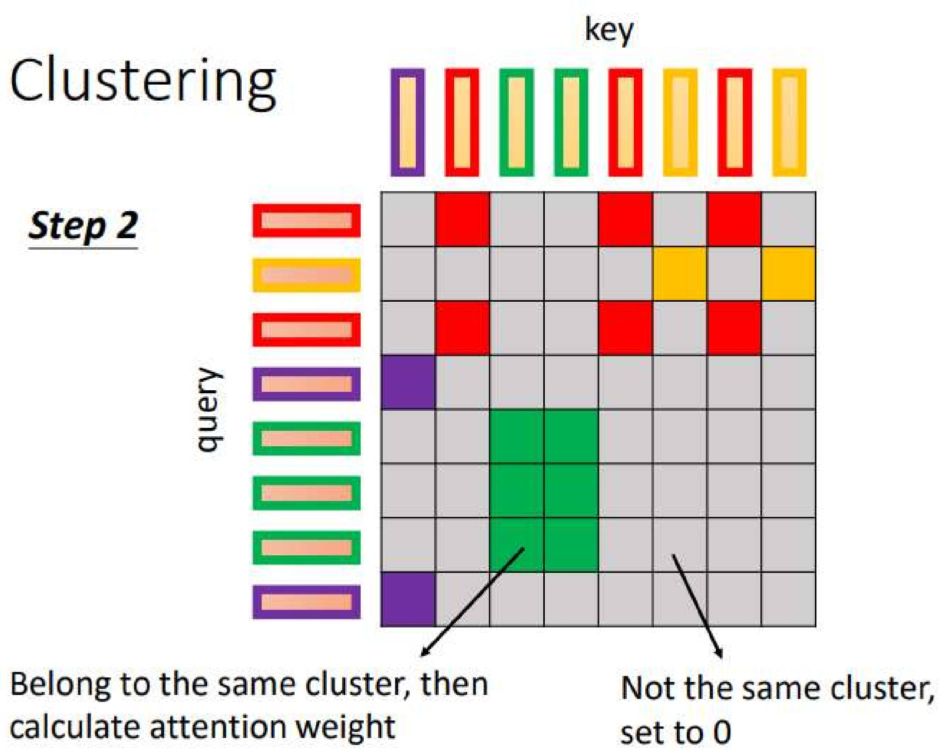


去計算有可能有比較大的 attention value 的位置

Clustering

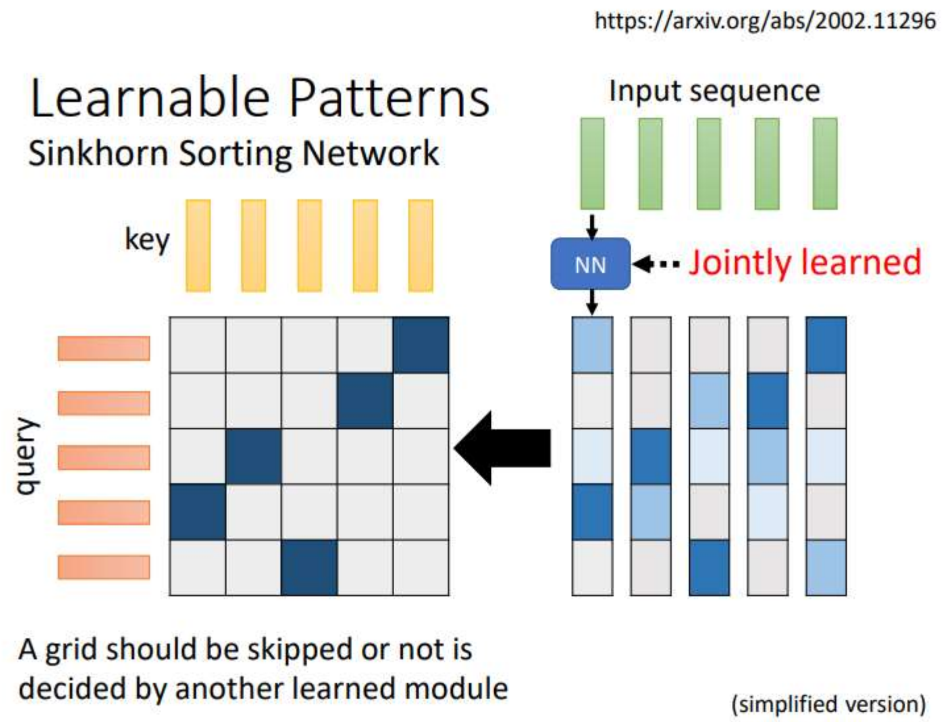


- 先把 query 跟 key 拿出來，根據 query 跟 key 相近的程度做 clustering
- 相近的就分類在一起，比較遠的就屬於不同的 clustering (相同顏色框的是相同的 clustering)



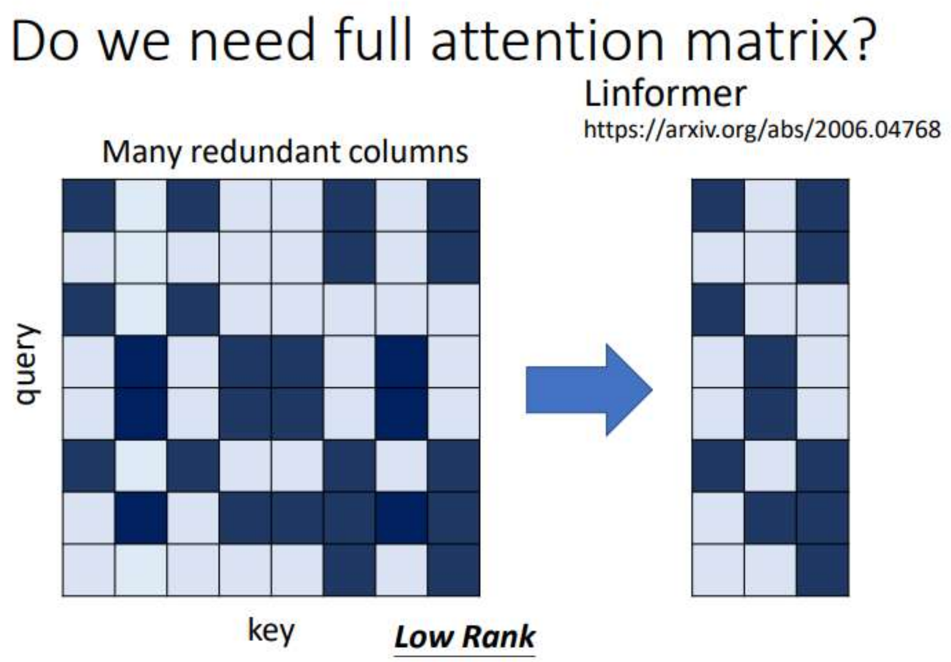
- 當 query 跟 key 在同一個 clustering 裡面的話，去計算它的 attention weight
- 否則直接設為 0

