



Vanishing Gradient, Gated RNN, Seq2Seq and Transformer

報告人: 蘇佳益

指導老師: 陳聰毅

國立高雄科技大學建功校區電子工程系

<https://github.com/chaiyisu/Natural-Language-Processing>

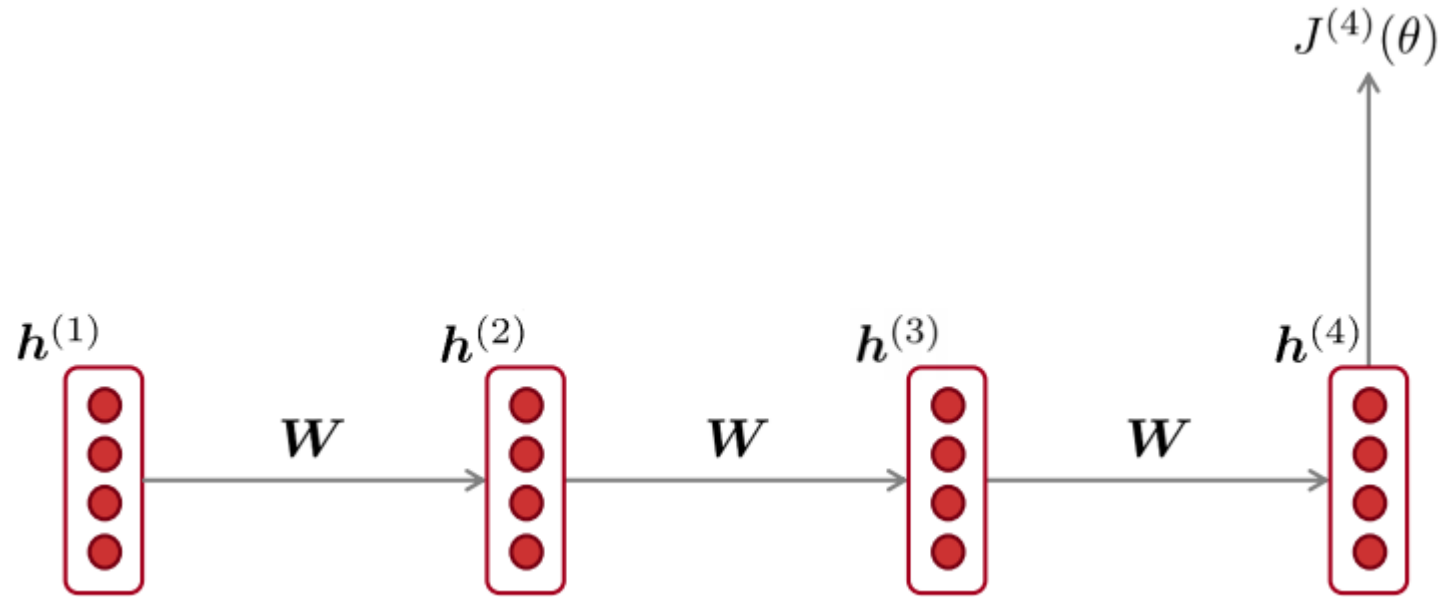
Agenda

- Vanishing Gradient
- Problems of Vanishing / Exploding Gradients
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- LSTM & GRU Comparison
- Neural Machine Translation (NMT)
- Attention
- Transformer



Vanishing Gradient

Vanishing Gradient Intuition



- $$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \boxed{\frac{\partial h^{(2)}}{\partial h^{(1)}}} * \boxed{\frac{\partial h^{(3)}}{\partial h^{(2)}}} * \boxed{\frac{\partial h^{(4)}}{\partial h^{(3)}}} * \boxed{\frac{\partial J^{(4)}}{\partial h^{(4)}}}$$

What if these values are small?

