自然語言處理
Pretrain Language Model

Instructor: 馬豪尚

Outline

- ELMo (Embeddings from Language Models)
- Transformer
- Bert(Bidirectional Encoder Representations from Transformers)

ELMo (Embeddings from Language Models)

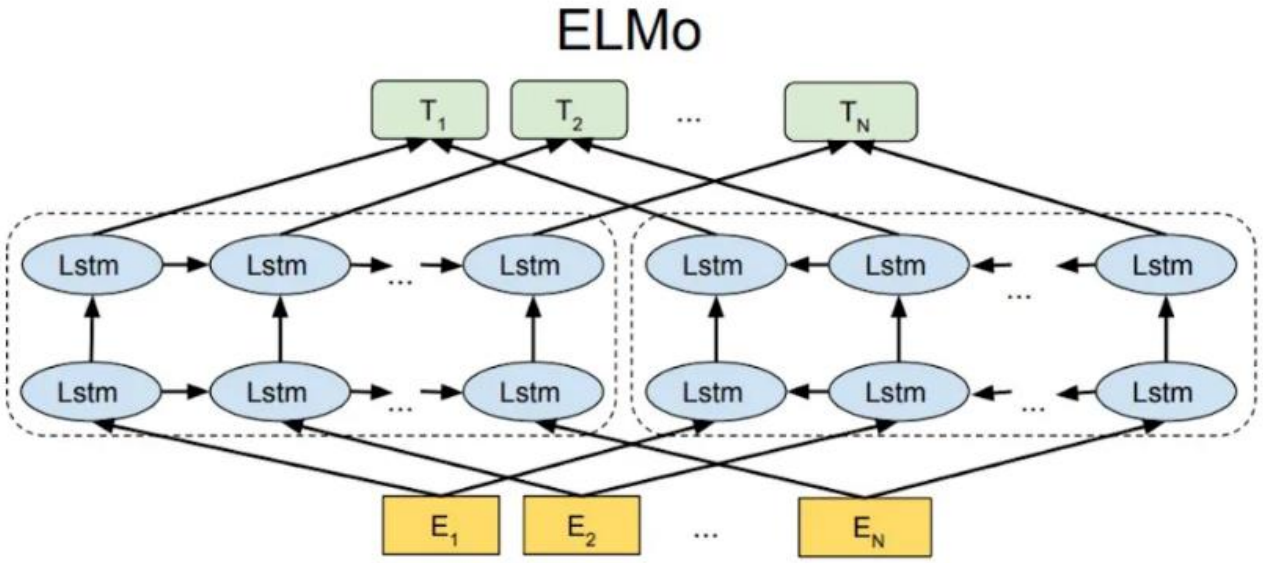
- 在 ELMo 之前，自然語言上的預訓練模型一直停留在淺層的詞向量
- ELMo特點
  - 非監督語言模型：ELMo 選擇了最原始的語言模型(Language Model)，也就是不使用標籤，而是由目前已有的文句預測緊接而來的單字，這樣不僅訓練的文句量大(不須標籤，可直接採用 wiki 等文件庫)，並且也避免了前述標籤以及語意所造成的洩漏
  - 雙向語言訓練：詞向量就是使用非監督的語言模型訓練的模型，如果只是單純用語言模型訓練，必須要有其他方式突破預訓練的深度，而 ELMo 給的答案就是同時參考上下文的「雙向模型」

雙向語言訓練模型BiLM

- RNN/LSTM只從單一個方向來看句子(前->後)，可能造成倒裝句的語意無法正確理解
- 雙向語言模型從前向（從左到右）和後向（從右到左）兩個方向同時建立語言模型
- biLM 的訓練與傳統詞向量訓練的最大差異，就是允許一字多義的存在

ELMo 模型內部架構

- 可以理解為兩個方向分離的 LSTM
- 在輸出的時候，把兩個方向的輸出向量再以單字為準合併
- 雖然與詞向量的輸出都一樣與訓練語句同樣長度，但是其實是包含兩個不同方向的輸出結果，因此在類比時才能滿足 biLM 與上下文都相關的結果



ELMo 模型

- 前向語言模型

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, ..., t_{k-1})$$

- 後向語言模型

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, ..., t_N)$$

- 雙向語言模型

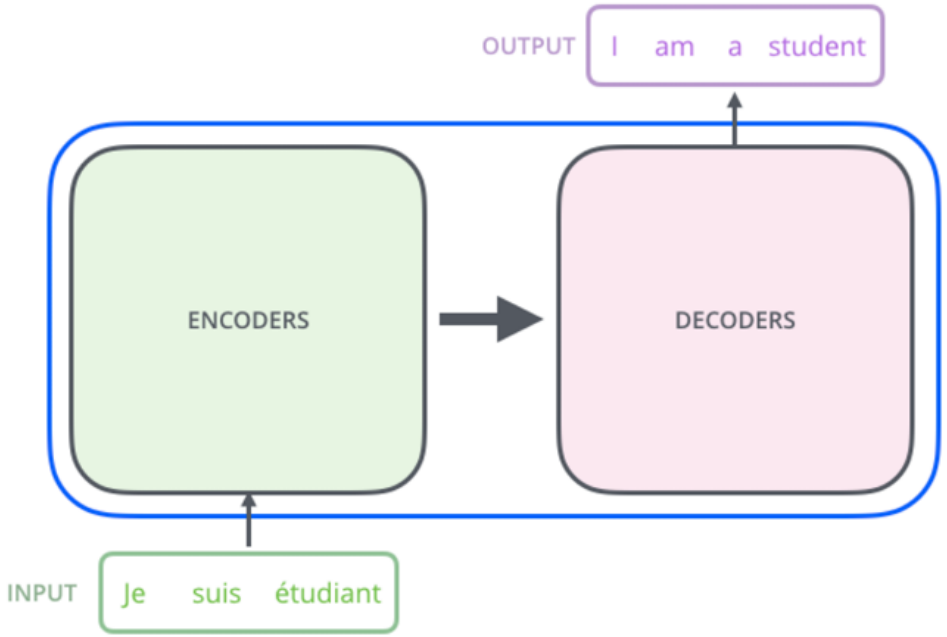
$$\sum_{k=1}^{N} \Big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big)$$

