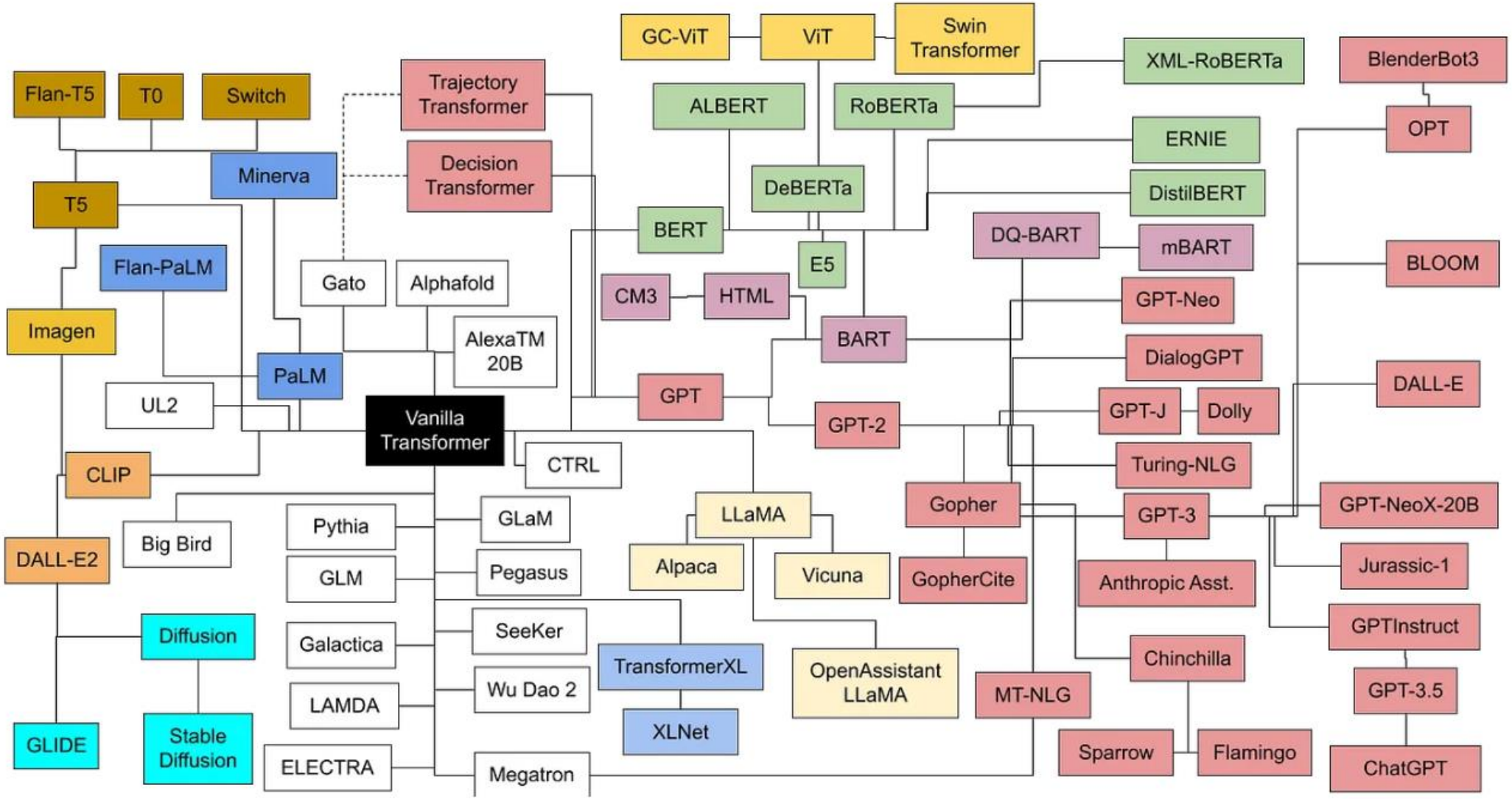
自然語言處理
Advance Pretrain Language Model

Instructor: 馬豪尚

Transformer Family Tree



Transformer 家族重要成員分類

- Encoder Pretraining
  - 這些模型也被稱為雙向或自動編碼器，它們僅在預訓練階段使用編碼器
  - 這通常是通過遮蔽輸入句子中的token，然後訓練模型來重建這些token
  - 這一系列的模型最適合於需要理解完整句子或段落的任務，例如文本分類、蘊含判斷，以及摘錄式問答
  - 例如: Bert家族
- Decoder Pretraining
  - 解碼器模型在預訓練期間只用到解碼器部分，也被稱作自回歸語言模型，因為訓練它們的目的是要能根據先前的token序列來預測接下來的token
  - 自注意力層只能接觸到句子中某個給定token之前的標記。
  - 這種模型最適合用在涉及文本生成的任務上
  - 例如: GPT家族