Sinkhorn Sorting Network

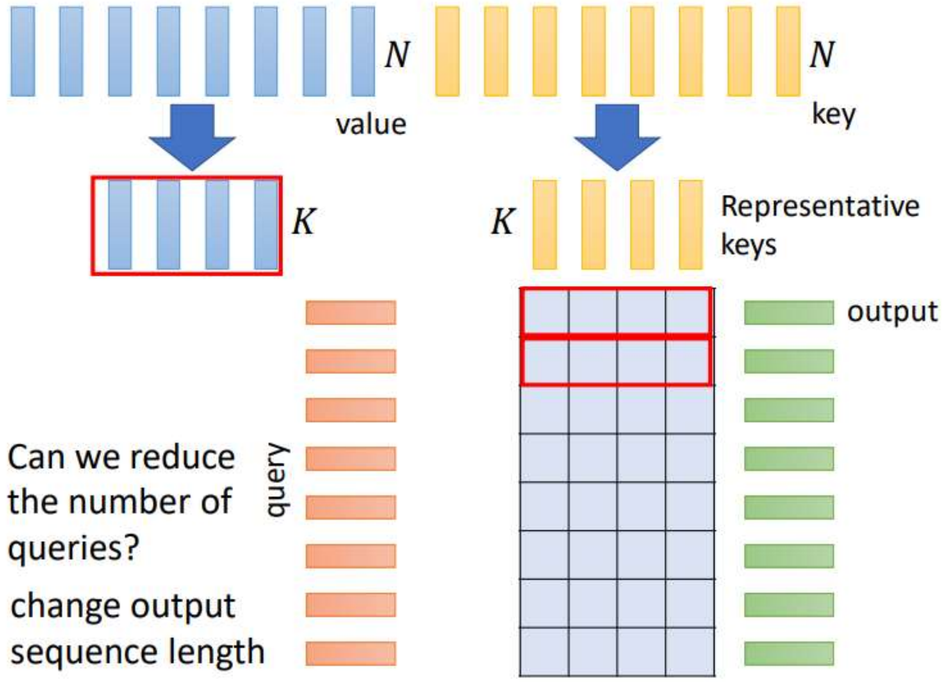


- 在 Sinkhorn Sorting Network 用另外一個 network 來決定哪些地方需要計算 attention

Linformer



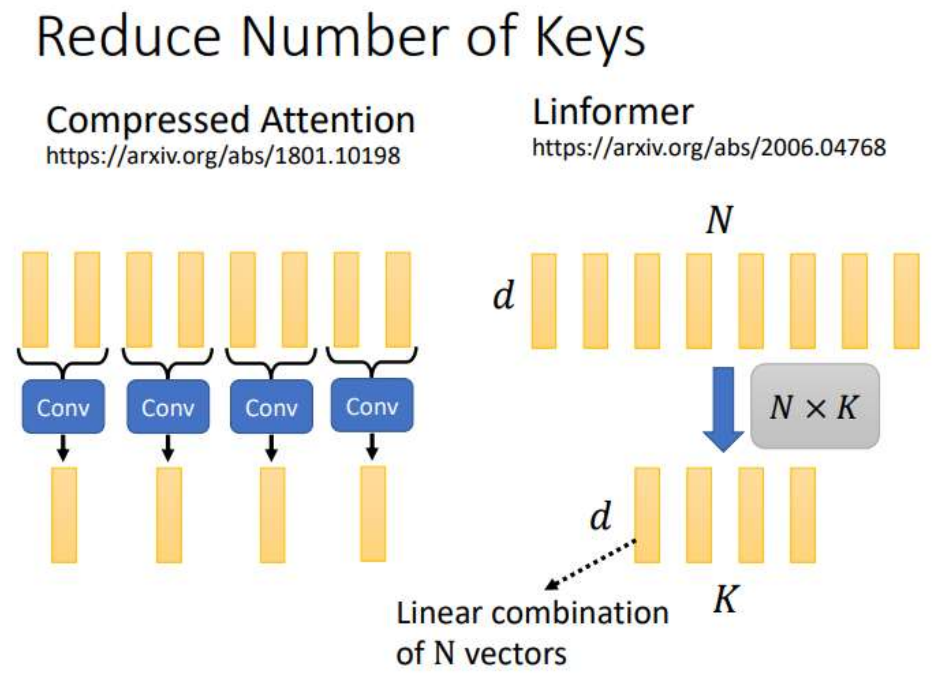
- 把重複的 column 拿掉，變成一個比較小的 attention matrix
- 可以加快運算速度



- 從 N 個 key 中，選有代表性的 K 個 key
- 從 N 個 value vector 中，選有代表性的 K 個 value vector
- 把 K 個 key 對第一個 query vector 算出來的 attention weight 對 K 個 value vector 做 weighted sum 得到 output
- 以此類推，可以得到整個 self-attention 的 output

