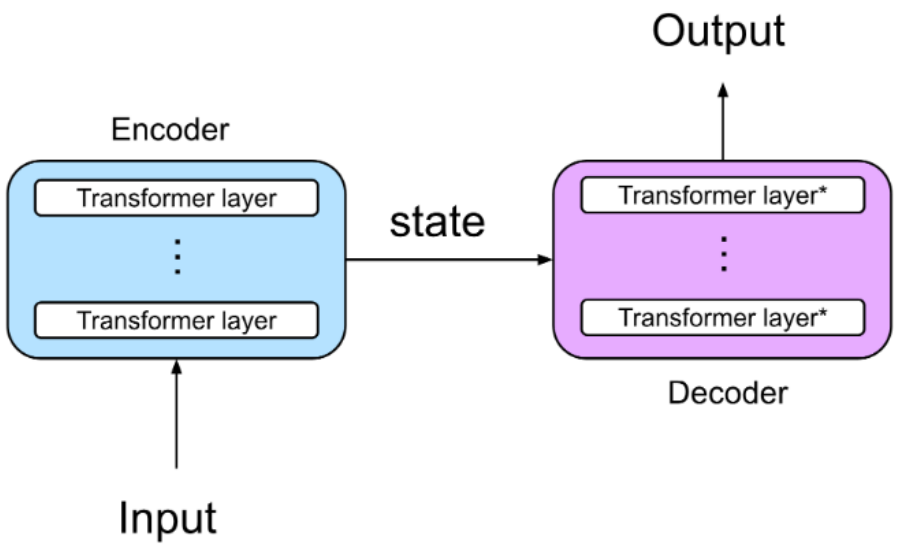
Transformer

Transformer 是一種基於自注意力機制（Self-Attention）的深度學習模型架構，最早由 Vaswani 等人在 2017 年提出，並在各種自然語言處理（NLP）任務中取得了顯著的成功。其主要創新是使用了完全基於注意力機制的結構，取代了傳統的循環神經網絡（RNN）和長短期記憶網絡（LSTM）。這使得 Transformer 在處理長程依賴和並行計算方面有著顯著優勢。

Transformer 的主要結構

Transformer 模型由**編碼器（Encoder）**和**解碼器（Decoder）**兩部分組成。每個編碼器和解碼器層由兩個主要的子結構組成：**自注意力（Self-Attention）和前饋神經網絡（Feedforward Neural Network）**。下面將詳細介紹 Transformer 的各個部分及其對應的數學公式。



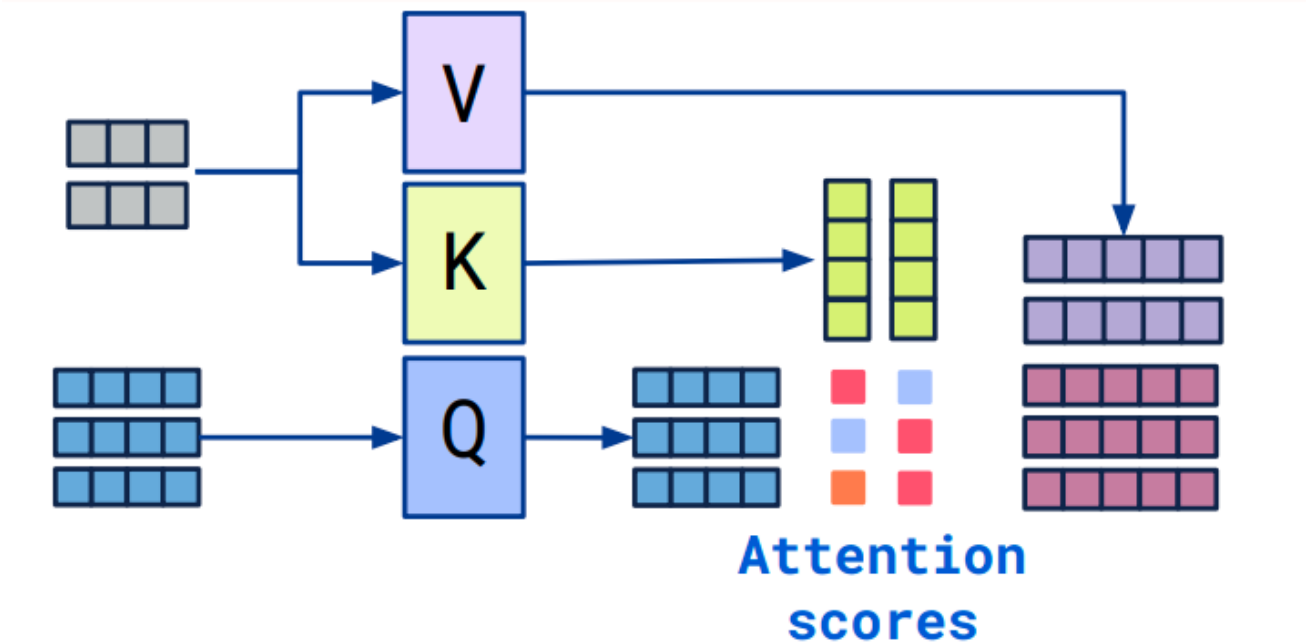
1. 自注意力機制（Self-Attention）

自注意力是 Transformer 的核心，它允許模型在處理某個詞時，考慮到該詞與序列中其他所有詞的關係。對於給定的輸入序列 $X = \{x_1, x_2, \ldots, x_n\}$，自注意力機制首先將每個詞向量 x_i 映射為三個向量：**查詢向量（Query）** Q、**鍵向量（Key）** K 和**值向量（Value）** V。

- **查詢、鍵和值的計算：**

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

其中 W_Q, W_K, W_V 是可學習的權重矩陣，X 是輸入的詞向量矩陣。



1. Query（查詢向量）

- **作用**：表示當前需要關注的內容，類似於提出問題或查詢的向量。
- **來源**：通常是來自輸入序列中某個詞或片段的編碼表示，經由線性變換生成。
- **功能**：與 Key 進行相似度計算，決定輸入序列中哪些部分與 Query 關聯性更高。

2. Key（鍵向量）

- **作用**：提供參考的標準，用於衡量輸入序列中每個詞或片段與 Query 的相關性。
- **來源**：由輸入序列中每個詞或片段的表示經線性變換生成。
- **功能**：與 Query 一起計算注意力分數（通常使用點積方式），以確定哪些資訊需要被關注。

3. Value（值向量）

- **作用**：包含具體的信息，用於生成最終的加權輸出。
- **來源**：同樣來自輸入序列中每個詞或片段的表示，經線性變換生成。
- **功能**：根據注意力分數分配權重，將相關資訊組合起來生成輸出。

Self-attention

input #1

1	0	1	0

input #2

0	2	0	2

input #3

1	1	1	1

- **計算注意力權重**： 計算查詢 Q 和鍵 K 之間的關聯性，並通過 Softmax 函數得到注意力權重：

注意力分數

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

注意力權重

其中，d_k 是鍵向量的維度，這一公式通過計算查詢與鍵的內積來衡量它們的相似度，然後通過 Softmax 進行歸一化，使得權重和為 1。

- **多頭注意力**： Transformer 中使用了**多頭注意力**機制，將自注意力計算並行化，使用多個注意力頭來捕捉不同的語義信息。每個注意力頭都有不同的查詢、鍵和值的投影矩陣，並且最終將各個頭的結果拼接後經過線性變換：

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W_O$$

其中，每個頭的計算為：

$$\text{head}_i = \text{Attention}(QW_Q^i, KW_K^i, VW_V^i)$$

h 是注意力頭的數量，W_O 是最終的投影矩陣。

2. 前饋神經網絡（Feedforward Neural Network）

每個自注意力層後面會接上一個前饋神經網絡，這個網絡通常包含兩個線性變換和一個激活函數（通常使用 ReLU）：

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

其中 W_1, W_2 是可學習的權重矩陣，b_1, b_2 是偏置項。

3. 編碼器（Encoder）

Transformer 的編碼器由多層相同結構的編碼器組成。每個編碼器層包括兩個主要部分：

1. **自注意力層**：用於捕捉序列中詞語之間的相互依賴關係。
2. **前饋神經網絡層**：用於進一步處理注意力的輸出。

每個編碼器層的輸入是來自上一層（或初始輸入）的輸出，並且在每一層中都會進行**殘差連接**（Residual Connection）和**層正則化**（Layer Normalization），以幫助訓練過程中的穩定性。

4. 解碼器（Decoder）

解碼器的結構與編碼器相似，不過它還額外引入了編碼器與解碼器之間的注意力機制，以便根據編碼器的輸出生成最終的預測。

1. **自注意力層**：這一層的計算與編碼器相同，不過解碼器中的自注意力機制會進行遮蔽（Masking），以防止模型在生成序列時看到未來的詞語。
2. **編碼-解碼注意力層**：這一層計算解碼器中每個位置對編碼器輸出的注意力。它的計算過程如下：