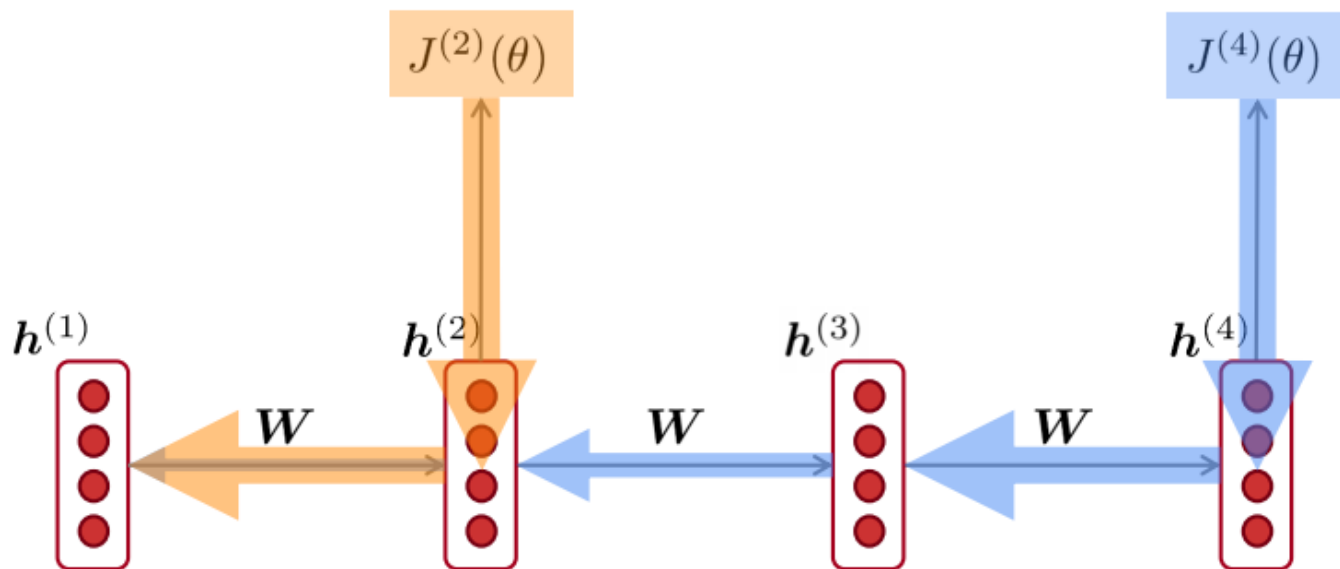
Idea: Vanishing Gradient

- 假設我們有以下三個值： $0.1, 0.01, 0.01$ ，並且將它相乘
- $0.1 * 0.01 * 0.01 = 0.00001$
- 因此：當我們將非常小的梯度相乘後，梯度將會越來越小



Problems of Vanishing / Exploding Gradients

梯度消失的問題



- 越接近輸出的梯度訊號將會越大
- 離輸出越遠的梯度訊號將會越小
- 模型只會考慮與輸出相近的資訊

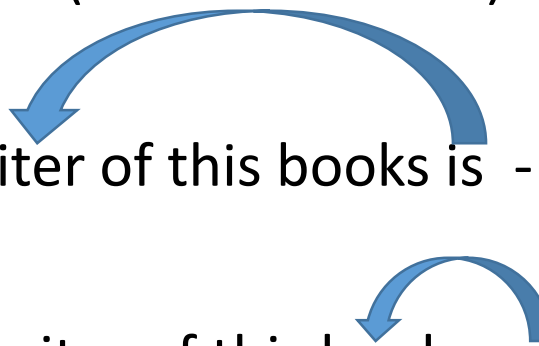
梯度消失的問題

- 當梯度消失發生在時間 t 與 $t+n$ 之間，我們很難知道以下資訊：
 - 時間 t 與 $t+n$ 的依賴是否真的不存在
 - 時間 t 與 $t+n$ 的依賴是否存在於其他不相關的參數

Vanishing Gradients : Effects on RNNLM

- **LM task:** 當她要列印票券的時候，她發現列表機沒有墨水夾。她去文具店買了非常貴的墨水夾。在她把墨水夾裝進去之後，她終於印出她的_____。
- 為了要正確的預測輸出為“票券”，模型需要學習到第一個“票券”與最後一個“票券”之間的關係
- 然而：當有梯度消失問題時，將會很難學習到這些資訊
- 因此：將會很難正確的預測出“票券”

Vanishing Gradients: Effects on RNNLM

- **LM task:** The writer of this books ____
 - The writer of this books is ... (Correct Answer)
 - Syntactic recency : The writer of this books is - Correct
 - Sequential recency : The writer of this books are – Incorrect
 - According to Linzen et al., RNN is prone to make the sequential recency error because of vanishing gradients problems.
- 
- The diagram consists of two blue curved arrows. The first arrow starts from the word 'is' in the third bullet point and points to the blank space in the first bullet point. The second arrow starts from the word 'are' in the fourth bullet point and points to the word 'is' in the third bullet point.



Long Short-Term Memory (LSTM)

LSTM: Introduction

- LSTM : 用來解決梯度消失的其中一種RNN模型
- 在每一個時序 t 都有一個隱含狀態 h^t 及一個細胞狀態 c^t
 - 每個狀態都是一個 n 維的向量
 - 細胞狀態用來儲存較遠的資訊
 - LSTM可以清除、寫入以及讀取細胞狀態的資訊
- 每個輯閘控制都可清除、寫入及讀取的資訊
 - 每個輯閘都是一個 n 維的向量
 - 輯閘的輸出介於0到1之間
 - 輯閘會根據目前的資訊來計算

LSTM: Equations

- We have a sequence of inputs $x^{(t)}$, and a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$ are computed.