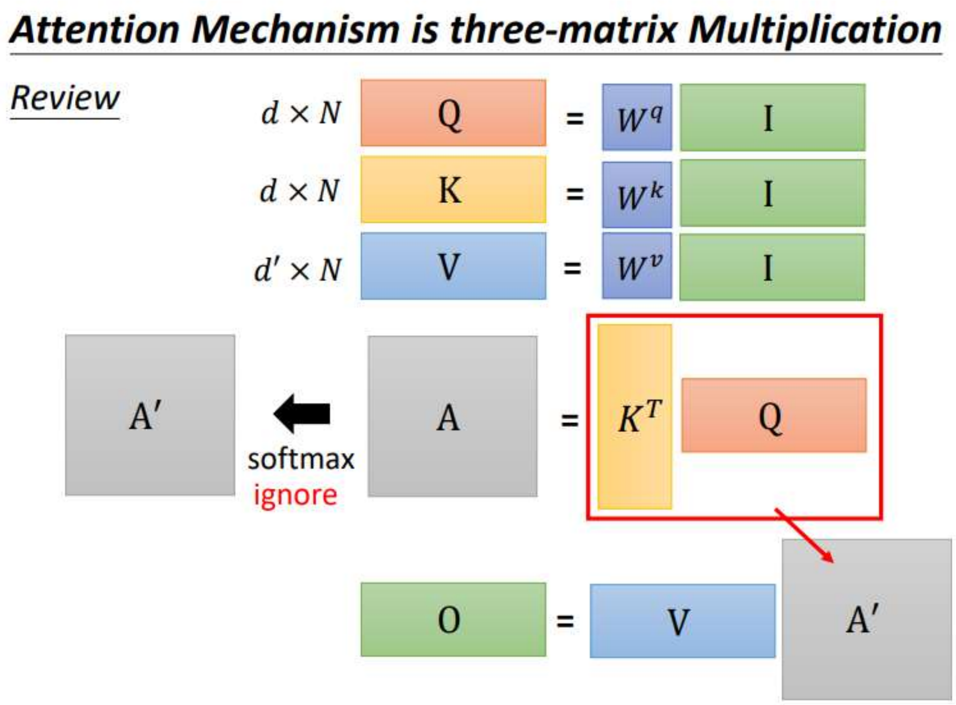
為甚麼不選有代表性的 query？

- 如果只選有代表性的 query，會遇到 output sequence length 縮小，因為 output sequence length = query length
- 會不會受影響呢？
  - case by case
  - 如果 input 一段句子，只需要輸入一個 label，這種例子就不會受影響，可以選有代表性的 query 就好
  - 如果是 input sequence 的每個位置都需要 output 一個 label，就會有問題

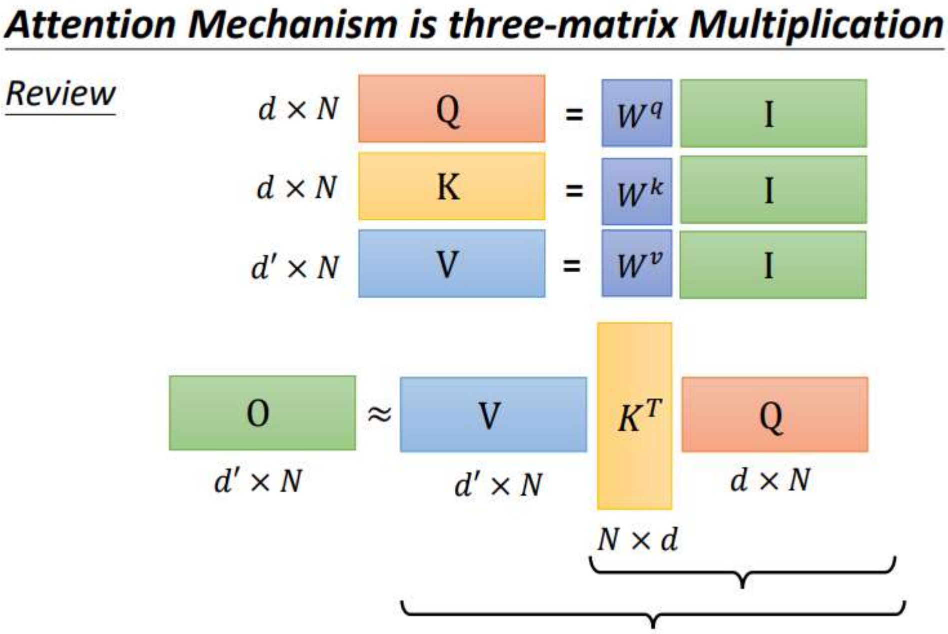


怎麼選出有代表性的 key？

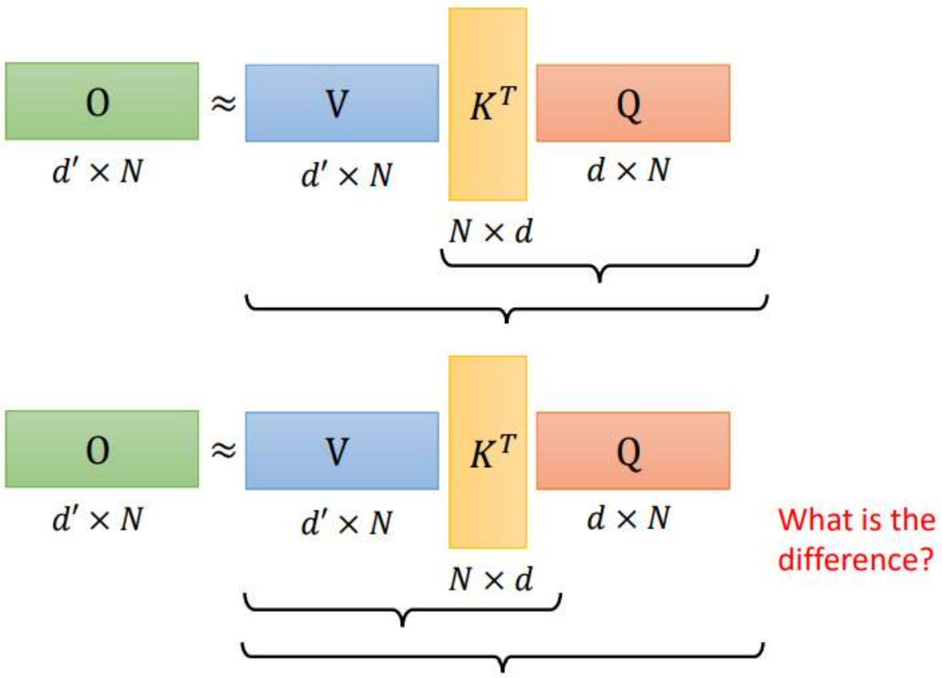
- 法1 : Compressed Attention
    - input 很長 key 的 sequence，用 CNN 掃過 sequence 將 sequence 長度變短，這些 sequence 就當成有代表性的 key
- 法2 : Linformer
    - input 的 key 集合起來，可以看做是 $d*N$ 的矩陣
    - 把 $d*N$ 的矩陣乘上 $N*K$ 的矩陣得到 $d*K$ 的矩陣 (Linear combination)
    - $d*K$ 的矩陣，K 個 column 就是有代表性的 key