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

   這裡的 Q 是解碼器的查詢向量，K 和 V 是編碼器的鍵和值。

3. **前饋神經網絡層**：與編碼器中的前饋層相同。

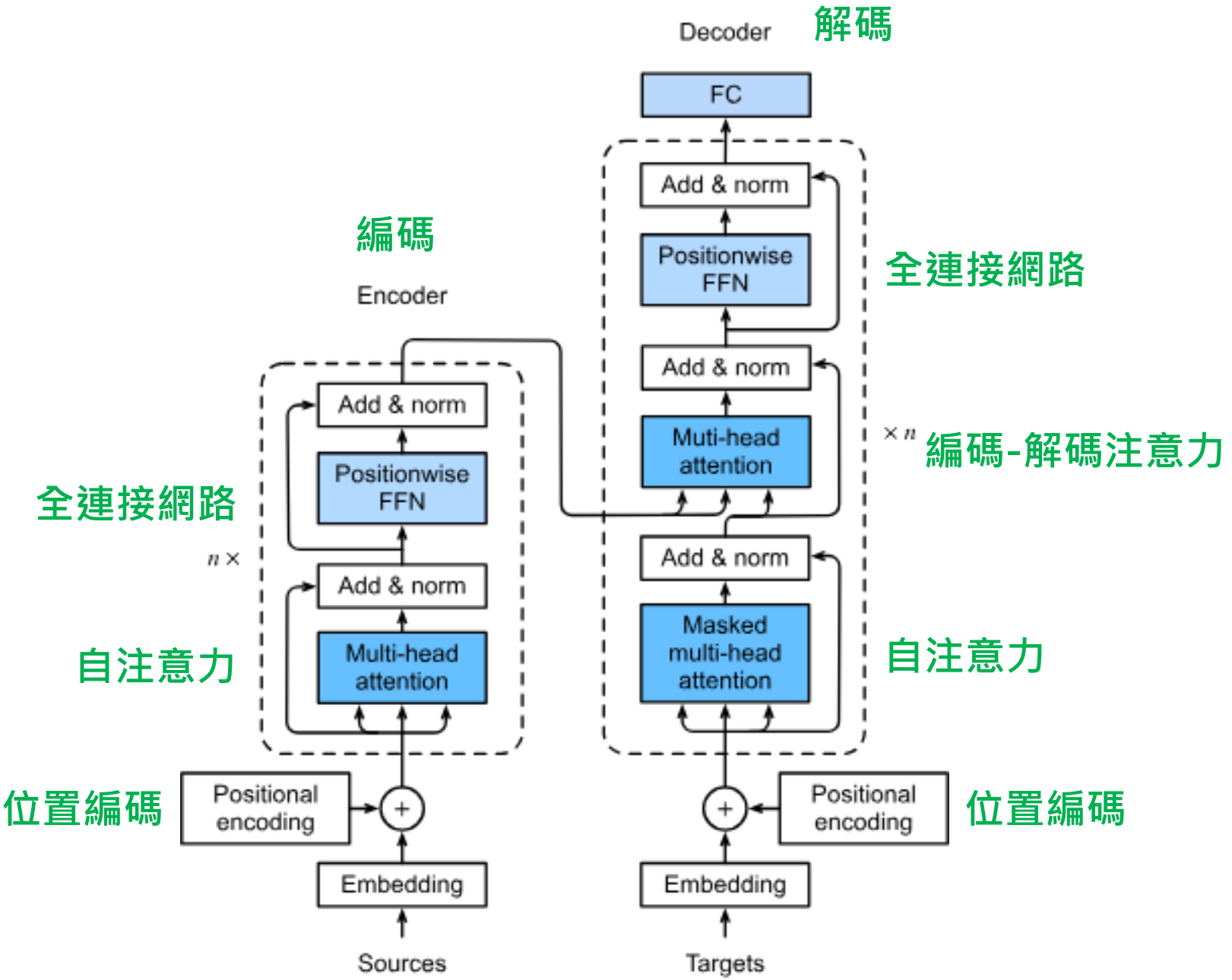
5. 位置編碼（Positional Encoding）

由於 Transformer 缺乏循環結構，因此無法像 RNN 或 LSTM 那樣天然地處理序列順序。為了解決這個問題，Transformer 引入了位置編碼（Positional Encoding），將每個詞的位置信息加入到其詞向量中。位置編碼的計算公式如下：

$$\mathrm{PE}(t, 2i) = \sin\left(\frac{t}{10000^{2i/d_{\text{model}}}}\right), \quad \mathrm{PE}(t, 2i+1) = \cos\left(\frac{t}{10000^{2i/d_{\text{model}}}}\right)$$

總結

Transformer 以其強大的自注意力機制和多頭注意力結構，成功解決了長程依賴問題，並且支持並行計算，使得它在處理大規模序列數據時表現出色。無論是在機器翻譯、文本生成還是其他自然語言處理任務中，Transformer 都取得了顯著的成果，其優越性已經使其成為現代 NLP 任務的基礎模型架構。


Decoder
解碼
FC
Add & norm
Positionwise FFN
全連接網路
Add & norm
Muti-head attention
× n
編碼-解碼注意力
Add & norm
Masked multi-head attention
自注意力
Positional encoding
位置編碼
Embedding
Targets
編碼
Encoder
Add & norm
Positionwise FFN
全連接網路
n ×
Add & norm
Multi-head attention
自注意力
Positional encoding
位置編碼
Embedding
Sources

字詞

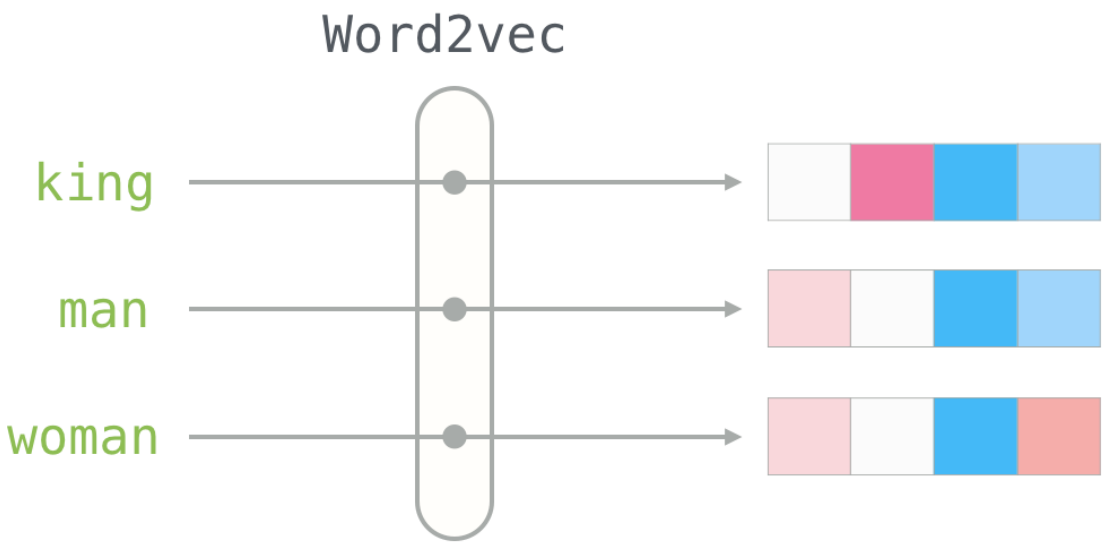
id	color
1	red
2	blue
3	green
4	blue

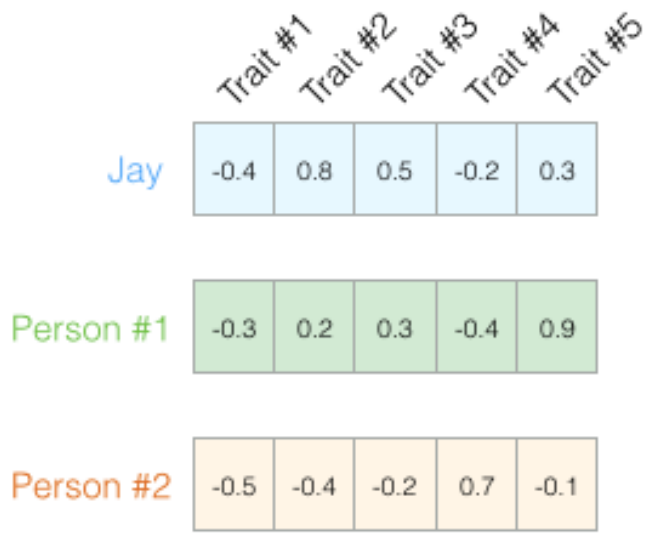
One Hot Encoding

向量

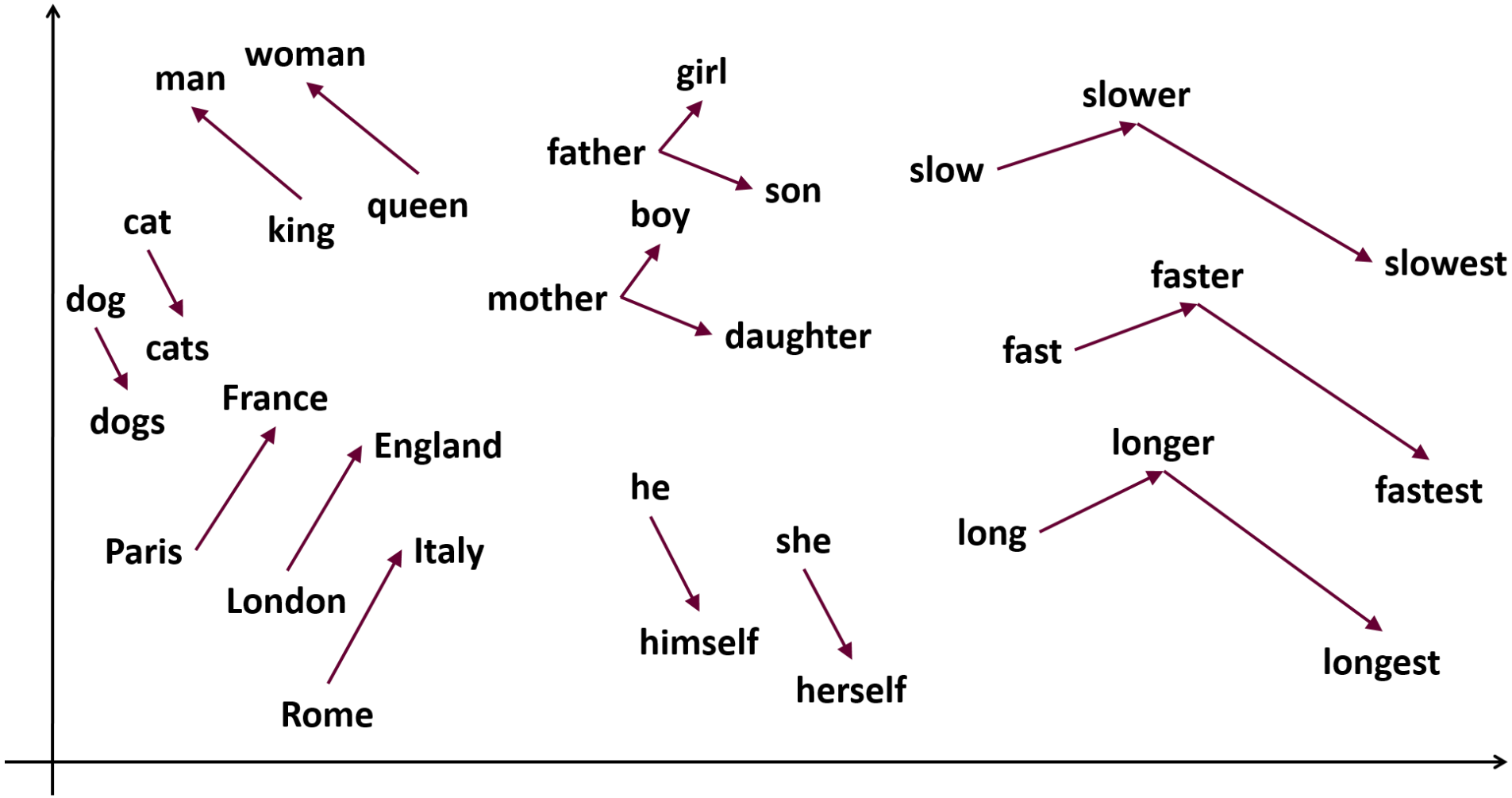
id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