- $f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f)$

Forget Gate

- $i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i)$

Input Gate

- $o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$

Output Gate

- $\tilde{c}^{(t)} = \tanh(W_c h^{(t-1)} + U_c x^{(t)} + b_c)$

New Cell Content

- $c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$

Cell State

- $h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$

Hidden State

How does LSTM solve vanishing gradients?

- 藉由採用LSTM比較容易將之前時序的資料保留
- LSTM可能還是有梯度消失的問題。然而，LSTM提供一個方法保留離輸出較遠之資訊。

LSTM: Real-World Success

- The SOTA that LSTM achieves (2013-2015)
 - Hang-writing recognition
 - Speech Recognition
 - Machine Translation
 - Parsing
 - Image Captioning
- However, other approaches (Transformer) become more dominant in 2019.
- E.g. WMT conference
- In WMT 2016, RNN related papers contain 44 times.
- In WMT 2018, RNN only appears 9 times. However, Transformer has 63 times.



Gated Recurrent Unit (GRU)

GRU: Equation

- 由 Chao et al. 提出為LSTM 的簡化版
- GRU 沒有細胞狀態。GRU 只包含輸入狀態 $x^{(t)}$ 及隱含狀態 $h^{(t)}$
- $u^{(t)} = \sigma(W_u h^{(t-1)} + U_u x^{(t)} + b_u)$
- $r^{(t)} = \sigma(W_r h^{(t-1)} + U_r x^{(t)} + b_r)$
- $\tilde{h}^{(t)} = \tanh(W_c(r^{(t)} \circ h^{(t-1)}) + U_h x^{(t)} + b_h)$
- $h^{(t)} = (1 - u^{(t)}) \circ h^{(t-1)} + u^{(t)} \circ \tilde{h}^{(t)}$

Update Gate

Reset Gate

New Hidden State Content

Hidden State



LSTM & GRU Comparison

LSTM VS. GRU

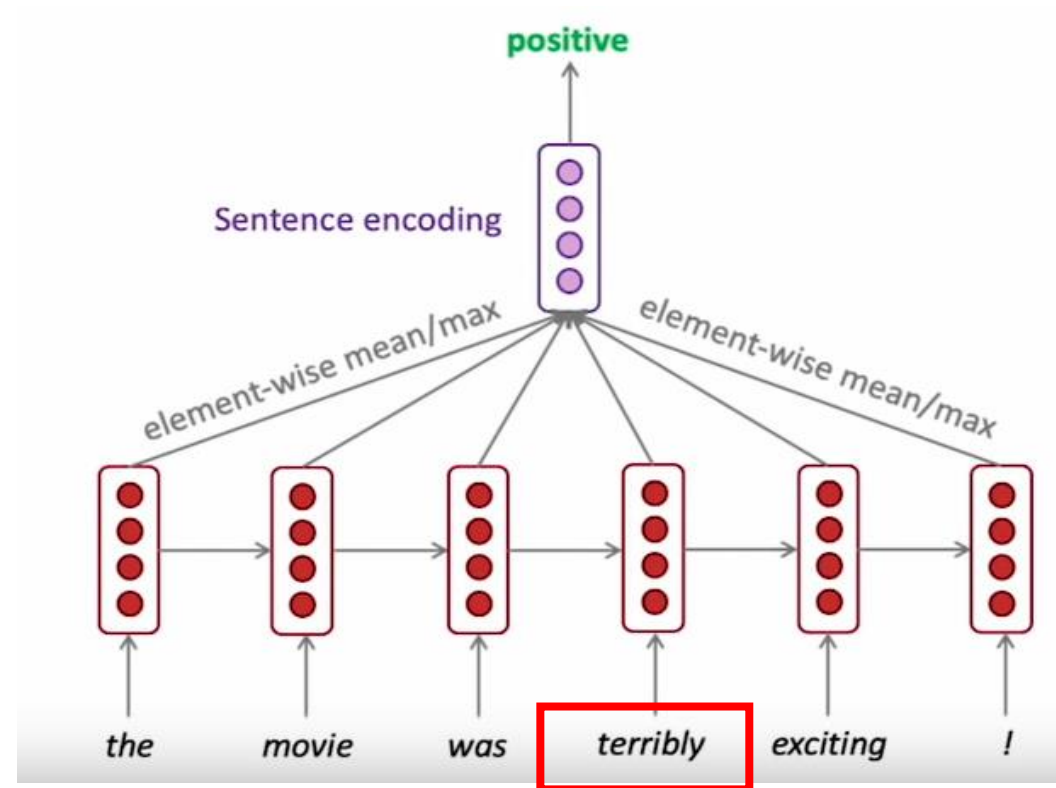
- 雖然LSTM及GRU為主要被採用的架構，仍然還有許多與輯聞相關的RNN被提出
- LSTM及GRU最大的差別為其參數及運算速度
- 目前沒有任何研究顯示哪一個架構較優
- 實際經驗：一開始可以先採用LSTM，當遇到運算問題再考慮採用GRU