Attention Mechanism is three-matrix Multiplication

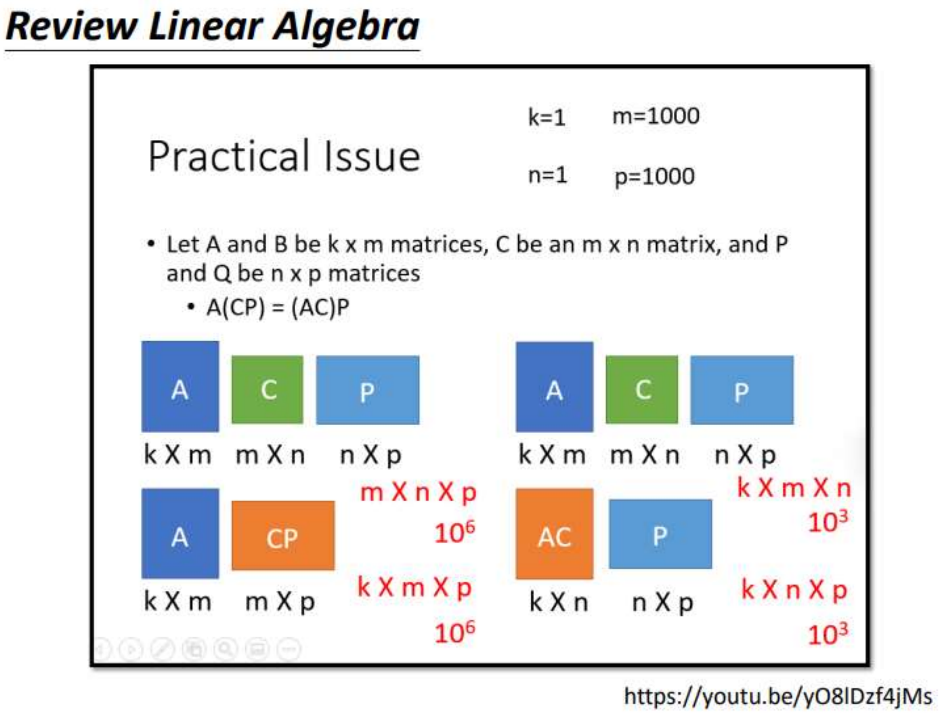


式子簡化

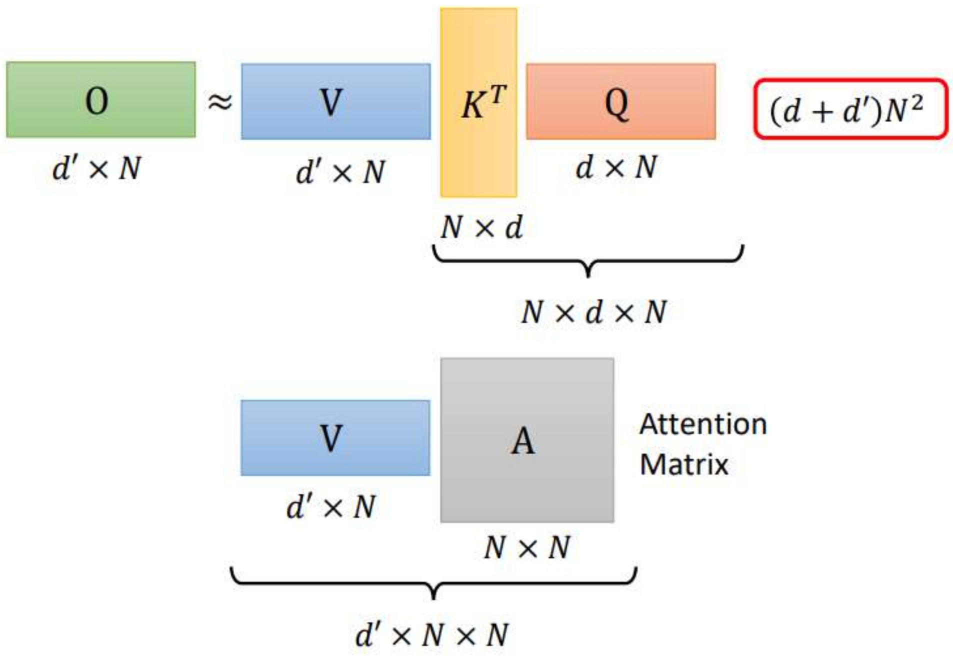


其實可以再加速，上下兩個式子有甚麼不同？

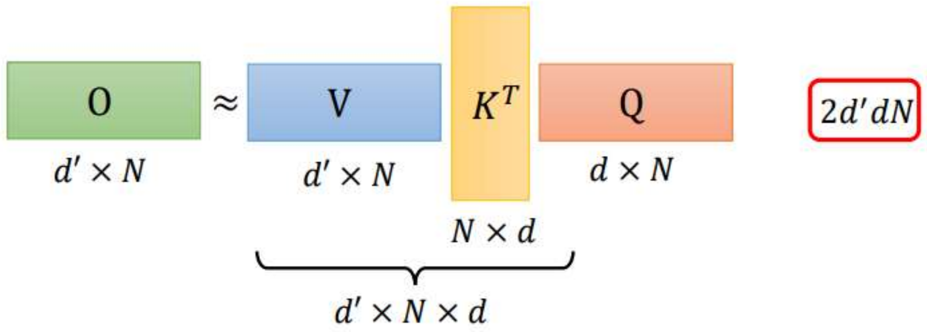
- 得到的結果相同，但中間需要的運算量不同

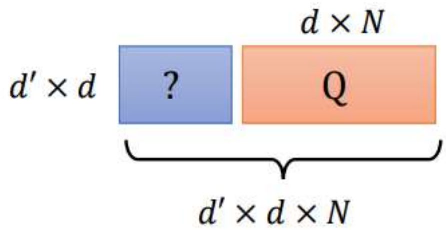


線性代數觀念複習

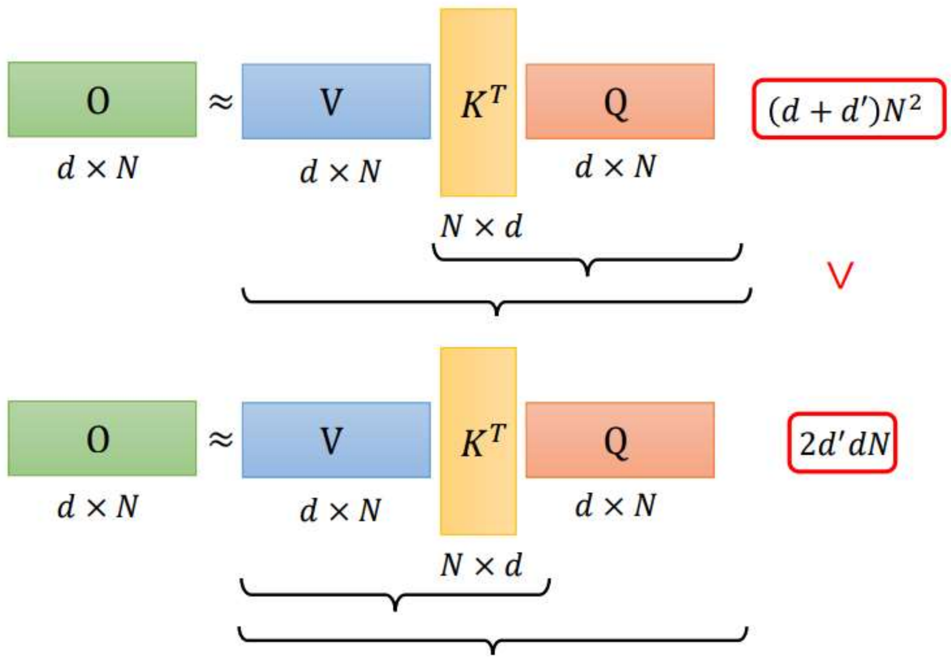


先做 $K^T * Q$ 得到 A，再乘上 V
得到的運算量總共是 $(d+d^{'})N^2$





先做 $V * K^T$，再乘上 Q
得到的運算量總共是 $2d^{'}dN$



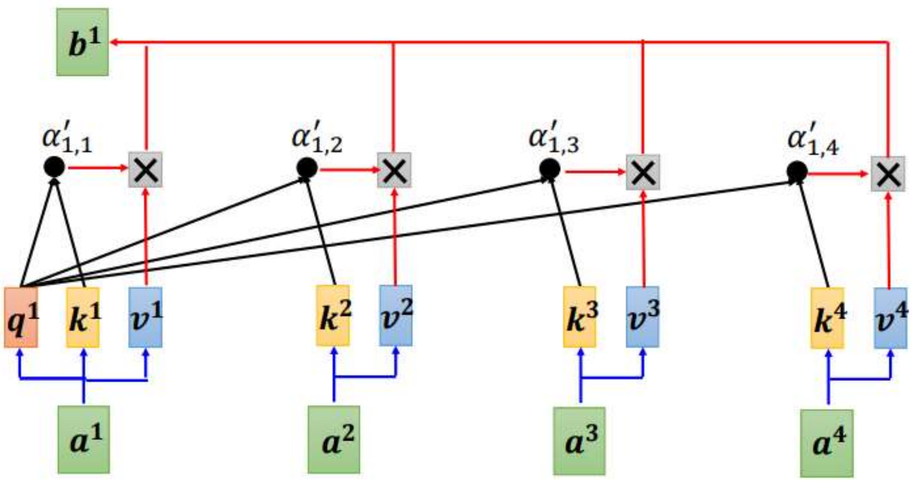
改變乘法的順序，運算的次數會有很大的改變

老師說不想聽數學很多的話，可以跳過ㄟ 那我要跳過ㄌ~

Let's put softmax back ...

Warning of math

$$b^1 = \sum_{i=1}^{N} \boxed{\alpha'_{1,i}} v^i = \sum_{i=1}^{N} \boxed{\frac{exp(q^1 \cdot k^i)}{\sum_{j=1}^{N} exp(q^1 \cdot k^j)}} v^i$$



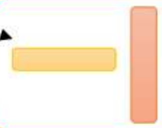
$$\boldsymbol{b}^1 = \sum_{i=1}^{N} \alpha'_{1,i} \boldsymbol{v}^i = \sum_{i=1}^{N} \frac{exp(\boldsymbol{q}^1 \cdot \boldsymbol{k}^i)}{\sum_{j=1}^{N} exp(\boldsymbol{q}^1 \cdot \boldsymbol{k}^j)} \boldsymbol{v}^i$$

$$\boxed{\begin{array}{c} exp(\boldsymbol{q} \cdot \boldsymbol{k}) \\ \approx \phi(\boldsymbol{q}) \cdot \phi(\boldsymbol{k}) \\ \boldsymbol{q} \rightarrow \boxed{\phi} \rightarrow \phi(\boldsymbol{q}) \end{array}}$$