簡單、高維度、無關聯





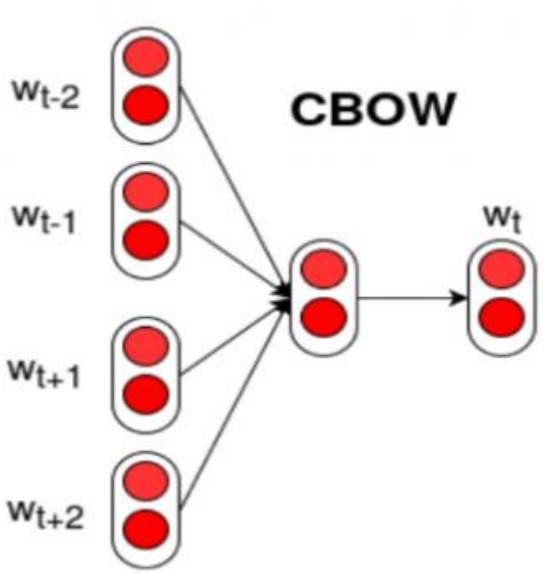
1. 透過學習大量文本資料，將字詞用數學向量的方式來代表他們的語意。
2. 將字詞嵌入到一個空間後，讓語意相似的單字可以有較近的距離。

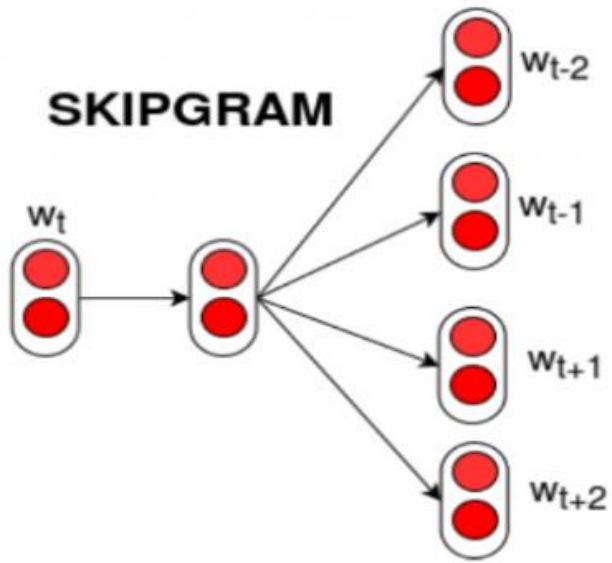

man
woman
king
queen
girl
father
son
boy
mother
daughter
slow
slower
slowest
cat
cats
dog
dogs
France
Paris
England
London
Italy
Rome
he
himself
she
herself
faster
fast
fastest
long
longer
longest

1. Word2Vec 概述

Word2Vec 是由 Google 提出的詞向量模型，通過神經網絡學習詞語的低維密集向量表示，捕捉詞之間的語義關係。它有兩種主要架構：

- Skip-gram
- Continuous Bag of Words (CBOW)





$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_t|w_{t+j})$$

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t)$$

2. Skip-gram 模型

目標

給定中心詞 w_t，預測上下文詞 w_{t+j}，最大化條件概率：

$$P(w_{t+j}|w_t)$$

模型公式

條件概率通過 softmax 定義為：

$$P(w_O|w_I) = \frac{\exp(\mathbf{v}_O^\top \mathbf{v}_I)}{\sum_{w \in V} \exp(\mathbf{v}_w^\top \mathbf{v}_I)}$$ Softmax函數

- w_I：中心詞（Input Word）。
- w_O：上下文詞（Output Word）。
- $\mathbf{v}_I$：中心詞的詞向量。
- $\mathbf{v}_O$：上下文詞的詞向量。
- V：詞彙表的大小。

損失函數

對整個語料的損失函數為負對數似然：

$$L = -\sum_{t=1}^{T} \sum_{j=-c, j\neq 0}^{c} \log P(w_{t+j}|w_t)$$

- T：語料中詞的總數。
- c：窗口大小。

最大化對數似然

3. CBOW 模型

目標

與 Skip-gram 相反，CBOW 根據上下文詞 $w_{t-c}, \ldots, w_{t+c}$ 預測中心詞 w_t。

模型公式

條件概率為：

$$P(w_t | w_{t-c}, \ldots, w_{t+c}) = \frac{\exp(\mathbf{v}_t^\top \mathbf{v}_C)}{\sum_{w \in V} \exp(\mathbf{v}_w^\top \mathbf{v}_C)}$$

其中：

$$\mathbf{v}_C = \frac{1}{2c} \sum_{j=-c, j \neq 0}^{c} \mathbf{v}_{t+j}$$

- $\mathbf{v}_C$：上下文向量的平均值。

損失函數

$$L = -\sum_{t=1}^{T} \log P(w_t | w_{t-c}, \ldots, w_{t+c})$$

king - man + woman ~= queen

語意相似性捕捉能力

Word Embedding

16:11

Word Embedding and Word2Vec, Clearly Explained!!!

1. Seq2Seq 模型概述

Seq2Seq 是一種用於處理序列輸入和輸出任務的模型架構，通常應用於：

- 機器翻譯
- 聊天機器人
- 文本摘要

核心結構包括：

1. **編碼器（Encoder）**：將輸入序列編碼為固定大小的上下文向量（Context Vector）。
2. **解碼器（Decoder）**：根據上下文向量生成輸出序列。

2. 模型結構與公式

編碼器

編碼器通常使用 RNN、LSTM 或 GRU，逐步處理輸入序列 $\mathbf{x} = (x_1, x_2, \ldots, x_T)$，生成隱狀態序列 $\mathbf{h} = (h_1, h_2, \ldots, h_T)$。

每個時間步的計算為：

$$h_t = f(h_{t-1}, x_t)$$

- h_t：時間步 t 的隱藏狀態。
- f：RNN 單元，如 LSTM 或 GRU 的遞迴函數。

最終，編碼器將輸入序列壓縮為上下文向量 c，通常為最後的隱藏狀態 h_T：

$$c = h_T$$

解碼器

解碼器同樣使用 RNN，根據上下文向量 c 和之前生成的輸出序列 $\mathbf{y} = (y_1, y_2, \ldots, y_{T'})$ 進行生成。

每一步的隱藏狀態計算為：

$$s_t = f(s_{t-1}, y_{t-1}, c)$$

- s_t：解碼器時間步 t 的隱藏狀態。

生成輸出的條件概率為：

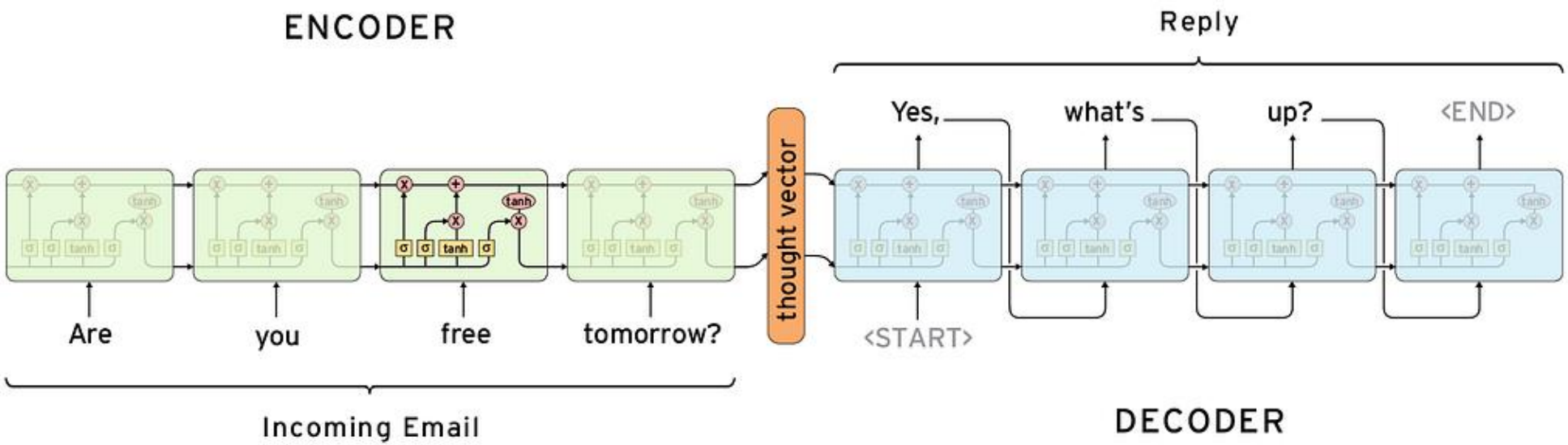
$$P(y_t | y_1, \ldots, y_{t-1}, c) = \text{softmax}(W_o s_t)$$

- W_o：輸出層的權重矩陣。

3. 損失函數

對於給定的輸入序列 $\mathbf{x}$ 和目標輸出序列 $\mathbf{y}$，損失函數為序列的交叉熵損失：

$$L = -\sum_{t=1}^{T'} \log P(y_t | y_1, \ldots, y_{t-1}, c)$$



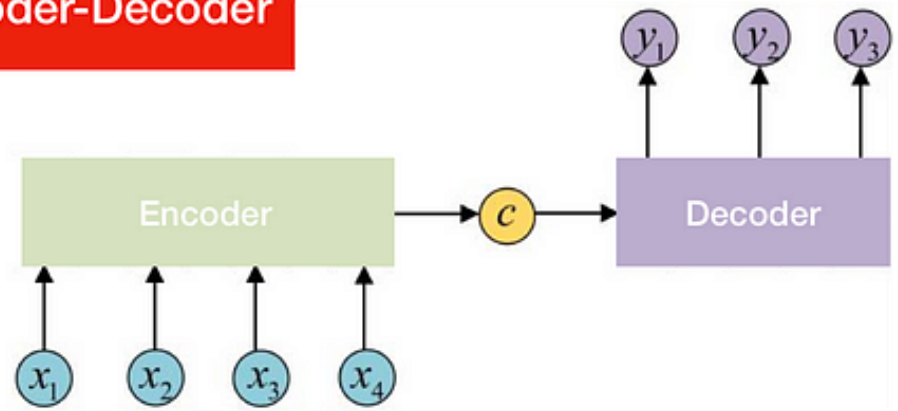
Sequence to Sequence

16:49

Seq2seq and Encoder-Decoder Neural Networks: Clearly Explained!!!