Transformer 家族重要成員分類

- Transformer (Encoder-Decoder) Pretraining
  - 同時包含訓練編碼器-解碼器模型，又被稱作序列對序列模型(Seq2Seq)
  - 編碼器的自注意力層能觀看所有的輸入token，而解碼器的自注意力層只能看到給定token之前的token。
  - 編碼器-解碼器模型非常適合於需要基於給定輸入生成新句子的任務，比如摘要、翻譯或生成式問答
  - 例如: T5 、TransformerXL、XLNet
- 特殊的類型
  - BART使用BERT for encoder結合GPT for decoder

常見的語言模型預訓練任務

- 語言模型 (LM)
  - 預測下一個token或者同時預測前一個和下一個token
- 因果語言模型(Causality-masked LM)
  - 按時間順序（通常是從左到右）自回歸地預測文本序列，類似於單向 LM。
- 前綴語言模型(Prefix LM)
  - 在這個任務中，一個獨立的'前綴'部分會從主序列中分離出來
  - 在前綴內部，任何token都能注意到其他的token（非因果的）。
  - 在前綴之外，解碼按自回歸方式進行。
- 遮蔽語言模型(Mask LM)
  - 從輸入句子中遮蔽一些標記，然後訓練模型使用周圍上下文來預測這些遮蔽的標記。

常見的語言模型預訓練任務

- 排列語言模型(Permuted LM)
  - 與 LM 相同，但是對輸入序列的隨機排列進行操作。從所有可能的排列中隨機抽取一個排列。然後選擇一些標記作為目標，訓練模型預測這些目標。
- 去噪自編碼器(Denoising Auto Encoder)
  - 接受部分受損的輸入，並旨在恢復原始的、未失真的輸入。受損輸入的例子包括從輸入中隨機抽取標記並將它們替換為“[MASK]”元素，從輸入中隨機刪除標記，或者將句子以隨機順序打亂。
- 取代的token檢測(Replaced Token Detection)
  - 使用一個“生成器”模型，隨機替換文本中的某些標記。而“鑑別器”則負責預測一個標記是來自原始文本，還是生成器模型。
- 下一句預測(Next Sentence Prediction)
  - 訓練模型區分兩個輸入句子是否為訓練語料庫中的連續片段。

Transformer XLNet

- 自迴歸語言模型（Autoregressive LM）
  - 任務為根據上文內容預測下一個token，預訓練任務為單向的語言模型任務
- 自編碼語言模型（Autoencoder LM）
  - 如BERT等，根據上下文單字來預測被【MASK】掉的token
- XLNet: Transformer-XL Net
  - XLNet的想法就是要使用AR的方式來預測單詞，又要能在不使用<Mask> token的前提下學習到上下文的資訊
  - 解決了BERT中存在的「預訓練-微調」不一致的問題