$$= \sum_{i=1}^{N} \frac{\phi(\boldsymbol{q}^1) \cdot \phi(\boldsymbol{k}^i)}{\sum_{j=1}^{N} \phi(\boldsymbol{q}^1) \cdot \phi(\boldsymbol{k}^j)} \boldsymbol{v}^i$$

$$= \frac{\sum_{i=1}^{N} [\phi(\boldsymbol{q}^1) \cdot \phi(\boldsymbol{k}^i)] \boldsymbol{v}^i}{\boxed{\sum_{j=1}^{N} \phi(\boldsymbol{q}^1) \cdot \phi(\boldsymbol{k}^j)}}$$

$$\phi(\boldsymbol{q}^1) \cdot \boxed{\sum_{j=1}^{N} \phi(\boldsymbol{k}^j)}$$



$\phi(\boldsymbol{q}^1)$

$$\boldsymbol{b}^1 = \sum_{i=1}^{N} \alpha'_{1,i} \boldsymbol{v}^i = \frac{\boxed{\sum_{i=1}^{N} [\phi(\boldsymbol{q}^1) \cdot \phi(\boldsymbol{k}^i)] \boldsymbol{v}^i}}{\phi(\boldsymbol{q}^1) \cdot \sum_{j=1}^{N} \phi(\boldsymbol{k}^j)}$$

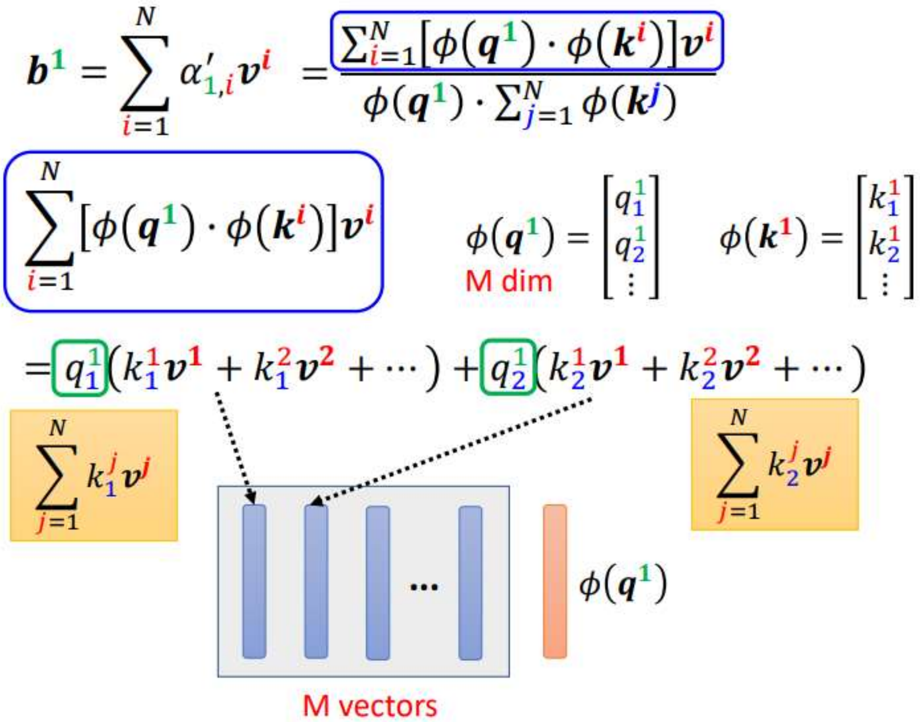
$$\boxed{\sum_{i=1}^{N} [\phi(\boldsymbol{q}^1) \cdot \phi(\boldsymbol{k}^i)] \boldsymbol{v}^i} \qquad \phi(\boldsymbol{q}^1) = \begin{bmatrix} q_1^1 \\ q_2^1 \\ \vdots \end{bmatrix} \qquad \phi(\boldsymbol{k}^1) = \begin{bmatrix} k_1^1 \\ k_2^1 \\ \vdots \end{bmatrix}$$