Attention Mechanism

Encoder-Decoder

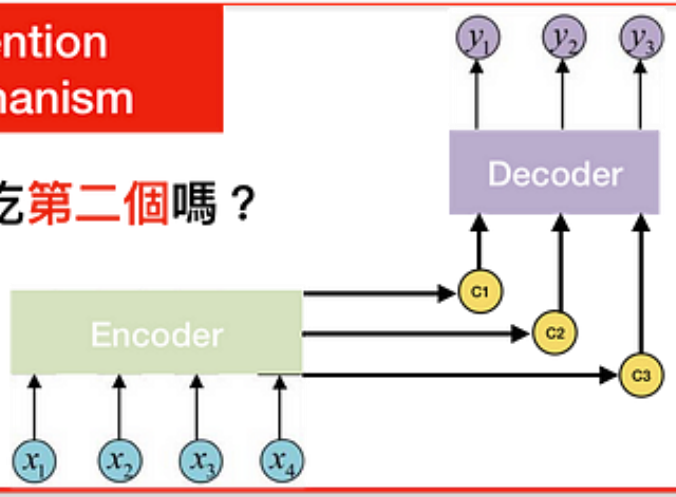


吃一個含所有信息的**Context vector**，訓練結果不好

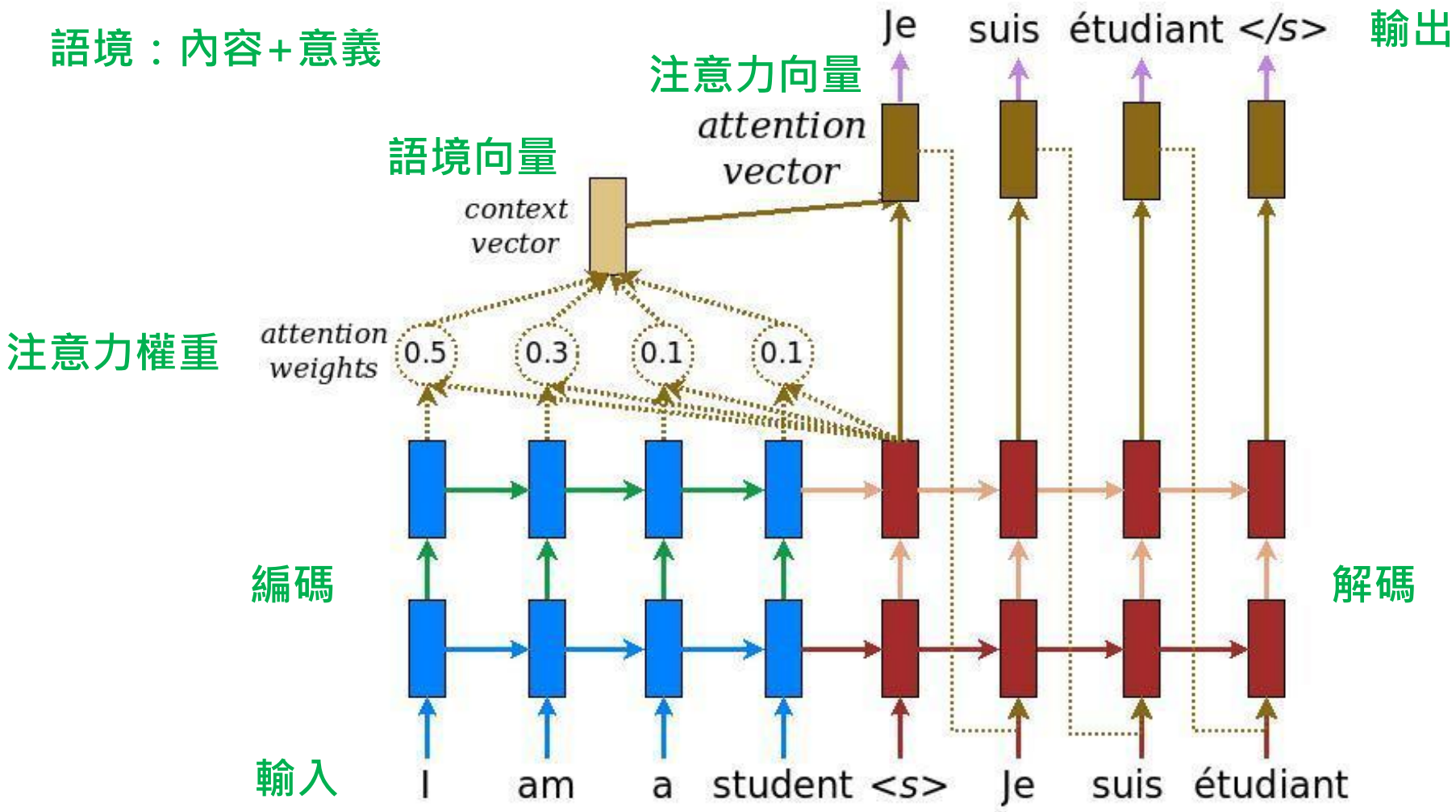
總統，
現在只吃一個
Context vector，
模型訓練不好，
怎麼辦？

Attention Mechanism

那你有吃第二個嗎？



Seq2seq with attention



4. 注意力機制（Attention Mechanism）

原始的 Seq2Seq 模型使用固定的上下文向量 c，在長序列上表現有限。注意力機制通過動態權重分配來改進模型。

上下文向量

上下文向量 c_t 在解碼過程中由加權的編碼器隱狀態組成：

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} h_i$$

- $\alpha_{t,i}$：注意力權重，表示時間步 t 的解碼器對編碼器隱狀態 h_i 的關注程度。

注意力權重

注意力權重通過對齊函數 $e_{t,i}$ 計算，並經過 softmax 正規化：

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{T} \exp(e_{t,j})}$$

對齊函數 $e_{t,i}$ 的常見形式：

$$e_{t,i} = v_a^\top \tanh(W_a s_{t-1} + U_a h_i)$$

- v_a, W_a, U_a：可學參數。

Attention Neural Network

 15:50

Attention:
Clearly Explained!!!

5. 模型變體

基於 Transformer 的 Seq2Seq

- 使用自注意力（Self-Attention）取代 RNN。
- 編碼器和解碼器基於多頭注意力機制和前饋神經網絡。

輸入序列與輸出序列通過注意力權重相互作用，公式為：

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

- Q, K, V：查詢、鍵、值矩陣。
- d_k：鍵的維度，用於縮放。

Positional Encoding（位置編碼）

Positional Encoding 是 Transformer 模型中的關鍵組件，旨在向詞嵌入中引入序列中詞語的位置信息。這是因為 Transformer 的自注意力機制是並行運算的，無法直接捕捉輸入序列中詞語的順序。

1. Positional Encoding 的公式

位置編碼是一個固定的函數，將每個位置 pos 映射到向量 $PE(\text{pos})$。公式如下：

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

索引/維度：位置的編碼

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

經驗值

其中：

- pos：序列中詞的絕對位置。
- i：向量維度中的索引。
- d_{model}：詞嵌入的維度。

資料維度大、頻率範圍大、編碼複雜度高

2. Positional Encoding 的直觀理解

1. **不同維度的頻率範圍**
   - 位置 pos 被轉換成多維度的正弦和餘弦值，每個維度的頻率由 $\frac{1}{10000^{2i/d_{\text{model}}}}$ 控制。
   - 高頻分量（低維）用於表示短距離位置信息，低頻分量（高維）用於表示長距離位置信息。
2. **正交性與穩定性**
   - 正弦和餘弦的週期性使不同位置的表示具有區分性。
   - 兩個位置的相對位置信息可以通過向量之間的線性運算提取。

這裡的 $10000^{2i/d_{\text{model}}}$ 是一個指數縮放因子，根據維度 i 的索引進行調整。

- 當 $i = 0$ 時，分母接近 1，頻率最高（快速振盪）。
- 當 $i = d_{\text{model}} - 1$ 時，分母接近 10000，頻率最低（緩慢變化）。

通過調整不同維度的頻率範圍，Positional Encoding 能夠捕捉不同尺度上的位置信息。

1. 頻率的範圍與分佈

位置編碼的核心目的是為不同的維度提供具有不同頻率的波形，使模型能夠區分不同位置並學習序列的順序特徵。

- 在公式 $\frac{pos}{10000^{\frac{2i}{d}}}$ 中，$10000^{\frac{2i}{d}}$ 為指數增長，導致頻率跨越非常大的範圍。
- 10000 作為基數是一個足夠大的數值，可以在常見的序列長度（例如文本的句子或段落）中生成頻率適中的正弦與餘弦波形。

2. 穩定的數值尺度

- 當 i 接近 d 的上限時，$10000^{\frac{2i}{d}}$ 的值會變得非常大。 **頻率極低**
- 如果基數過小，指數增長的結果會導致數值變得過於極端，可能引發數值溢出或導致梯度不穩定。
- 10000 是經驗選擇，能有效避免數值過大或過小，並與序列長度、編碼維度等實際模型參數匹配。