排列語言模型

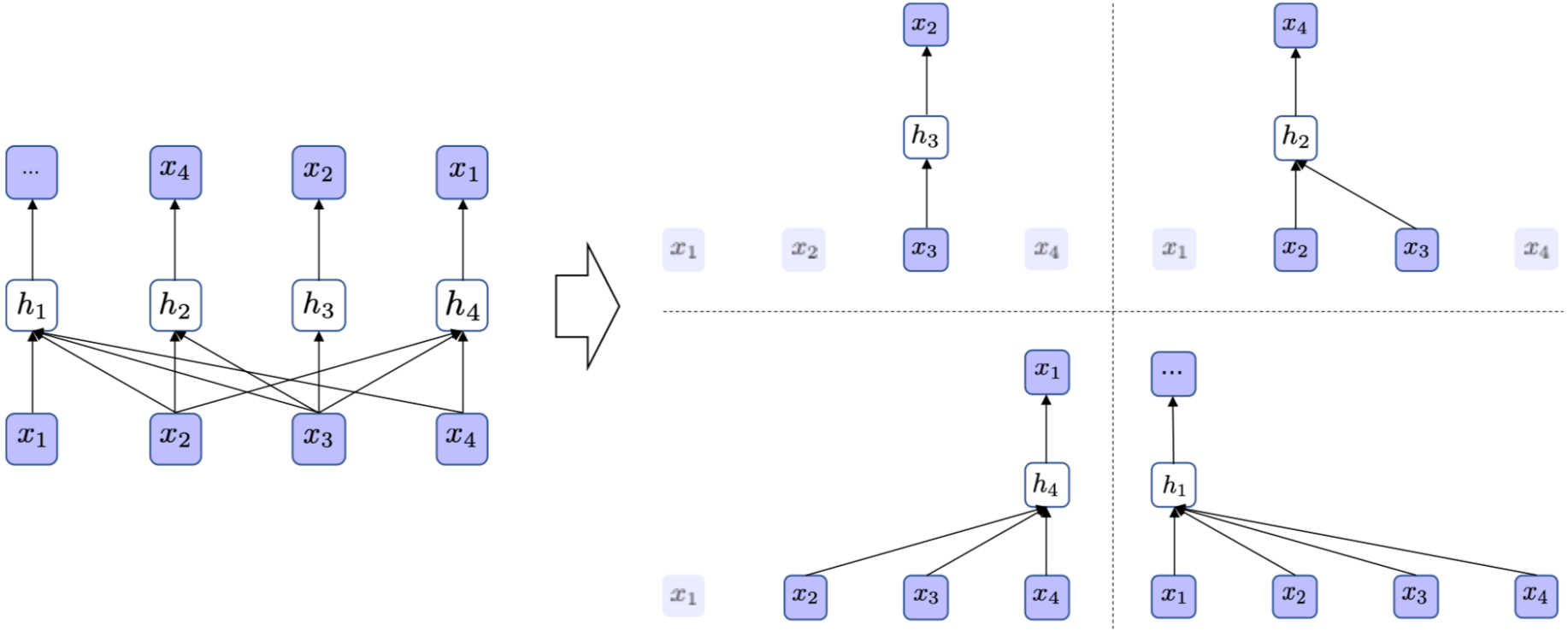
- 假設我們要預測第3個token。 正常的語言模型序列是1 -> 2 -> 3 -> 4，也就是只能看到token1和token2的訊息，不能看到token4的訊息，這是自回歸語言模型的缺點。
- 為了解決這個問題，我們將所有token進行排列組合，排列方式有4！ =24種
- 從所有可能的排列當中均勻採樣一種排列順序（factorization order）去最大化對數似然函數

$$\mathbb{E}_{z\sim\mathbb{Z}_N}[\log P(x|z)] = \mathbb{E}_{z\sim\mathbb{Z}_N}\left[\sum_{i=1}^{N} P(x_{z_i}|x_{1:i-1}, z_i)\right]$$

XLNet: Transformer-XL Net

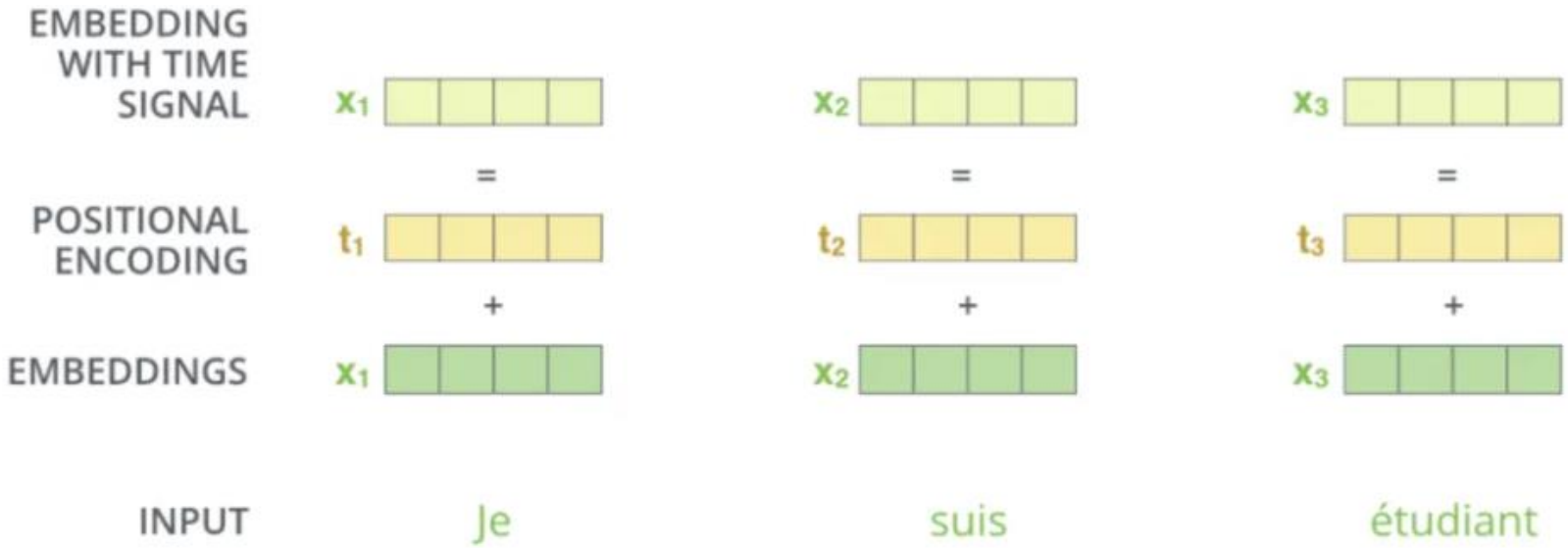
- XLNet: Transformer-XL Net
  - 改變句子順序：3→2→4→1