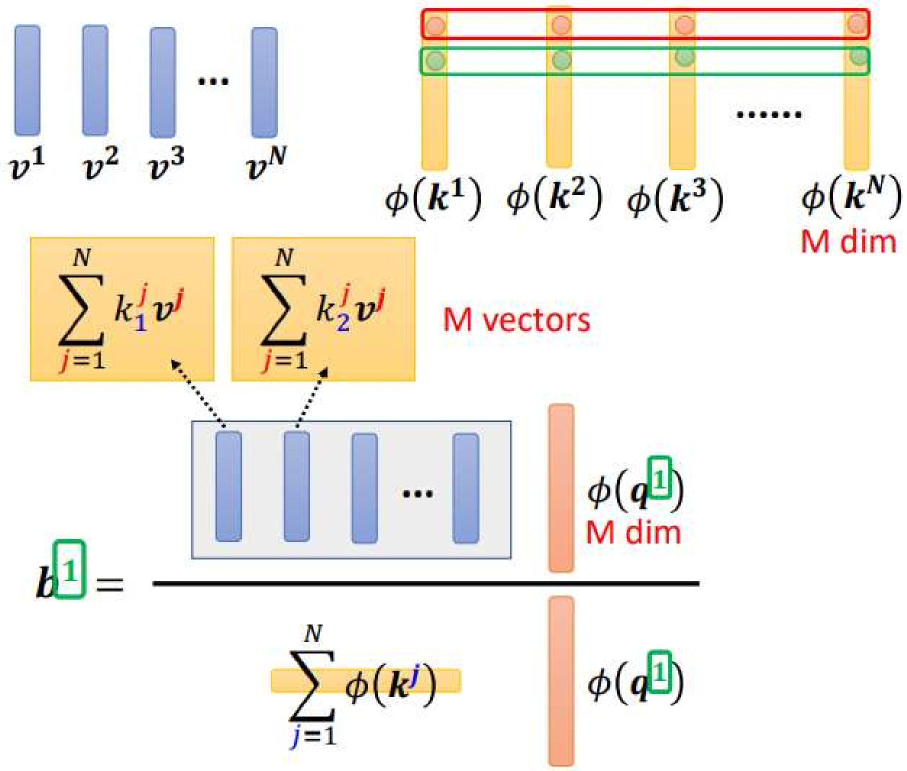
$$= [\phi(\boldsymbol{q}^1) \cdot \phi(\boldsymbol{k}^1)] \boldsymbol{v}^1 + [\phi(\boldsymbol{q}^1) \cdot \phi(\boldsymbol{k}^2)] \boldsymbol{v}^2 + \cdots$$

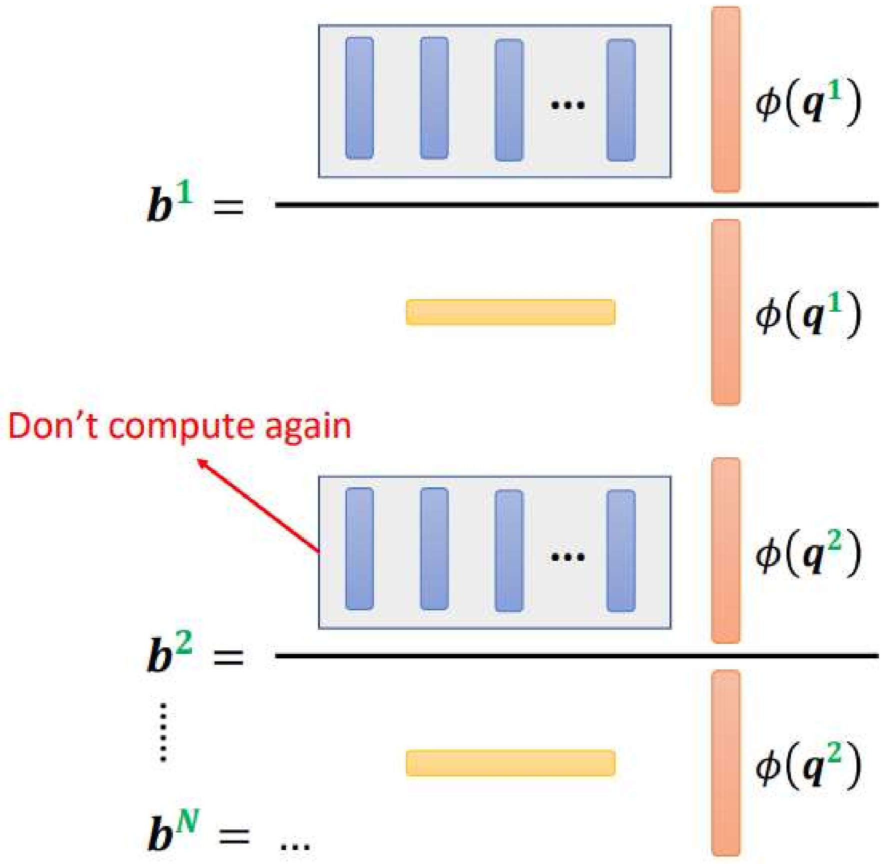
$$= (q_1^1 k_1^1 + q_2^1 k_2^1 + \cdots) \boldsymbol{v}^1 + (q_1^1 k_1^2 + q_2^1 k_2^2 + \cdots) \boldsymbol{v}^2 + \cdots$$

$$= \underline{q_1^1 k_1^1 \boldsymbol{v}^1} + \underline{q_2^1 k_2^1 \boldsymbol{v}^1} + \cdots + \underline{q_1^1 k_1^2 \boldsymbol{v}^2} + \underline{q_2^1 k_2^2 \boldsymbol{v}^2} + \cdots + \cdots$$

$$= q_1^1 (k_1^1 \boldsymbol{v}^1 + k_1^2 \boldsymbol{v}^2 + \cdots) + q_2^1 (k_2^1 \boldsymbol{v}^1 + k_2^2 \boldsymbol{v}^2 + \cdots)$$