Positional Encoding

資料順序

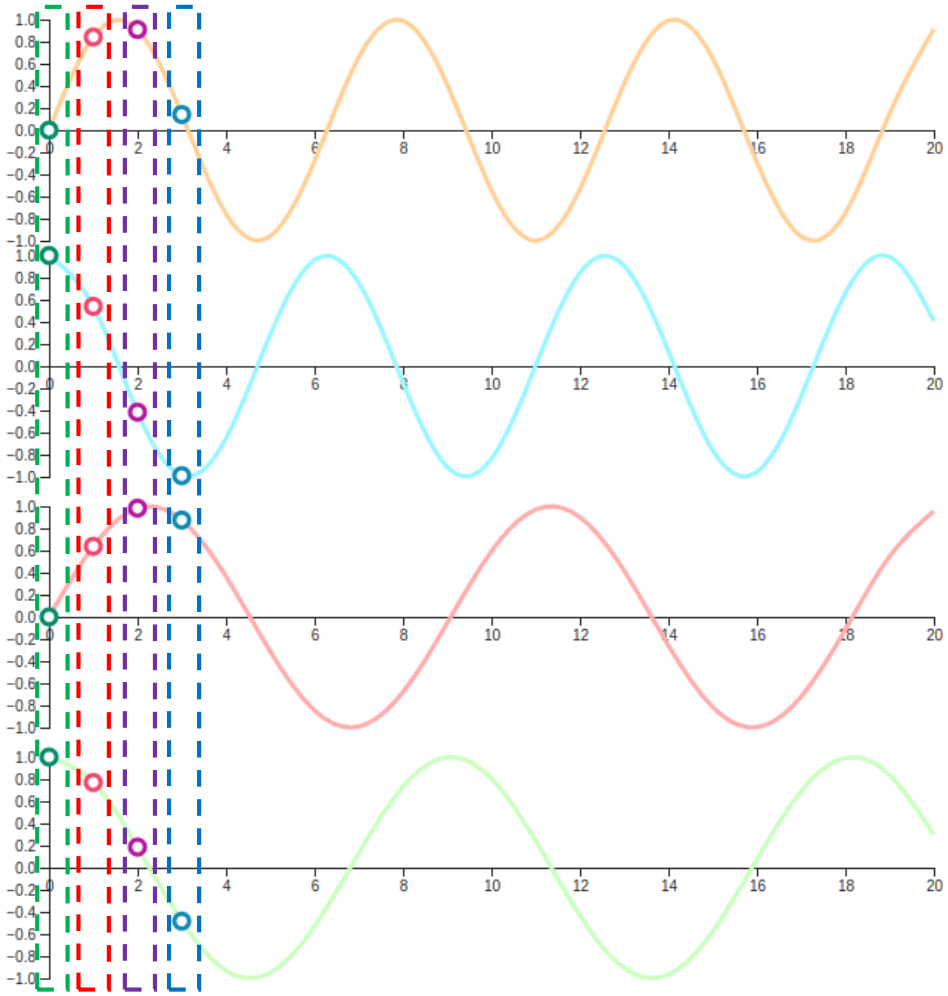
索引

i=0

i=1

i=2

i=3



pos

	p0	p1	p2	p3	
	0.000	0.841	0.909	0.141	i=0
	1.000	0.540	-0.416	-0.990	i=1
	0.000	0.638	0.983	0.875	i=2
	1.000	0.770	0.186	-0.484	i=3

d=4(資料維度)

Positional Encoding

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

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

位置編碼

Settings: d = 50

The value of each positional encoding depends on the *position* (*pos*) and *dimension* (*d*). We calculate result for every *index* (*i*) to get the whole vector.

Cosine Similarity（餘弦相似度）

Cosine Similarity 是一種衡量兩個向量之間相似度的方法，特別適用於高維空間中，廣泛用於文本處理和自然語言處理（NLP）中的向量比較，例如詞嵌入（Word Embeddings）或文檔向量。

---

1. 定義

Cosine Similarity 衡量的是兩個向量之間的夾角餘弦值，而非它們的歐幾里得距離。它的公式為：

$$\text{cosine\_similarity}(\mathbf{u}, \mathbf{v}) = \cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

其中：

- $\mathbf{u}, \mathbf{v}$：待比較的向量。

- $\mathbf{u} \cdot \mathbf{v}$：向量的點積，定義為：

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i$$

- $\|\mathbf{u}\|, \|\mathbf{v}\|$：向量的 L^2 範數（模長），定義為：

$$\|\mathbf{u}\| = \sqrt{\sum_{i=1}^{n} u_i^2}$$

2. 範圍與意義

Cosine Similarity 的值範圍為：

$$\text{cosine\_similarity}(\mathbf{u}, \mathbf{v}) \in [-1, 1]$$

- **1**：完全相似（方向相同）。
- **0**：無相似性（向量正交）。
- **-1**：完全不相似（方向相反）。

Cosine Similarity

🕐 10:13

Cosine Similarity:
Clearly Explained!!!

Residual Connection（殘差連接）

Residual Connection 是深度神經網路中一種架構設計，用於緩解模型訓練中因梯度消失或梯度爆炸而導致的性能下降問題。它首次在 ResNet（Residual Network）中被提出，並廣泛應用於 Transformer 等現代深度學習模型。

1. 定義

在殘差連接中，輸入 $\mathbf{x}$ 與經過一個或多個層（例如線性層、卷積層等）變換後的輸出 $\mathcal{F}(\mathbf{x})$ 相加：

$$\mathbf{y} = \mathcal{F}(\mathbf{x}) + \mathbf{x}$$

- $\mathbf{x}$：輸入張量（Residual Input）。
- $\mathcal{F}(\mathbf{x})$：表示需要學習的非線性映射（如神經網路層的輸出）。
- $\mathbf{y}$：殘差連接的輸出。

2. 為什麼需要 Residual Connection

1. **解決深層網路的退化問題**

   在深度模型中，增加網路層數可能導致性能下降，這被稱為網路的退化問題。殘差連接可以幫助優化過程，讓學習更深層網路變得可行。

2. **保留原始信息**

   殘差連接將輸入直接傳遞給輸出，保留了原始特徵信息，有助於梯度的穩定傳播。

3. **易於學習的恒等映射**

   如果網路層的學習無效，模型可以輕鬆退化為恒等映射（即 $\mathcal{F}(\mathbf{x}) = 0$），從而確保模型的基本性能。

3. 在 Transformer 中的應用

在 Transformer 中，Residual Connection 用於每個子層（例如 Multi-Head Attention 和 Feed-Forward Network）之間。其公式為：

$$\text{Output} = \text{LayerNorm}(\mathbf{x} + \mathcal{F}(\mathbf{x}))$$

- **步驟：**
  1. 計算子層的輸出 $\mathcal{F}(\mathbf{x})$。
  2. 將 $\mathcal{F}(\mathbf{x})$ 與原始輸入 $\mathbf{x}$ 相加。
  3. 對結果進行層歸一化（Layer Normalization）。

這樣的設計確保每個子層的輸入包含了來自前層的直接信息。

Layer Normalization（層歸一化）

Layer Normalization 是一種神經網路正規化技術，用於加速訓練和提高模型的穩定性。它通過對每個樣本的神經元激活值進行標準化，使其在訓練過程中保持一致的分佈。

1. 定義

Layer Normalization 的核心操作是對每個輸入樣本的激活值進行標準化，公式如下：

標準化公式：

$$\hat{\mathbf{h}}_i = \frac{\mathbf{h}_i - \mu}{\sigma + \epsilon}$$

輸出公式：

$$\mathbf{y}_i = \gamma \hat{\mathbf{h}}_i + \beta$$

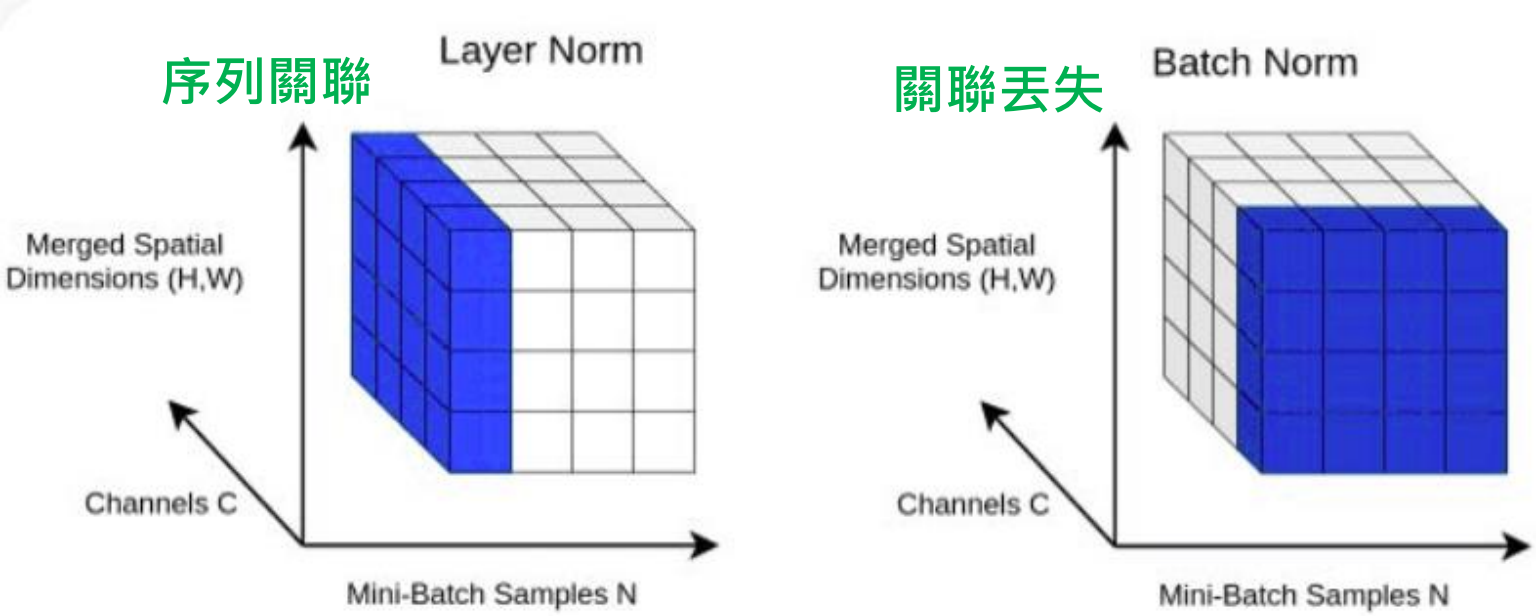
其中：

- $\mathbf{h}_i$：某一層的輸入激活值向量。
- $\mu = \frac{1}{H} \sum_{j=1}^{H} \mathbf{h}_j$：該層神經元激活值的均值。
- $\sigma = \sqrt{\frac{1}{H} \sum_{j=1}^{H} (\mathbf{h}_j - \mu)^2}$：該層神經元激活值的標準差。
- ϵ：一個小正數，防止分母為零。
- γ 和 β：可學參數，分別用於縮放和平移標準化後的值。
- H：該層中神經元的數量。