$$P(x) = P(x_3)P(x_2|x_3)P(x_4|x_3x_2)P(x_1|x_3x_2x_4)$$



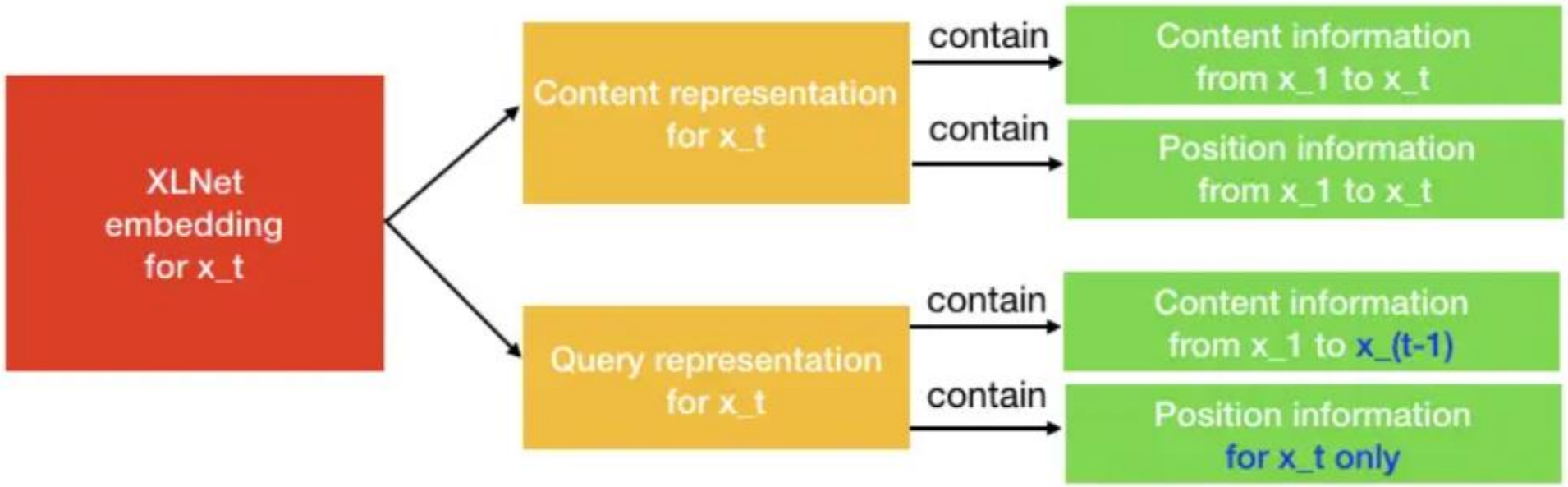
排列組合造成的問題

- 排列語言模型學習的目標函數，即以t-1個tokens為上下文，預測第t個token
- 當預測第t個token時，它應該要能看到第t個token的位置資訊而不能看到這個token的內容資訊
  - Transformer已經將輸入和position編碼合併無法單獨拆開