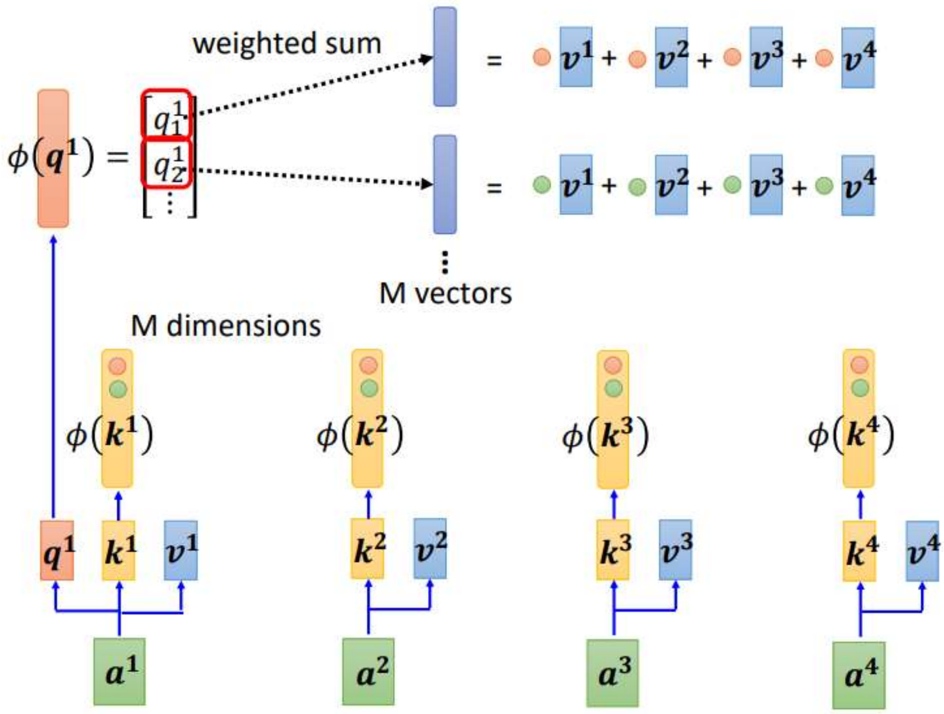

$b^1 = \sum_{i=1}^{N} \alpha'_{1,i} v^i = \frac{\sum_{i=1}^{N} [\phi(q^1) \cdot \phi(k^i)] v^i}{\phi(q^1) \cdot \sum_{j=1}^{N} \phi(k^j)}$
$\sum_{i=1}^{N} [\phi(q^1) \cdot \phi(k^i)] v^i$
$\phi(q^1) = \begin{bmatrix} q_1^1 \\ q_2^1 \\ \vdots \end{bmatrix}$
M dim
$\phi(k^1) = \begin{bmatrix} k_1^1 \\ k_2^1 \\ \vdots \end{bmatrix}$
$= q_1^1 (k_1^1 v^1 + k_1^2 v^2 + \cdots) + q_2^1 (k_2^1 v^1 + k_2^2 v^2 + \cdots)$
$\sum_{j=1}^{N} k_1^j v^j$
$\sum_{j=1}^{N} k_2^j v^j$
$\phi(q^1)$
M vectors

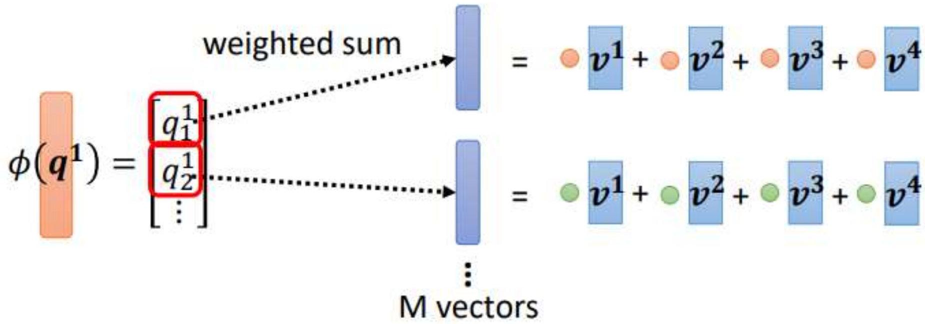

v^1 v^2 v^3 v^N
$\phi(k^1)$ $\phi(k^2)$ $\phi(k^3)$ $\phi(k^N)$
M dim
$\sum_{j=1}^{N} k_1^j v^j$
$\sum_{j=1}^{N} k_2^j v^j$
M vectors
$\phi(q^1)$
M dim
$b^1 =$
$\sum_{j=1}^{N} \phi(k^j)$
$\phi(q^1)$

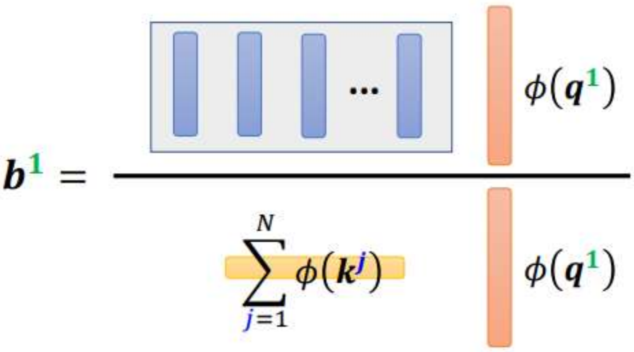


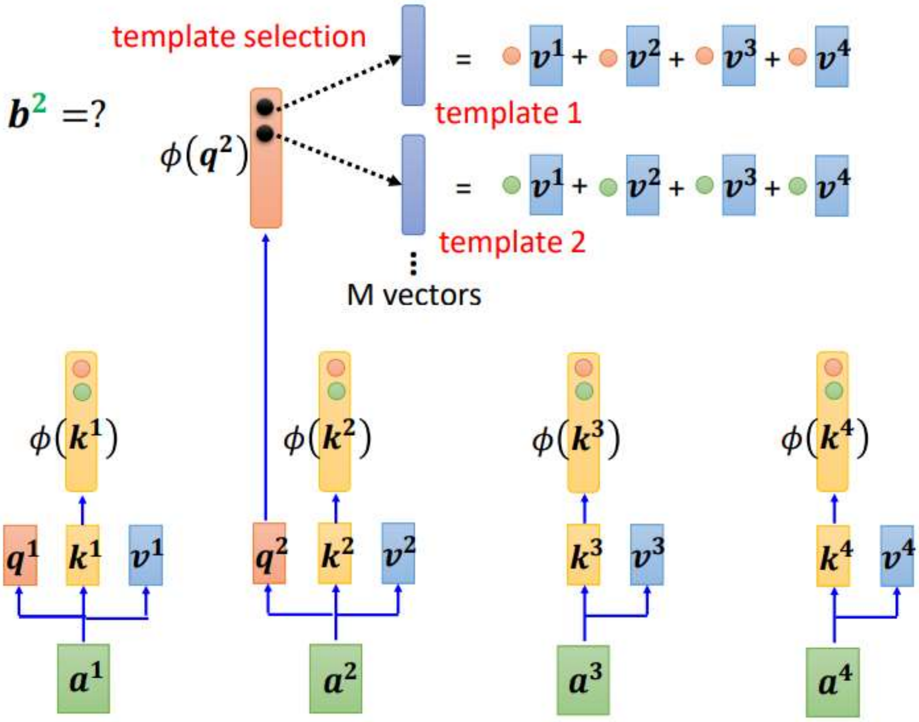
Let's put softmax back ...