ELMo模型的實作

- 資料準備
  - 使用清洗後並經過分詞等預先處理的語料需要同時建構字詞層級與字元層級的訓練語料，並建立對應的詞表
- 雙向語言模型ELMo
  - 模型的核心是雙向語言模型
  - 編碼器部分主要包括基於字元的輸入表示層以及前向、後向 LSTM 層
- 訓練
  - 在資料、模型組件建置完成後，使用實際資料對模型進行訓練
  - 訓練完成後，便可利用雙向語言模型的編碼器編碼輸入文字並取得動態詞向量

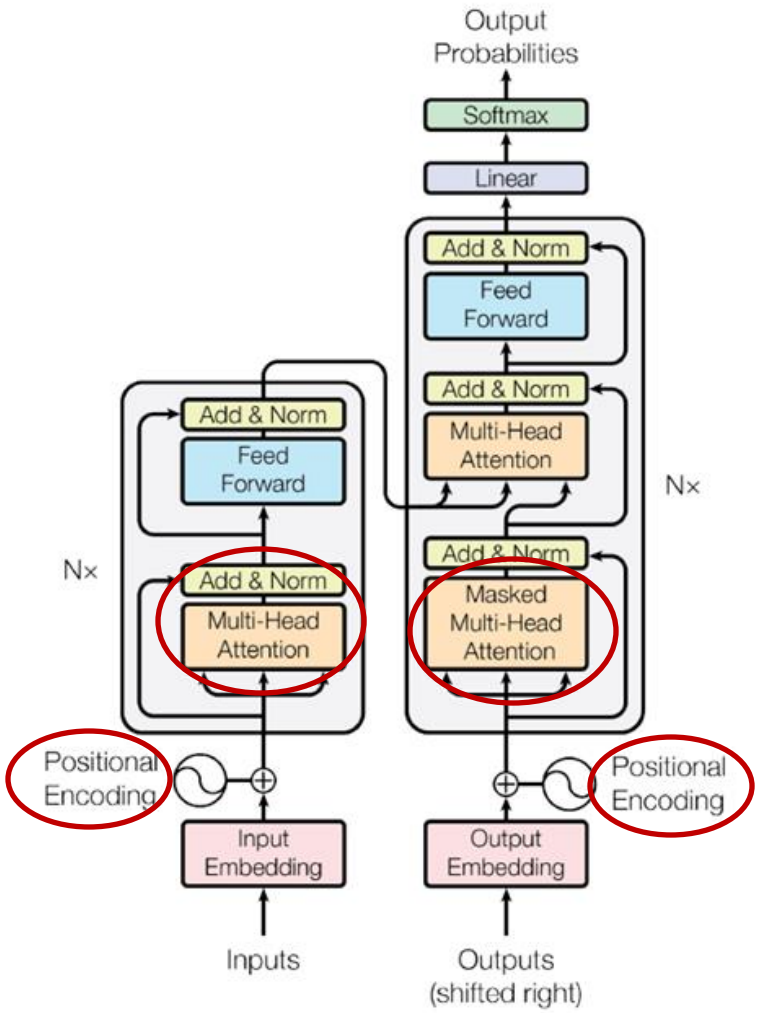
Transformer模型

- Transformer 模型本來是為了翻譯任務而設計的。
- 在訓練過程中，Encoder 接受原始語言的句子作為輸入，而 Decoder 則接受目標語言的翻譯作為輸入



Transformer特色

- 編碼神經網路/解碼神經網路
- 位置編碼(Positional Encoding)
- 自注意力機制(self-attention mechanism)



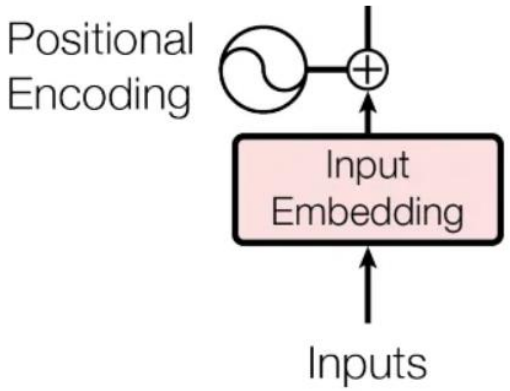
位置編碼(Positional Encoding)