XLNet: Transformer-XL Net

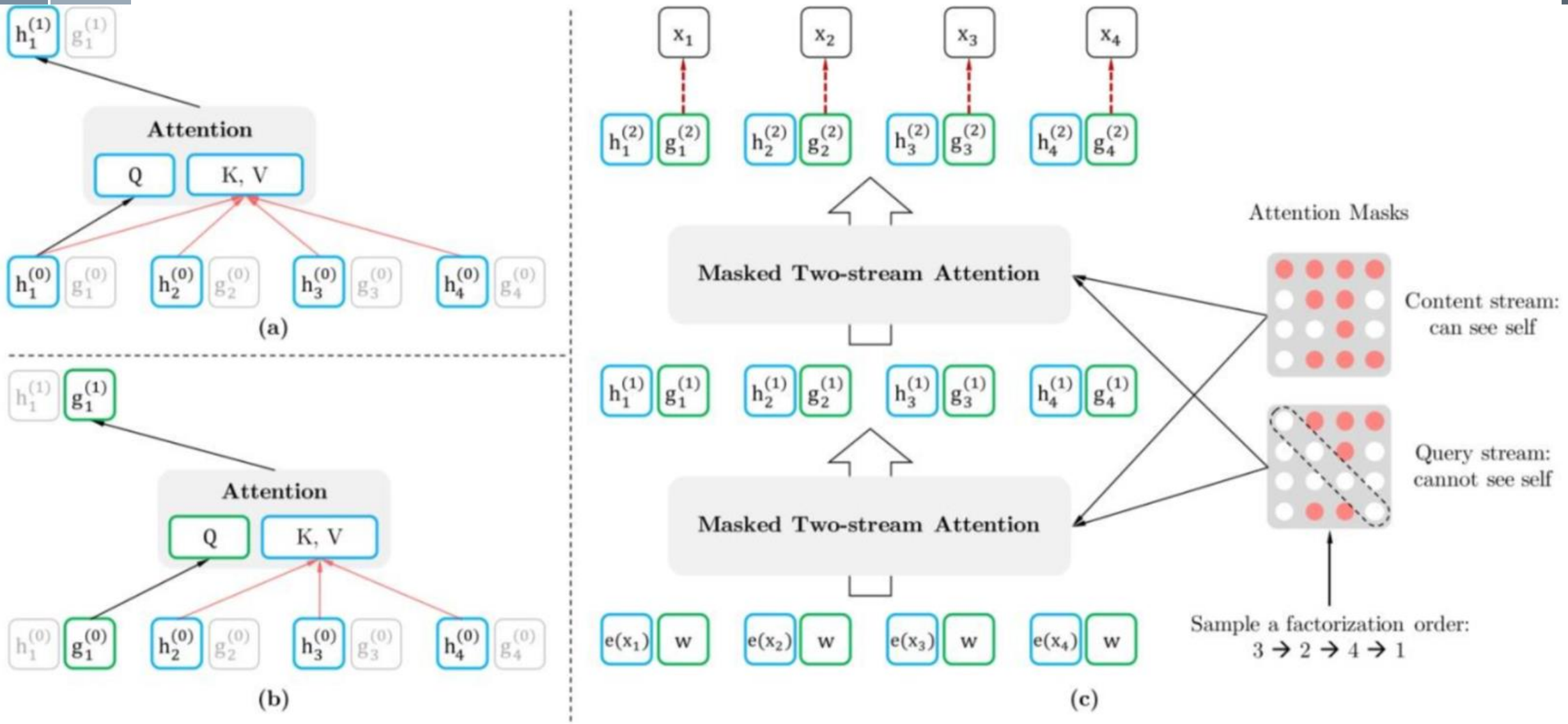
- 雙流自注意力機制
  - content stream attention，它是Transformer中的標準自注意力，負責學習上下文
  - query stream attention，XLNet用它來取代BERT中的[MASK] token



雙流自注意力機制 vs 自注意力機制

- 想用上下文單字x_1和x_2的知識來預測x_3
- BERT使用[MASK]來表示x_3 token ，[MASK]只是一個替代符號
  - x_1和x_2的嵌入包含位置資訊，幫助模型「知道」[MASK]是 x_3
- XLNet的一個token x_3將分別扮演兩種角色
  - 當它被用作內容來預測其他token時，我們可以使用內容表示(透過內容流注意力來學習)來表示x_3
  - 如果我們想要預測x_3，我們應該只知道它的位置而不是它的內容。

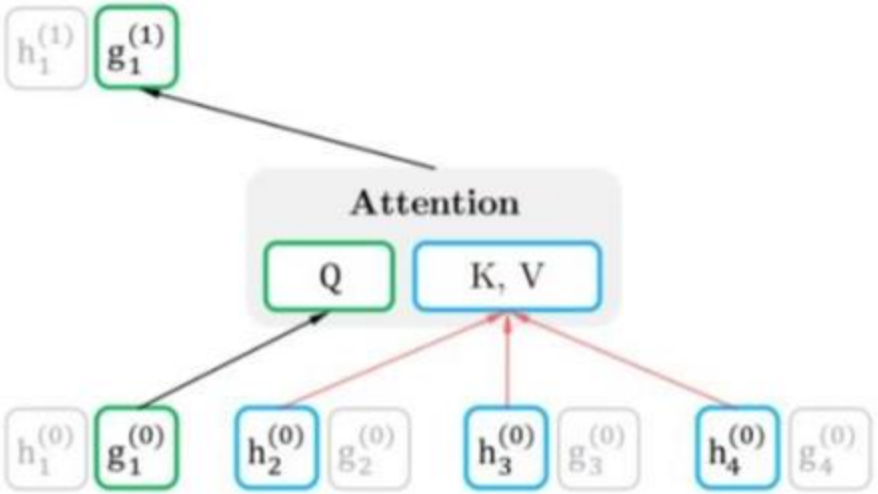
雙流自注意力機制


Attention
Q
K, V
(a)
Attention
Q
K, V
(b)
Masked Two-stream Attention
Masked Two-stream Attention
Attention Masks
Content stream:
can see self
Query stream:
cannot see self
Sample a factorization order:
3 → 2 → 4 → 1
(c)