End of warning

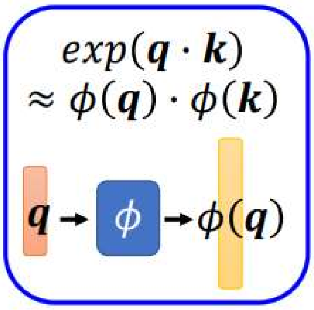

weighted sum
M vectors
M dimensions


weighted sum
M vectors



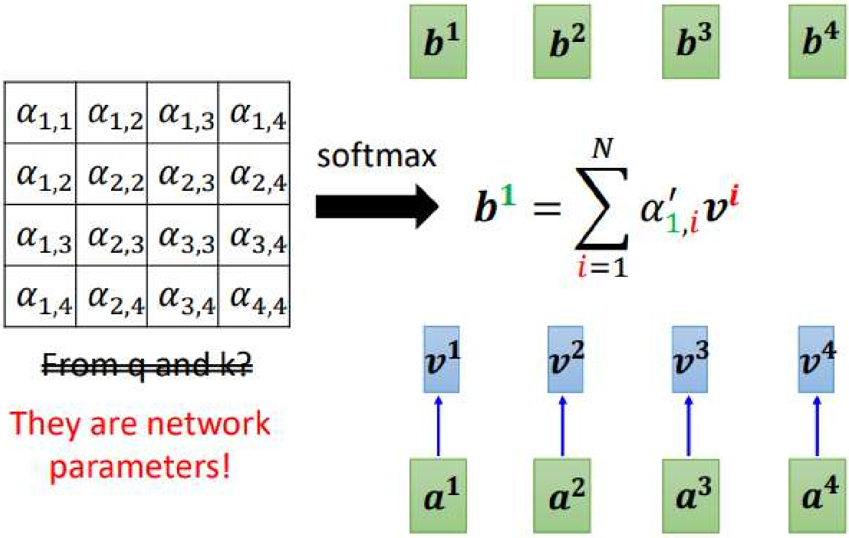


Realization

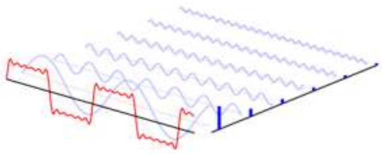


- Efficient attention
  https://arxiv.org/pdf/1812.01243.pdf
- Linear Transformer
  https://linear-transformers.com/
- Random Feature Attention
  https://arxiv.org/pdf/2103.02143.pdf
- Performer
  https://arxiv.org/pdf/2009.14794.pdf

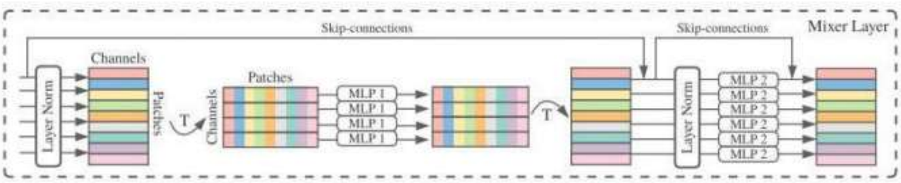
Do we need q and k to compute attention? Synthesizer!

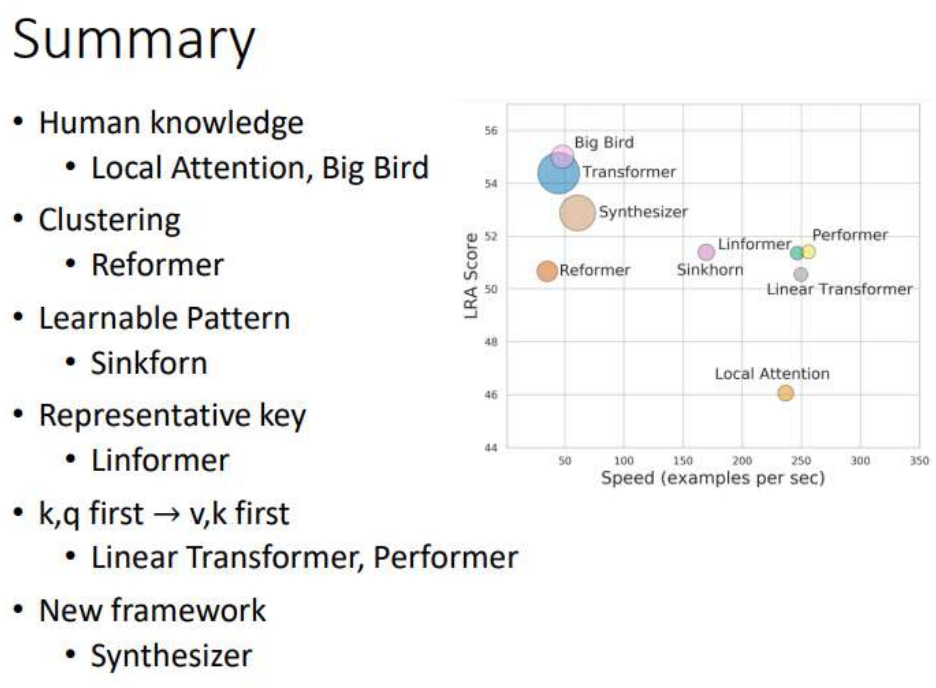


Attention-free?



- Fnet: Mixing tokens with fourier transforms https://arxiv.org/abs/2105.03824
- Pay Attention to MLPs https://arxiv.org/abs/2105.08050
- MLP-Mixer: An all-MLP Architecture for Vision https://arxiv.org/abs/2105.01601





課程網頁 (https://speech.ee.ntu.edu.tw/~hylee/ml/2022-spring.php)

tags: `2022 李宏毅_機器學習`