- Positional Encoding 就是詞語的位置代表編碼
- 目的是為了讓模型考慮詞語之間的順序
- 使用 sin, cos 函數進行位置編碼

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

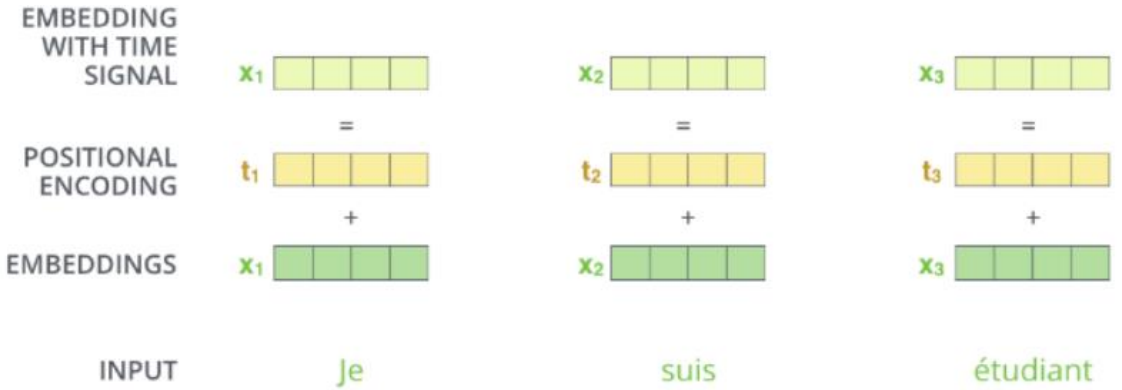
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

  - pos 為詞語在序列中的位置
  - 2i, 2i+1 為該詞語在 Positional Encoding 維度上的 index
  - d_model為 Positional Encoding 的維度



位置編碼(Positional Encoding)

› 舉個例子，假設有一個句子如下，d_model =4



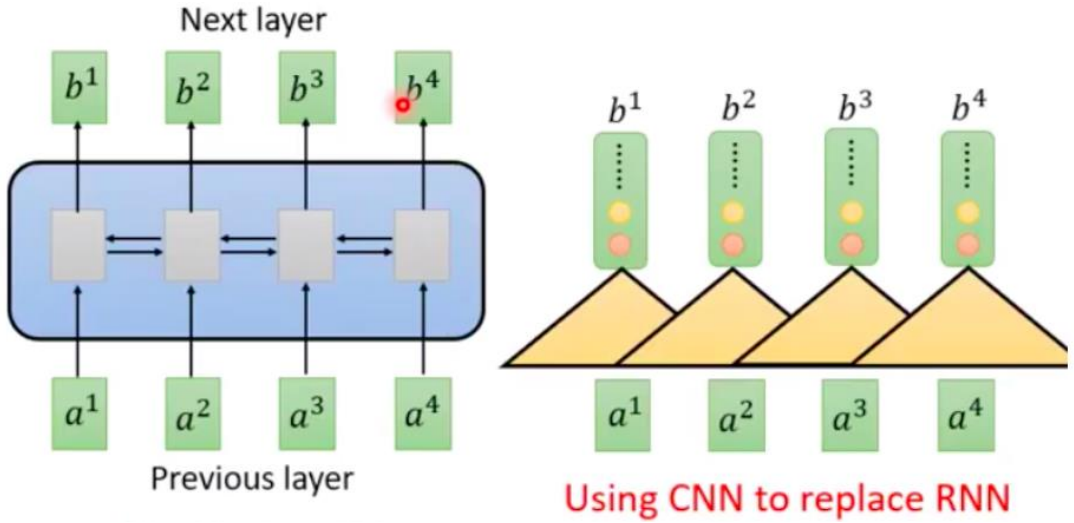
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

$$\overbrace{d\_model = 4}$$

$$P(0) = \left[sin(\frac{0}{10000^{\frac{0}{4}}}), cos(\frac{0}{10000^{\frac{0}{4}}}), sin(\frac{0}{10000^{\frac{1}{4}}}), cos(\frac{0}{10000^{\frac{1}{4}}})\right]$$
$$P(1) = \left[sin(\frac{1}{10000^{\frac{0}{4}}}), cos(\frac{1}{10000^{\frac{0}{4}}}), sin(\frac{1}{10000^{\frac{1}{4}}}), cos(\frac{1}{10000^{\frac{1}{4}}})\right]$$
$$P(2) = \left[sin(\frac{2}{10000^{\frac{0}{4}}}), cos(\frac{2}{10000^{\frac{0}{4}}}), sin(\frac{2}{10000^{\frac{1}{4}}}), cos(\frac{2}{10000^{\frac{1}{4}}})\right]$$

自注意力機制(self-attention)

- RNN
  - 是最經典的處理Sequence的模型，單向RNN或雙向RNN等等。
  - RNN的問題：難以平行處理=>就有人提出用CNN取代RNN
- CNN取代RNN
  - CNN filters：每一個三角形代表一個filter輸入為seq的一小段，輸出一個數值（做內積得到），不同的filter對應seq中不同的部分。