2. 與其他正規化方法的區別

特性	Layer Normalization	Batch Normalization
歸一化範圍	每個樣本內的所有神經元	整個批次中的每個神經元
計算依賴性	僅依賴單個樣本	依賴於整個批次（需要計算批次均值和方差）
使用場景	更適合序列建模、RNN、Transformer 等場景	更適合 CNN 等需要大批次訓練的場景
批量大小影響	與批量大小無關	批量大小過小時性能可能下降

場景	原因
序列建模（如 RNN）	Batch Normalization 對時間序列的依賴可能導致信息丟失，而 Layer Normalization 能保留時間序列關係。
小批次訓練	Layer Normalization 不依賴於批次均值和方差，適合小批次甚至單樣本場景。
Transformer 架構	每層輸出經 Layer Normalization 正規化，有效提升多層深度模型的穩定性和性能。

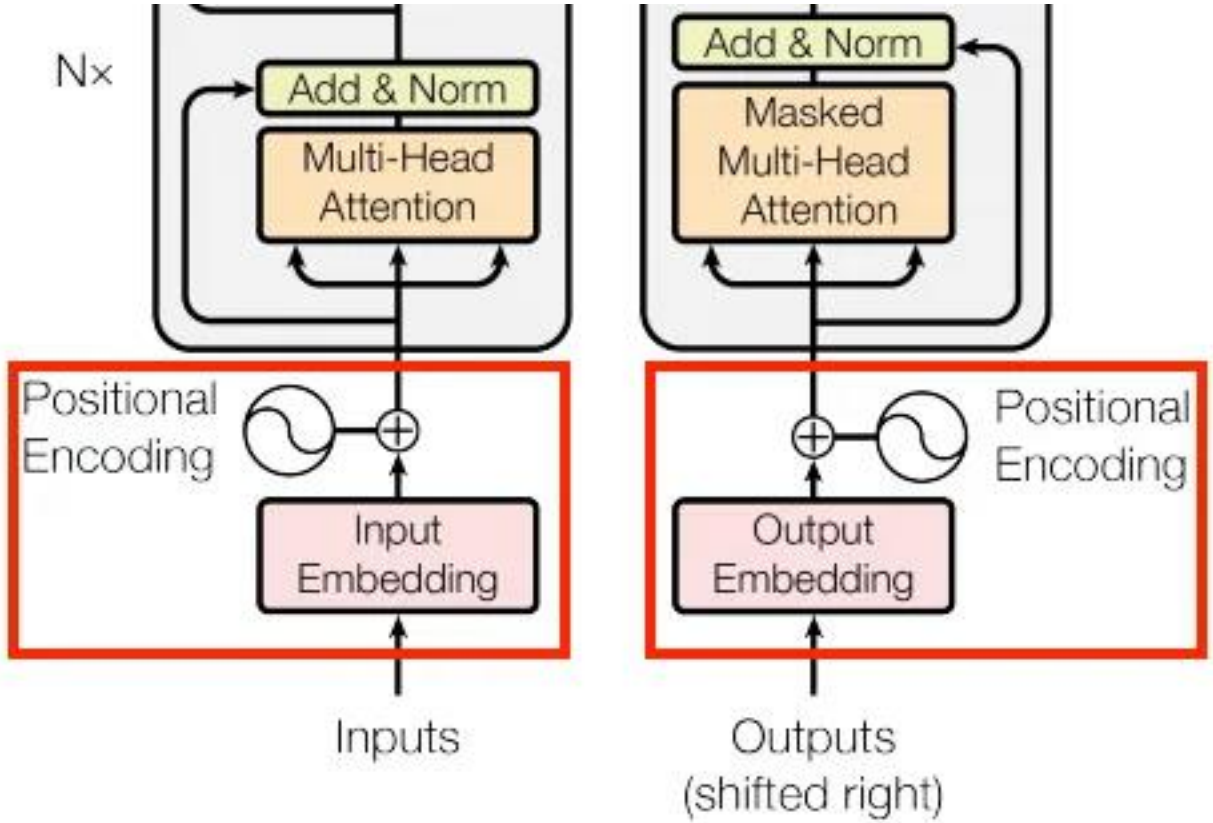


Transformer Neural Network

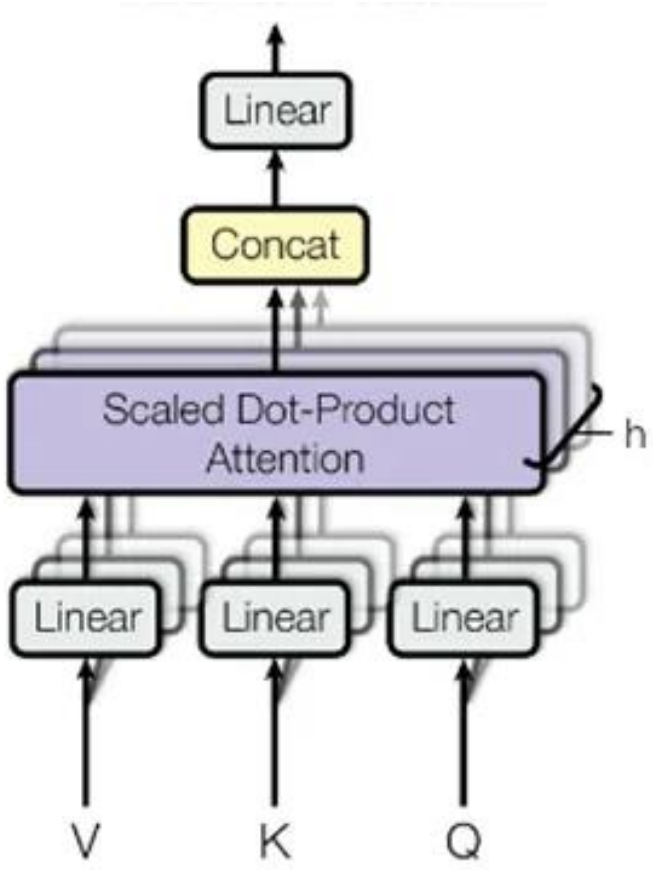
🕐 36:14

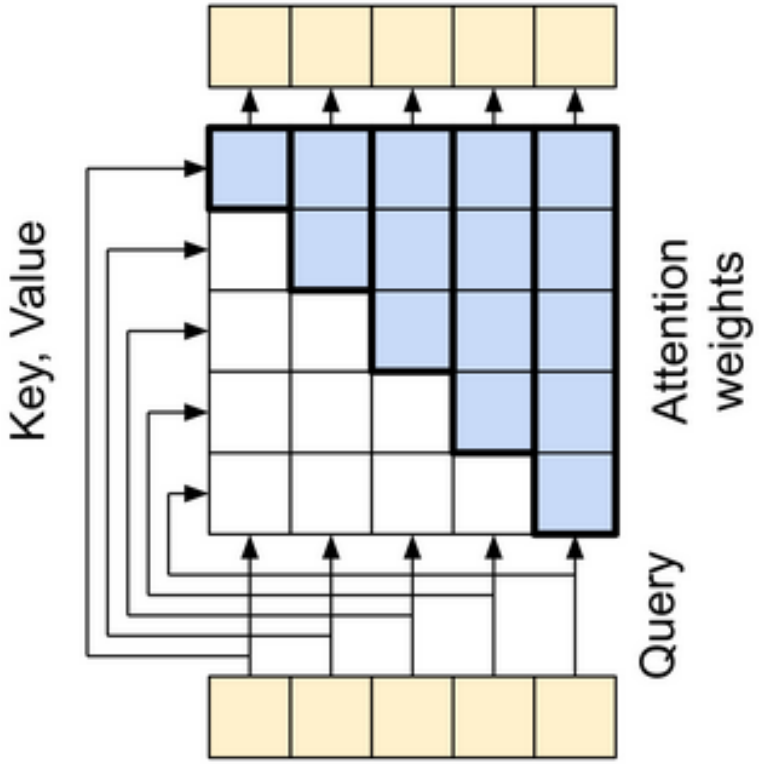
Transformer Neural Networks: Clearly Explained!!!