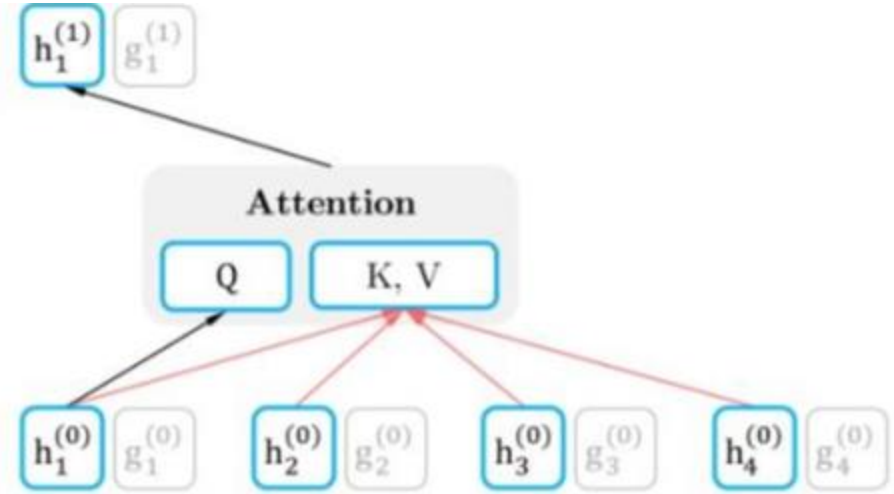
雙流自注意力機制

$$g_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = g_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta), \quad \text{(query stream: use } z_t \text{ but cannot see } x_{z_t})$$

$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}).$$



- 在預測x1的content representation時， 我們應該要參考所有4個token內容資訊
- KV = [h1, h2, h3, h4]和Q = h1

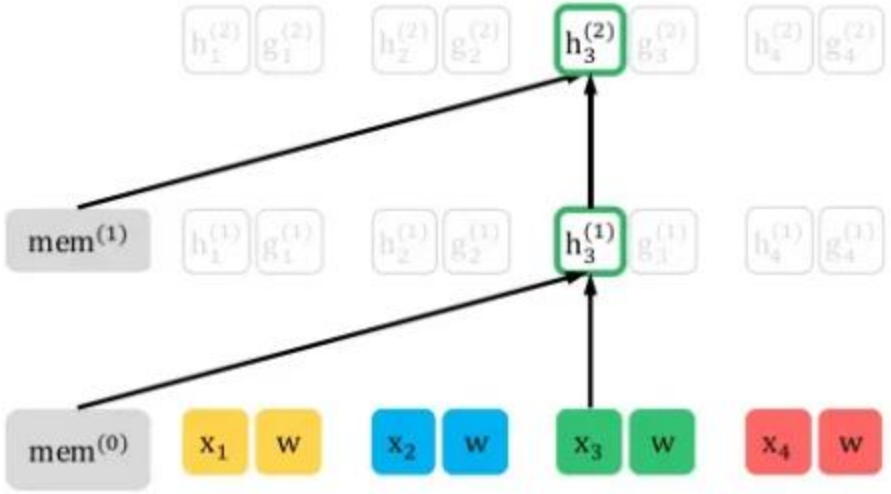


- 在預測x1的query representation 時，我們不能看到x1本身的content representation
- KV = [h2, h3, h4]， Q = g1。

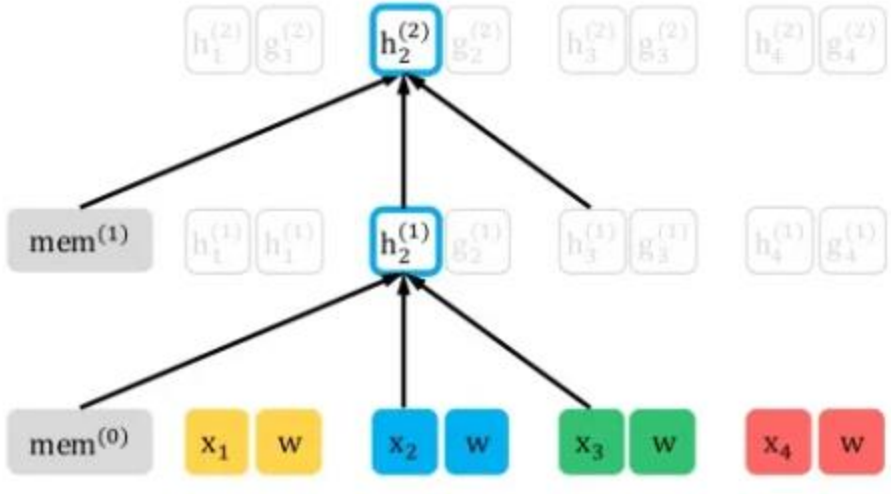
雙流自注意力機制

- h和g被初始化為e(x_i)和w
- 句子的順序(x_3->x_2->x_4->x_1)決定content stream和query stream之後，雙流注意力將輸出第一層輸出$h^{(1)}$和 $g^{(1)}$然後計算第二層
- 在content stream mask中
  - 第一行有4個紅點，代表第一個token (x_1)可以看到所有其他tokens，包括它自己。 第二行有兩個紅點
  - 第二行只有2個紅點，因為token(x_2)只能看到兩個token(x_3->x_2)