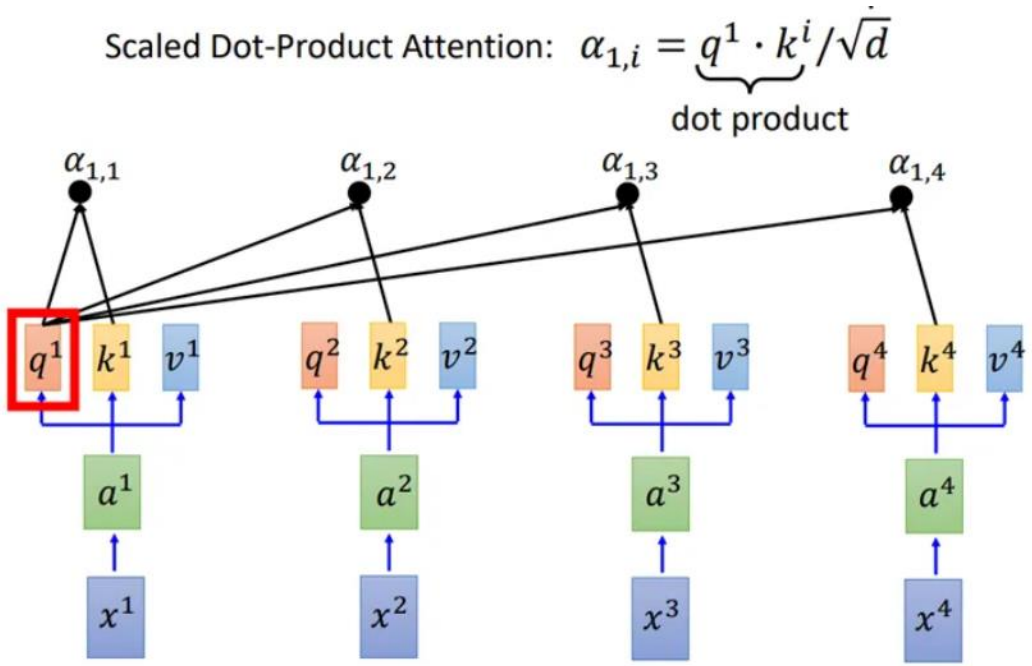


自注意力機制(self-attention)

- 每一個CNN只能考慮有限的內容，RNN能考慮一整個句子
- 考慮很長的句子：疊很多層CNN，上層的filter就可以考慮較多資訊，因為上層的filter會把下層的filter的輸出當作輸入
- 問題：必須疊很多層才有辦法做到考慮較長的句子，因此出現了self-attention機制

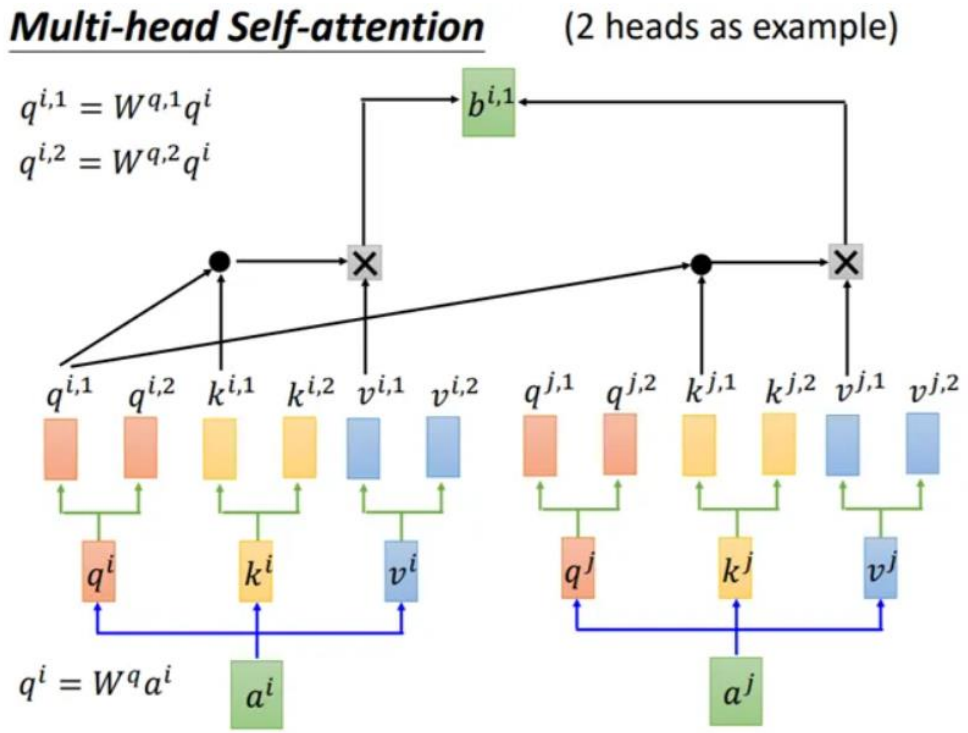
自注意力機制(self-attention)

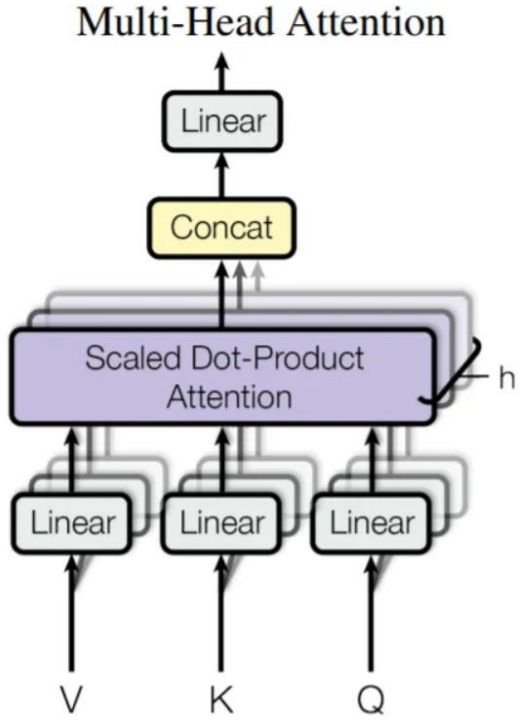
- q (query): 是指當前的詞向量，用於對每個 key 做匹配程度的打分
- k (key): 是指序列中的所有詞向量
- v (value): 是指實際的序列內容
- q1 會對每一個 k 做 Inner Product 得到 q, k 之間匹配的相似程度