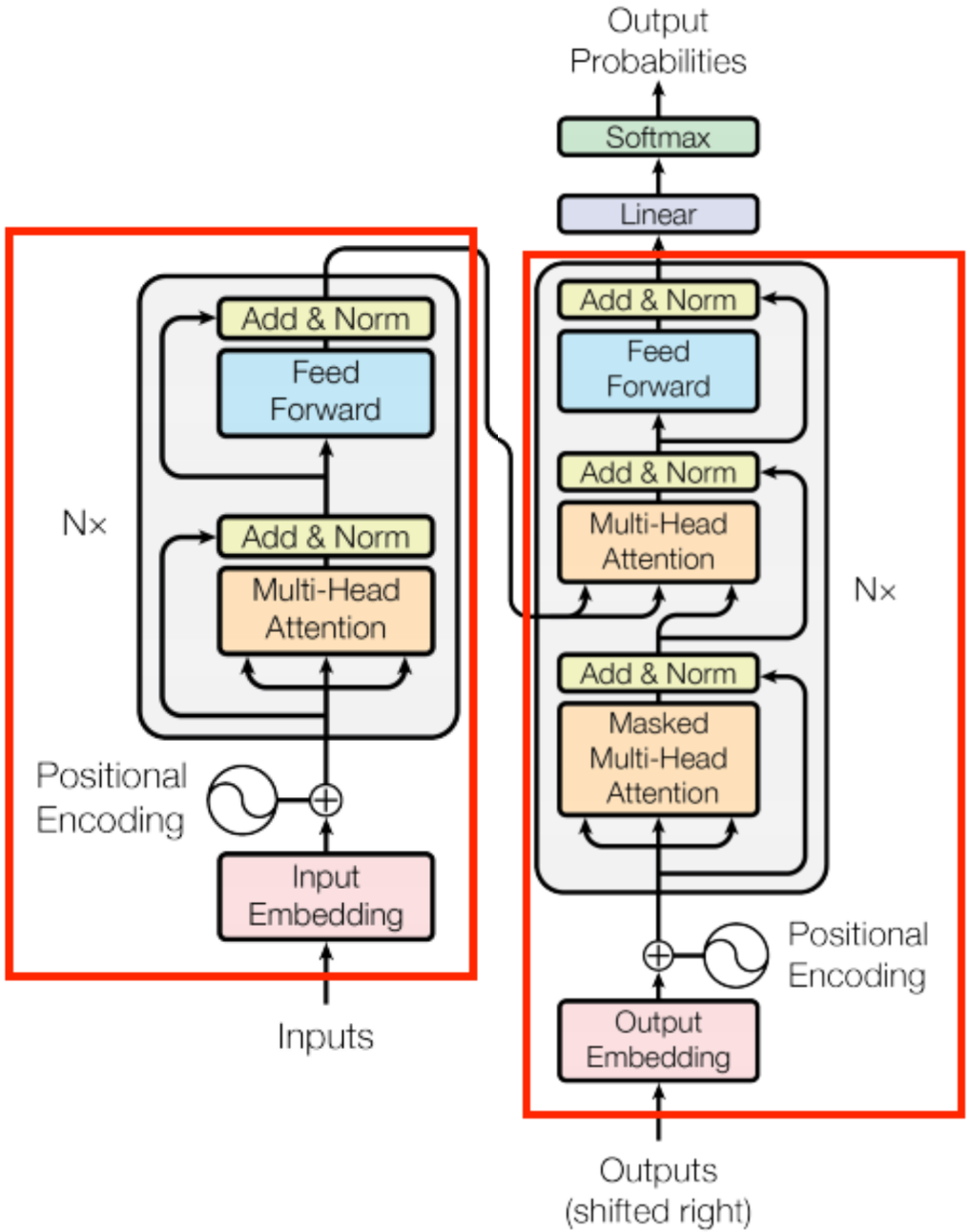
Word Embedding and Positional Encoding

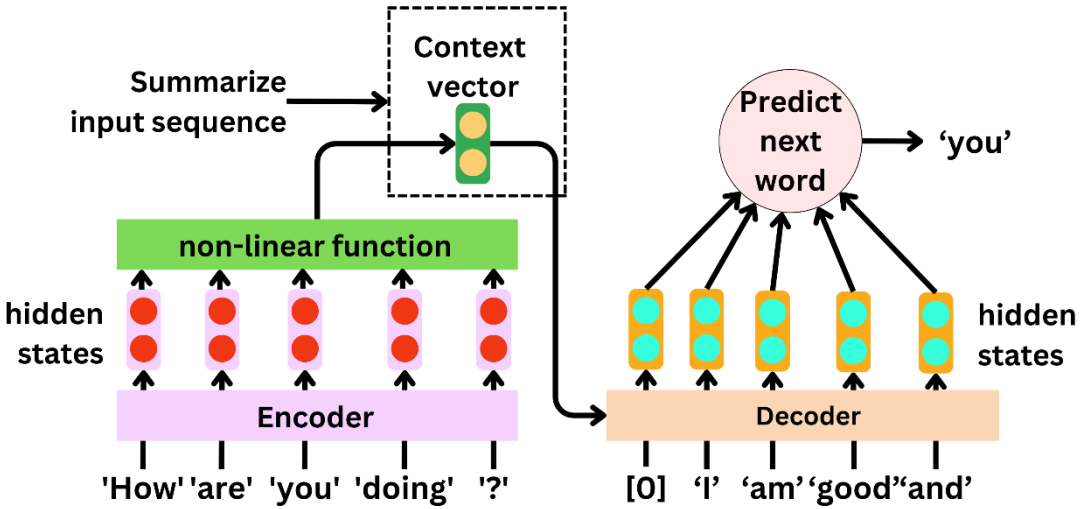


Self-Attention






Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
Outputs (shifted right)


Summarize input sequence
Context vector
Predict next word
'you'
non-linear function
hidden states
hidden states
Encoder
Decoder
'How' 'are' 'you' 'doing' '?'
[0] 'I' 'am' 'good' 'and'

Decoder-only Transformer 是一種基於 Transformer 架構的模型，專注於 **序列生成任務**，如自然語言生成（NLG）、語音生成、程式碼生成等。與標準的編碼器-解碼器（Encoder-Decoder）架構不同，解碼器專用 Transformer 僅包含 Transformer 的解碼器部分。這種設計的核心特點是透過自回歸（auto-regressive）方法生成序列。

架構簡介

1. 單向處理（Unidirectional Processing）

- **解碼器專用 Transformer** 僅處理輸入序列，並根據已處理的內容逐步生成新輸出。
- 模型中的注意力機制採用 **遮罩自注意力（Masked Self-Attention）**，確保每個位置只能關注當前位置及其之前的內容。

2. 模型結構

每個 Transformer 解碼器層包括以下部分：

1. **遮罩自注意力層（Masked Self-Attention Layer）：**
    - 計算序列中每個位置的表示，僅基於當前和之前位置的資訊。
    - 防止模型在生成時使用未來的資訊。
2. **前饋神經網絡（Feed-Forward Neural Network, FFN）：**
    - 提升序列表示的非線性轉換能力。
3. **殘差連接與層正則化（Residual Connections and Layer Normalization）：**
    - 加速收斂，穩定訓練過程。

運作機制

解碼器專用 Transformer 通過自回歸生成輸出：

1. **輸入處理**：
    - 將輸入序列嵌入（embedding）為固定維度向量，並加入位置編碼（Positional Encoding），提供序列順序資訊。
2. **遮罩注意力**：
    - 將未來位置遮罩，使模型在每個位置只能看到過去及當前資訊。
3. **逐步生成**：
    - 輸出序列的每個位置依賴於之前的位置逐步生成：
        - 第一步生成第一個 token。
        - 使用第一個 token 預測第二個 token。
        - 以此類推。

數學表達

假設輸入序列為 $x = [x_1, x_2, \ldots, x_T]$，模型目標是生成對應的輸出序列 $y = [y_1, y_2, \ldots, y_T]$。

1. **遮罩自注意力計算**：

$$A_{\text{masked}} = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)$$

- Q, K, V 為查詢、鍵、和值向量。
- 遮罩矩陣 M 確保 $A_{ij} = 0$ 當 $j > i$。

$$M = \begin{cases} 0 & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases}$$

2. **輸出生成**： 模型生成序列的概率分佈：

$$P(y_1, \ldots, y_T) = \prod_{t=1}^{T} P(y_t | y_1, \ldots, y_{t-1})$$

這表示每個位置的輸出依賴於過去的位置。

Decoder-only Transformer

36:44

Matrix Algebra

30:00

Essential Matrix Algebra for Neural Networks, Clearly Explained!!!

Matrix Algebra

23:42

The Matrix Math Behind Transformer Neural Networks, One Step at a Time!!!

GPT（Generative Pre-trained Transformer）本身就是一種 **Decoder-Only Transformer**

比較維度	Decoder-Only Transformer	GPT
是否具體化應用	通用框架	具體化實現，專注生成任務
訓練框架	不限（僅指架構本身）	預訓練（Pre-training）+ 微調（Fine-tuning）
模型版本演進	無特定版本或規範	GPT-1, GPT-2, GPT-3, GPT-4 等版本
數據規模與資源	依需求調整	通常基於超大規模語料（特別是 GPT-3/4）
應用能力	理論上可靈活應用	展現出強大的泛用性與生成能力

- **因果遮罩（Causal Masking）**：
  二者都利用因果遮罩來控制注意力機制，只能考慮當前及之前的序列內容，從而進行自回歸生成。
- **適用任務**：
  主要用於生成相關的任務，如文本生成、補全、摘要、對話等。

ChatGPT 是基於 GPT 的專門版本，經過微調以優化為對話代理，旨在模擬人類對話的流暢性和上下文理解能力。

特別針對用戶與 AI 的互動進行優化，重點在於提供更自然且上下文相關的回答。

比較維度	GPT	ChatGPT
主要用途	通用生成模型，用於多種 NLP 任務	對話優化模型，專注於人機互動
訓練目標	預測下一個詞	提供自然、連貫的對話體驗
訓練方式	預訓練（無監督學習）	預訓練 + 微調（如 RLHF）
上下文處理	主要基於單一輸入進行生成	支援多輪對話與上下文記憶
安全性與控制	安全性控制相對較弱	強調過濾不當內容與優化用戶體驗

LLM 是大型語言模型（Large Language Model）的縮寫，它是基於深度學習技術構建的一種人工智慧模型，專門用於自然語言處理任務。以下是 LLM 的一些主要特點和應用：