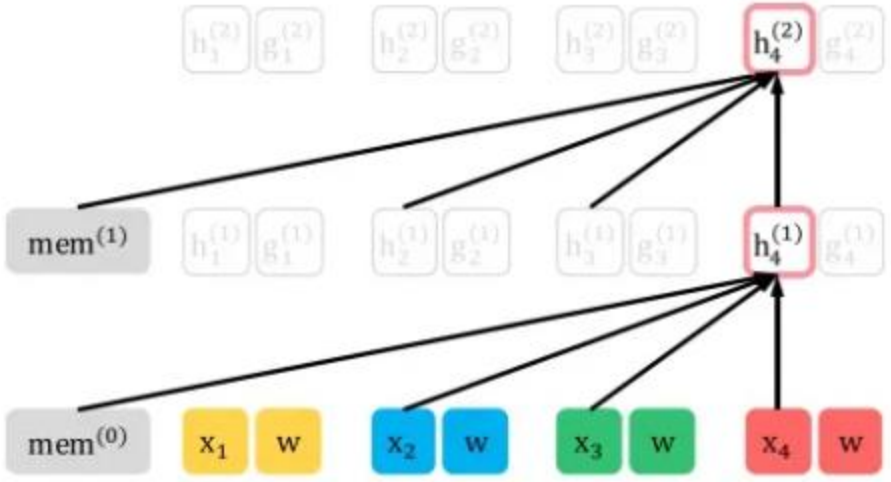
Content Stream



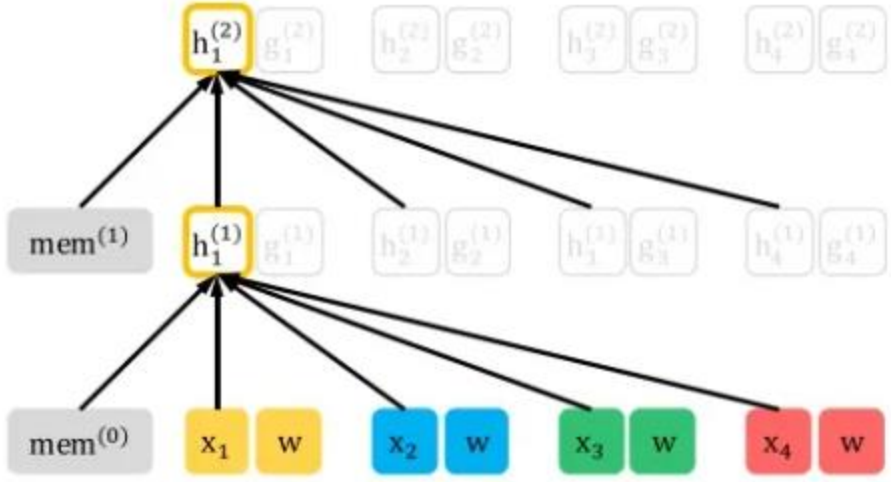
Position-3 View



Position-2 View

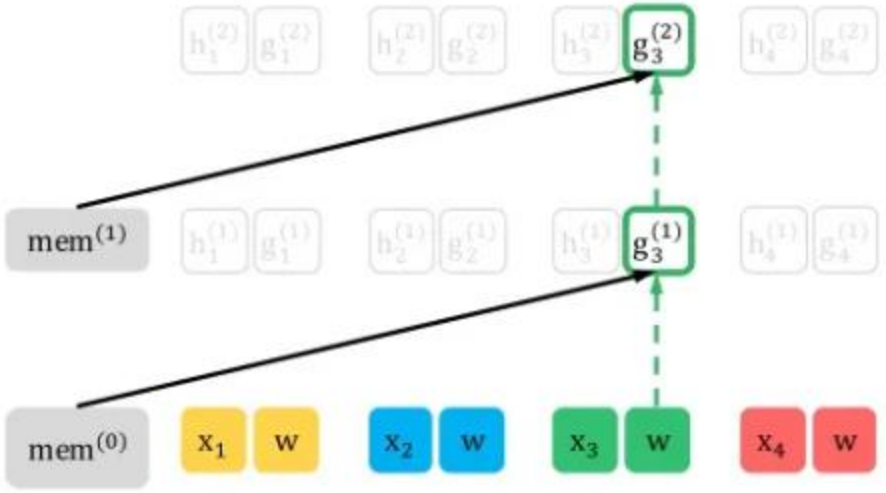


Position-4 View