Multi-Head Attention

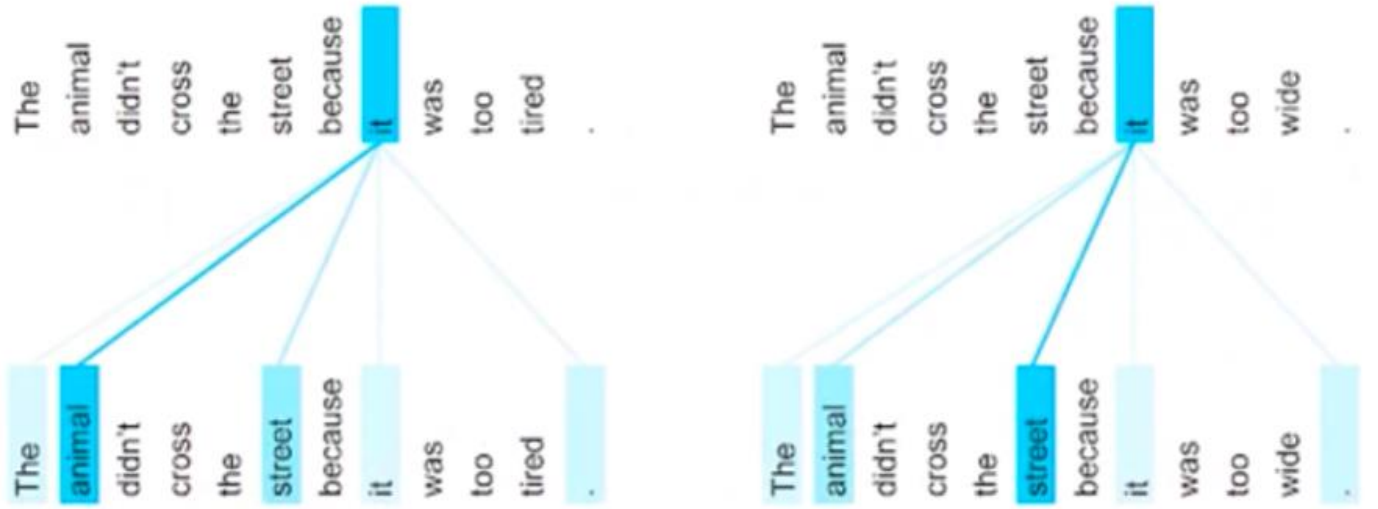
- 運算方式與 self-attention mechanism 相同，差異在於會先將 q, k, v 拆分成多個低維度的向量， q, k, v 會做 h 次的線性映射到低維度的向量，再進行 Scaled Dot-Product Attention，最後將其 concat、linear 得到輸出
- 每個head所關注的資訊不同，有的只關心local資訊（鄰居的資訊），有的只關心global（較長時間）資訊等等





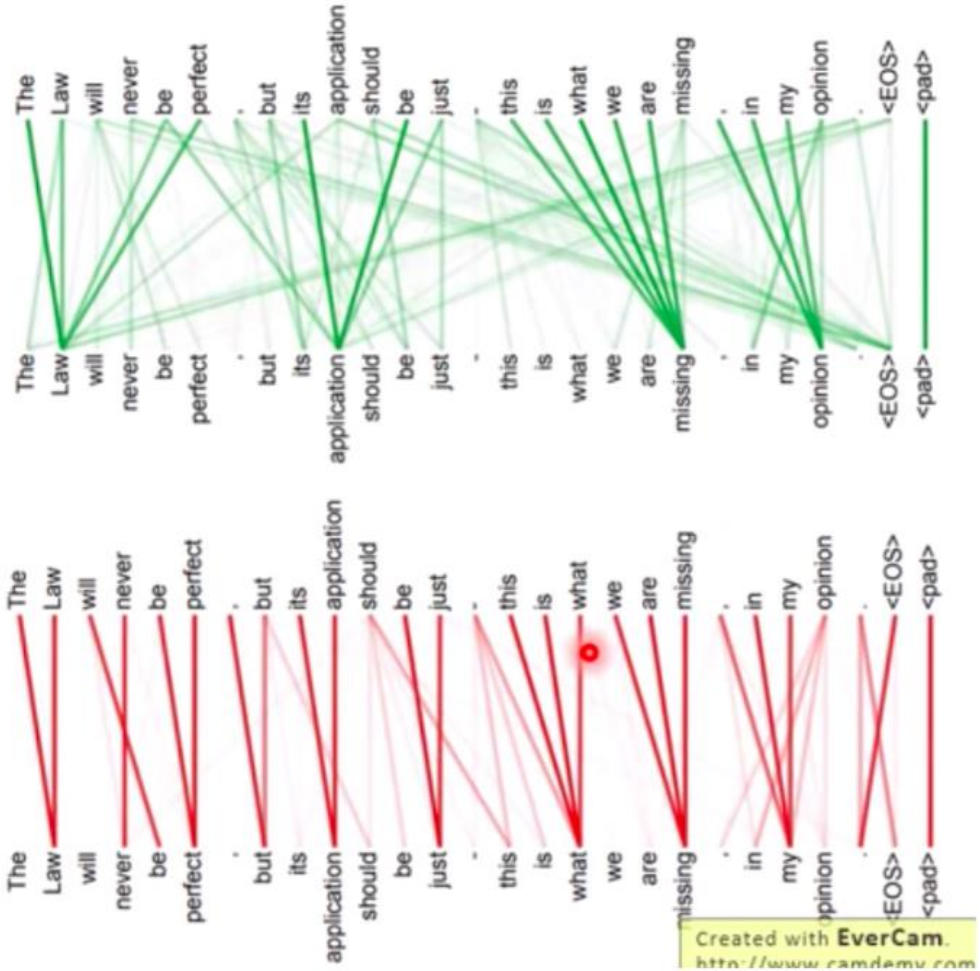
自注意力結果(single self-attention)