特點

1. **基於大量數據訓練**

   LLM 通常由包含數十億到數千億參數的神經網路構成，並利用大規模的文本數據進行訓練。

2. **多樣化任務能力**

   它可以應用於多種自然語言處理任務，如文本生成、翻譯、摘要、問題回答、對話等。

3. **上下文理解**

   LLM 能夠分析和生成與上下文相關的高品質文本。

4. **預訓練與微調**

   通過預訓練（Pre-training）學習一般語言知識，並通過微調（Fine-tuning）適應特定任務。

應用

1. **文本生成**

   撰寫文章、寫作輔助（如 OpenAI 的 GPT 系列）。

2. **機器翻譯**

   自然語言的多語種翻譯。

3. **對話系統**

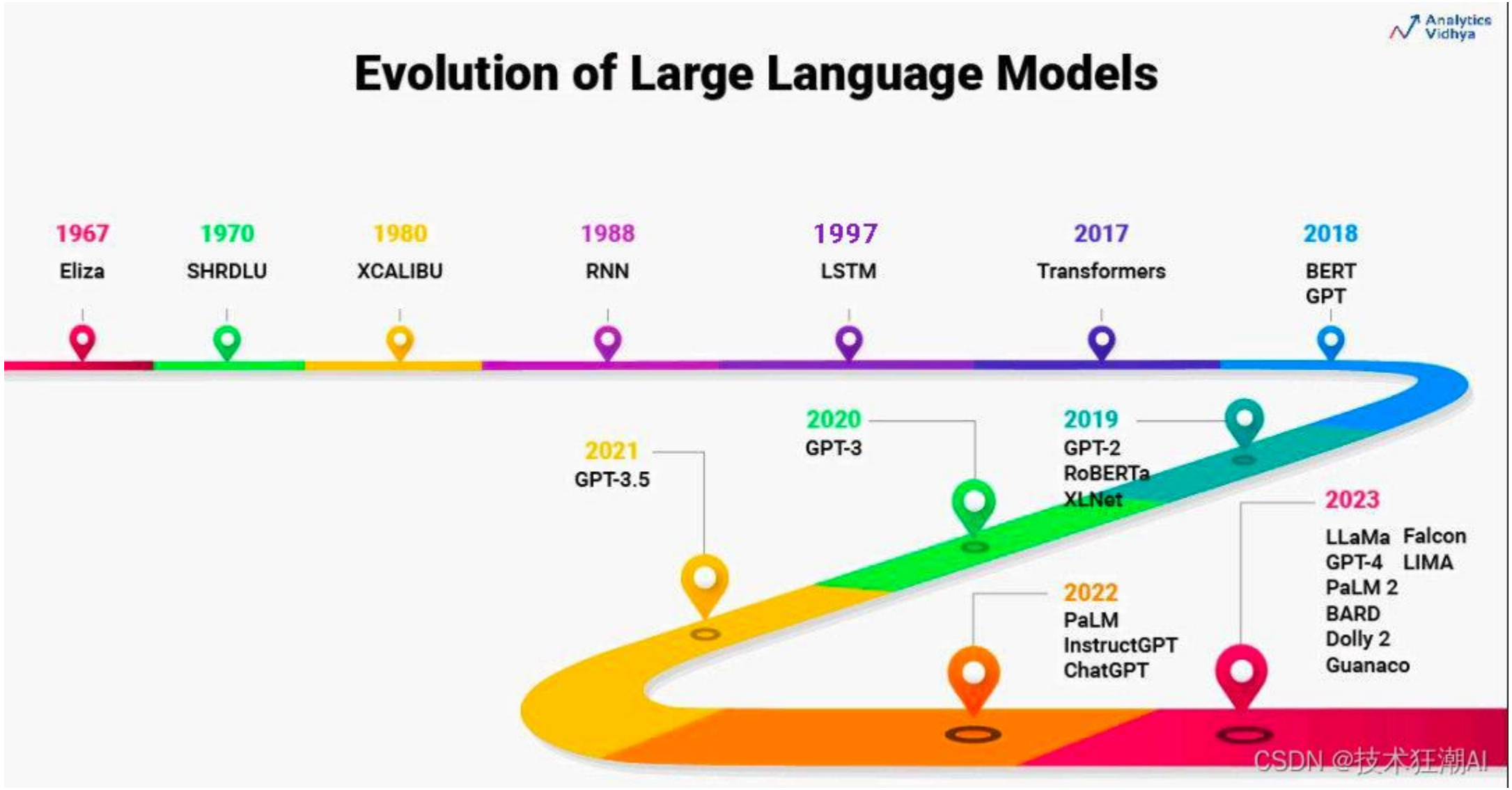
   智慧客服、聊天機器人（如 ChatGPT）。

4. **搜索引擎與推薦系統**

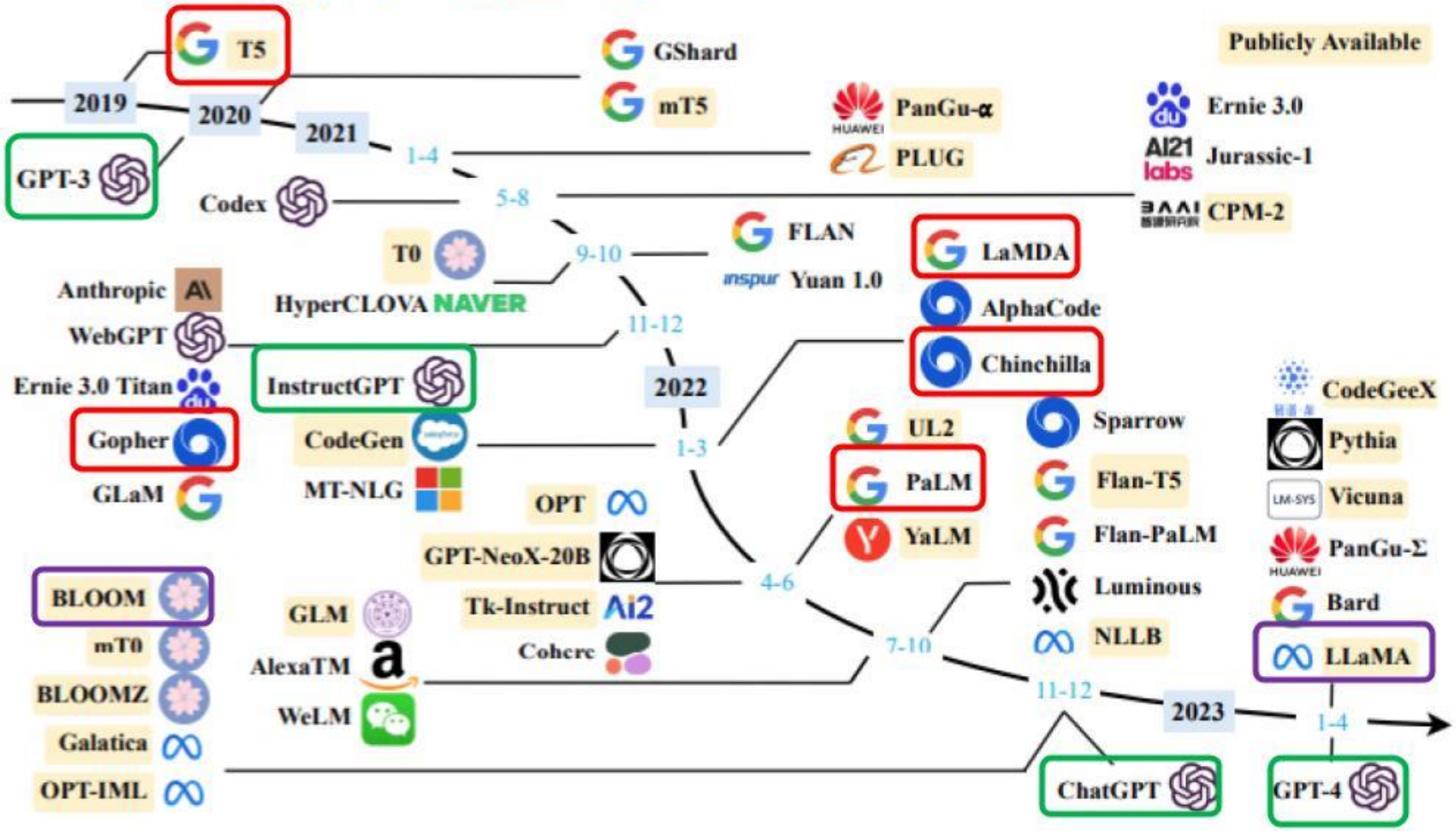
   提升資訊檢索和內容推薦的精確度。

5. **教育與學術輔助**

   提供解答、教學輔助及研究建議。


Analytics Vidhya
Evolution of Large Language Models
1967
Eliza
1970
SHRDLU
1980
XCALIBU
1988
RNN
1997
LSTM
2017
Transformers
2018
BERT
GPT
2019
GPT-2
RoBERTa
XLNet
2020
GPT-3
2021
GPT-3.5
2022
PaLM
InstructGPT
ChatGPT
2023
LLaMa Falcon
GPT-4 LIMA
PaLM 2
BARD
Dolly 2
Guanaco
CSDN @技术狂潮AI

LLMs 和它們的產地



圖片來源 : https://arxiv.org/abs/2303.18223

開放式 LLM

1. **GPT 系列**（OpenAI）
    - **版本**：GPT-3、GPT-4
    - **特點**：專注於通用自然語言生成和對話，支援多語言，具備優秀的上下文理解能力。
    - **應用**：ChatGPT、Codex（程式碼生成）、DALL·E 整合。
2. **LLaMA 系列**（Meta）
    - **版本**：LLaMA、LLaMA 2
    - **特點**：輕量化設計，開源版本可用於研究與開發，支援多種 NLP 任務。
    - **應用**：社群研究及應用原型開發。

3. **BERT 系列**（Google）
    - **版本**：BERT、RoBERTa（Meta 優化版）、ALBERT（輕量版）
    - **特點**：專注於句子分類、情感分析和問答系統，較適合 NLP 微調。
    - **應用**：Google Search、資訊提取。
4. **OPT 系列**（Meta）
    - **版本**：OPT-175B
    - **特點**：開源大模型，注重與 GPT-3 相似的設計，但降低了運行成本。
    - **應用**：學術與開源社群實驗。

專業用途 LLM

1. **PaLM 系列**（Google）
    - **版本**：PaLM 2
    - **特點**：專為對話生成和多模態處理設計，支援多語言和高級推理。
    - **應用**：Google Bard、醫學專業應用（Med-PaLM）。
2. **Claude 系列**（Anthropic）
    - **版本**：Claude 1、Claude 2
    - **特點**：注重 AI 安全性和可控性，設計更符合倫理需求。
    - **應用**：企業客服、自動化助手。

3. **Bloom**（BigScience）
    - **特點**：完全開源，支援 46 種語言和 13 種程式語言。
    - **應用**：學術研究與多語言應用。
4. **Grok**（Samsung）
    - **特點**：專為企業內部應用開發，結合內部生態系統的資料。