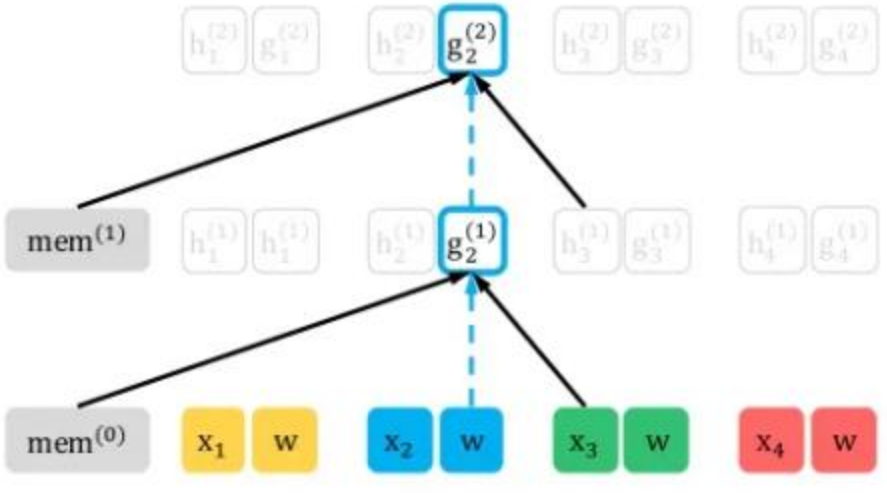


Position-1 View

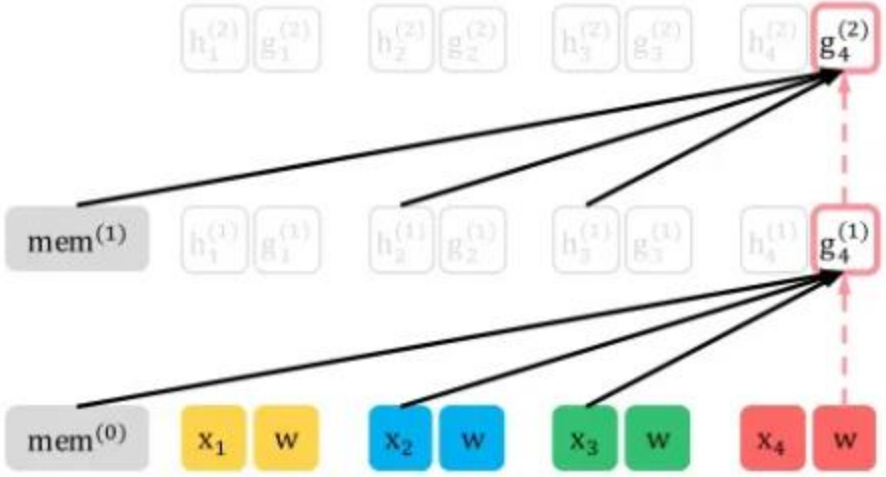
Query Stream



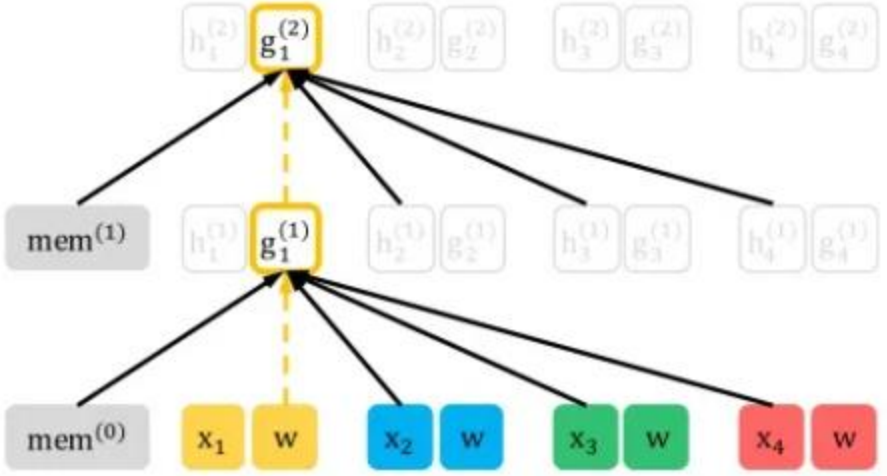
Position-3 View



Position-2 View



Position-4 View



Position-1 View