› 文字倆倆之間的關係，線條越粗代表關聯越深



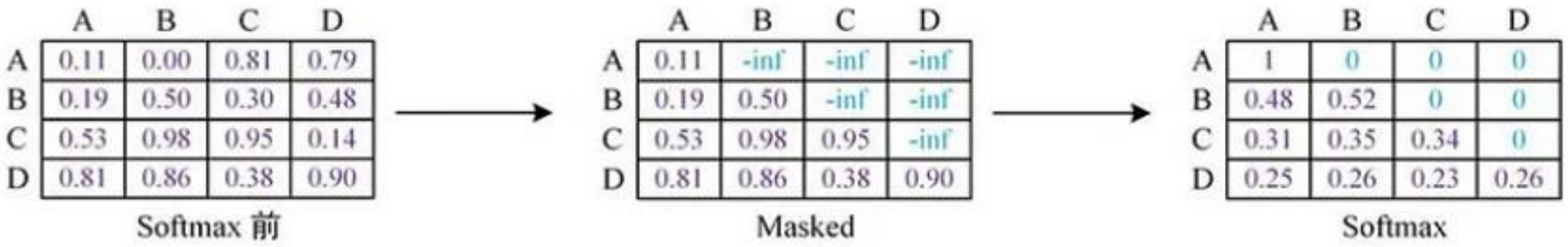
自注意力結果(multi-head self-attention)

- 用不同組的q.k配對出來的結果有所不同，代表不同組的q.k擁有不同資訊



Masked Multi-head Attention

- Padding Mask
  - Padding Mask 在 Encoder和 Decoder 中都有使用到，目的是為了限制每個輸入的長度要相同，對於較短的句子會將不足的部分補 0
- Sequence Mask
  - 只用於 Decoder，目的是為了防止模型看到未來的資訊，因此在超過當前時刻 t 的輸出會加上一個 mask，確保模型的預測只依賴小於當前時刻的輸出。



	A	B	C	D
A	0.11	0.00	0.81	0.79
B	0.19	0.50	0.30	0.48
C	0.53	0.98	0.95	0.14
D	0.81	0.86	0.38	0.90

Softmax 前

	A	B	C	D
A	0.11	-inf	-inf	-inf
B	0.19	0.50	-inf	-inf
C	0.53	0.98	0.95	-inf
D	0.81	0.86	0.38	0.90

Masked

	A	B	C	D
A	1	0	0	0
B	0.48	0.52	0	0
C	0.31	0.35	0.34	0
D	0.25	0.26	0.23	0.26

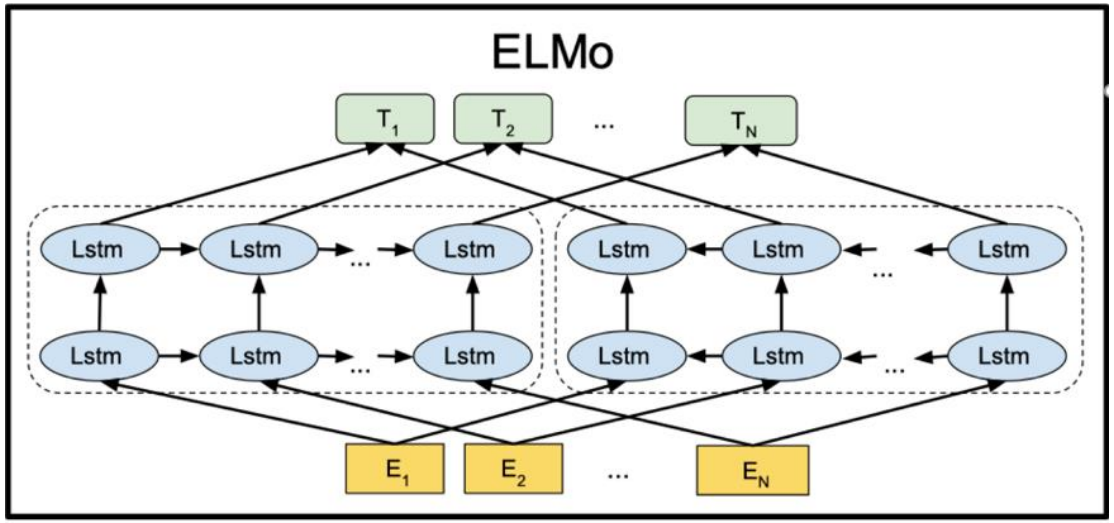
Softmax

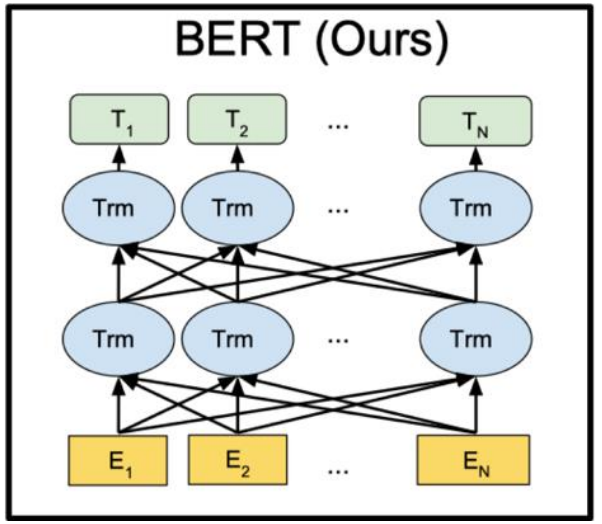
BERT: Bidirectional Encoder Representations from Transformers

- 提出了雙向預訓練語言模型方法
- 利用大規模自由文本訓練兩個無監督預訓練任務
- BERT在眾多NLP任務中獲得了顯著效能提升
- 進一步強調了使用通用預訓練取代繁雜的任務特定的模型設計

BERT/ELMo之間的對比

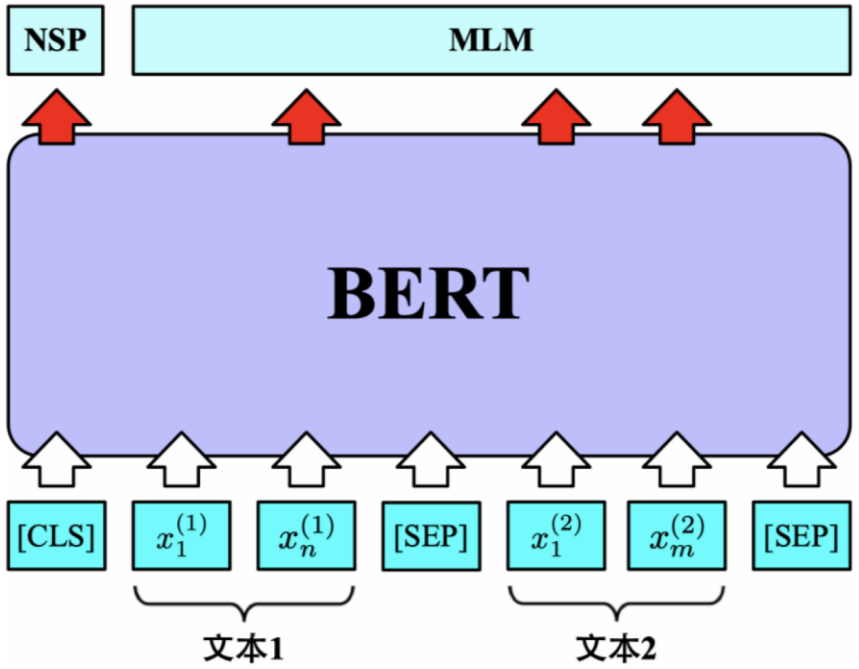
- ELMo: 將獨立的前向和後向的LSTM語言模型拼接所得
- BERT: 雙向 Transformer語言模型





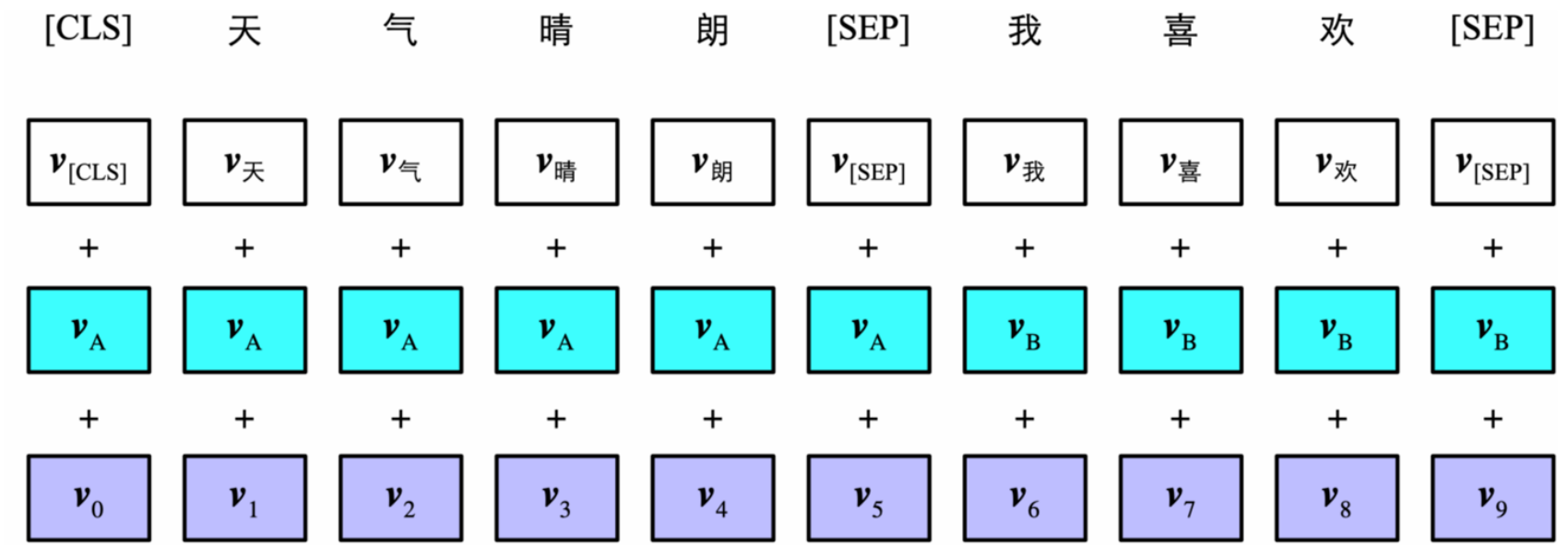
BERT模型架構

- 整體結構
  - 由深層Transformer模型構成
    - base：12層，參數量110M
    - large：24層，參數量330M
- 預訓練任務
  - 遮蔽語言模型（Masked Language Model, MLM）
  - 下一個句子預測（Next Sentence Prediction, NSP）



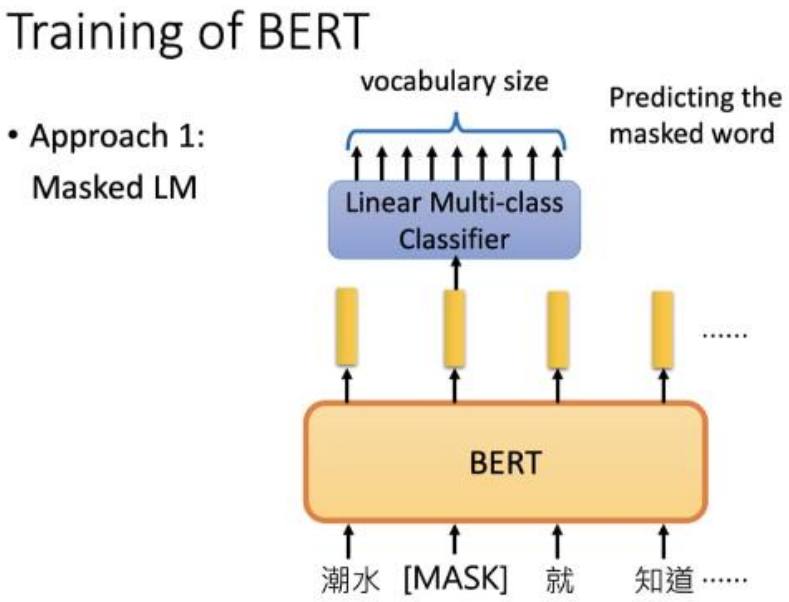
BERT模型輸入

- BERT的輸入表示由三個部分組成
  - 詞向量：透過詞向量矩陣將輸入文字轉換為實值向量
  - 表示句向量：編碼當前詞屬於哪一個片段
  - 位置向量：編碼當前詞的位置



遮蔽語言模型（Masked Language Model)

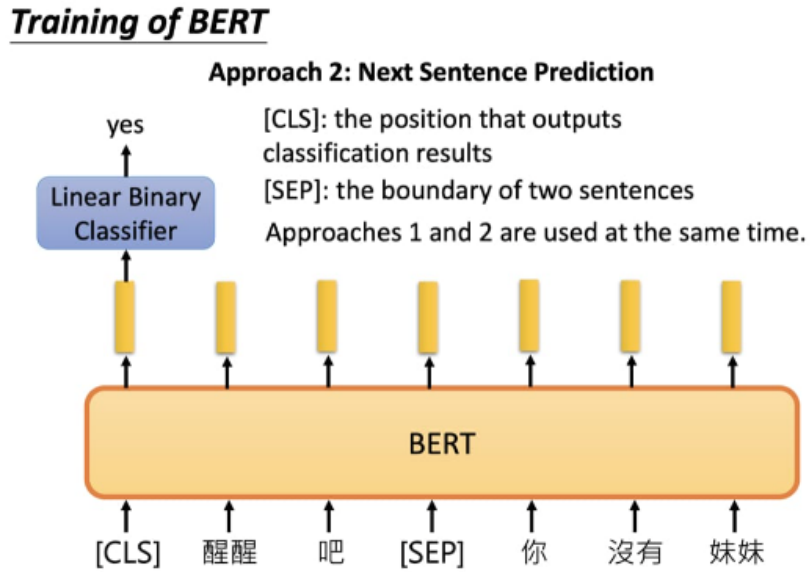
- 將輸入序列中的部分token進行遮蔽，並且要求模型將它們進行還原
  - 會將15%的輸入文字進行mask
    - 以 80% 的機率替換為 [MASK] 標記
    - 以 10% 的機率替換為詞表中的任一個隨機詞
    - 以 10% 的機率保持原詞不變，即不替換



下一個句子預測 (Next Sentence Prediction)

- 學習兩段文本之間的關係（上下文資訊）
- 預測Sentence B是否是Sentence A的下一個句子
  - 正樣本：文本中相鄰的兩個句子"句子 A"和"句子 B"，構成"下一個句子"關係
  - 負樣本：將「句子 B」替換為語料庫中任一個句子，構成「非下一句子」關係

	正樣本	負樣本
句子 A	The man went to the store.	The man went to the store.
句子 B	He bought a gallon of milk.	Penguins are flightless.



預訓練語言模型應用

- 特徵擷取(feature extraction)
  - 僅利用BERT提取輸入文字特徵，產生對應的上下文語意表示
  - BERT本身不參與目標任務的訓練，即BERT部分只進行解碼（無梯度回傳）
- 模型微調(fine-tuning)
  - 利用BERT作為下游任務模型基底，產生文字對應的上下文語意表示
  - 參與下游任務的訓練，即在下游任務學習過程中，BERT對自身參數進行更新通常使用「模型微調」的方法，因其效果更佳