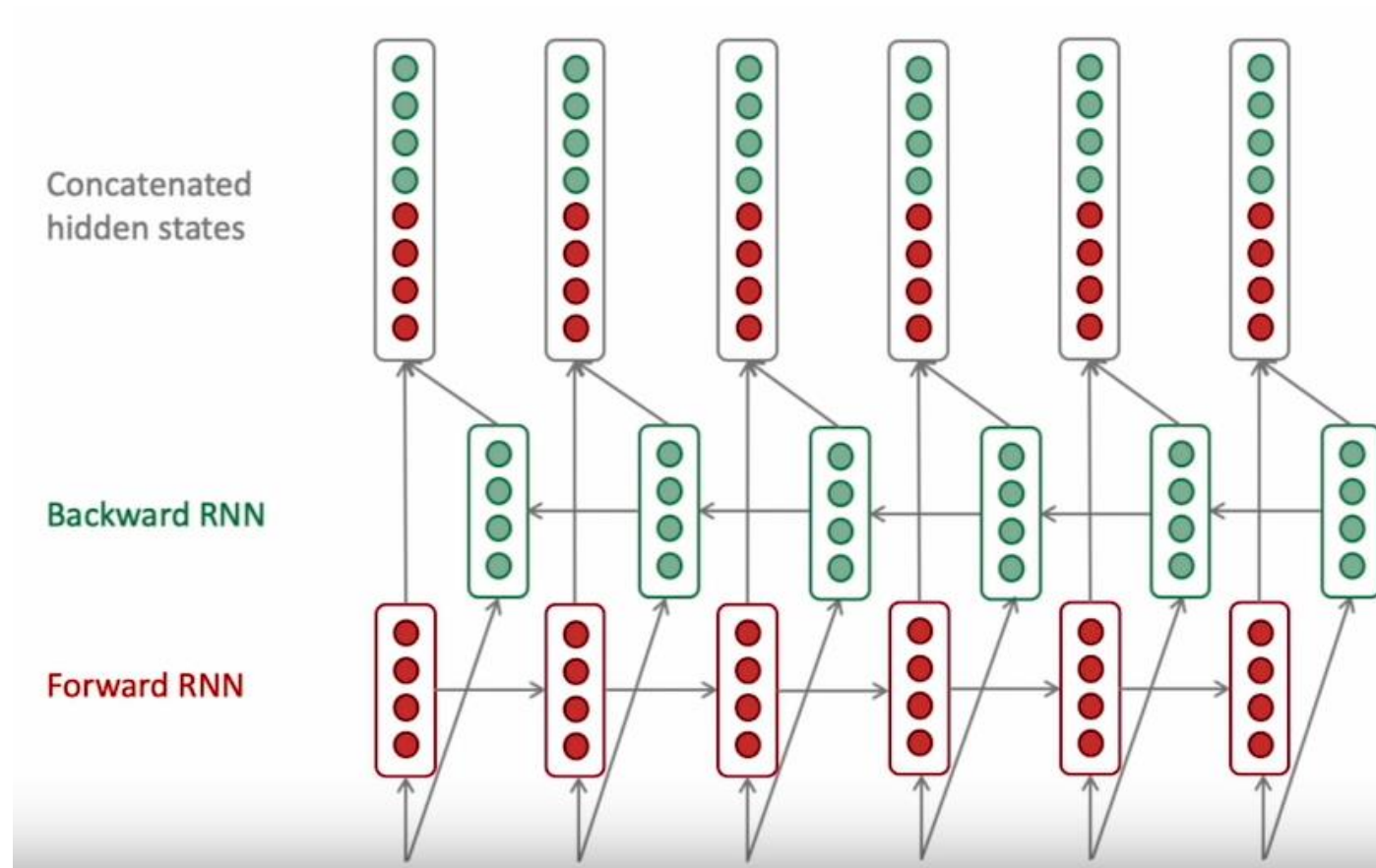
Bidirectional RNN

Bidirectional RNNs: Motivation

- Terribly can have two kinds of meaning.



Bidirectional RNN



Bidirectional RNN

- On time t
- Forward RNN : $\vec{h}^{(t)} = RNN_{FW}(\vec{h}^{(t-1)}, x^{(t)})$
- Backward RNN : $\overleftarrow{h}^{(t)} = RNN_{BW}(\overleftarrow{h}^{(t+1)}, x^{(t)})$
- Concatenated hidden state : $h^{(t)} = [\vec{h}^{(t)}, \overleftarrow{h}^{(t)}]$

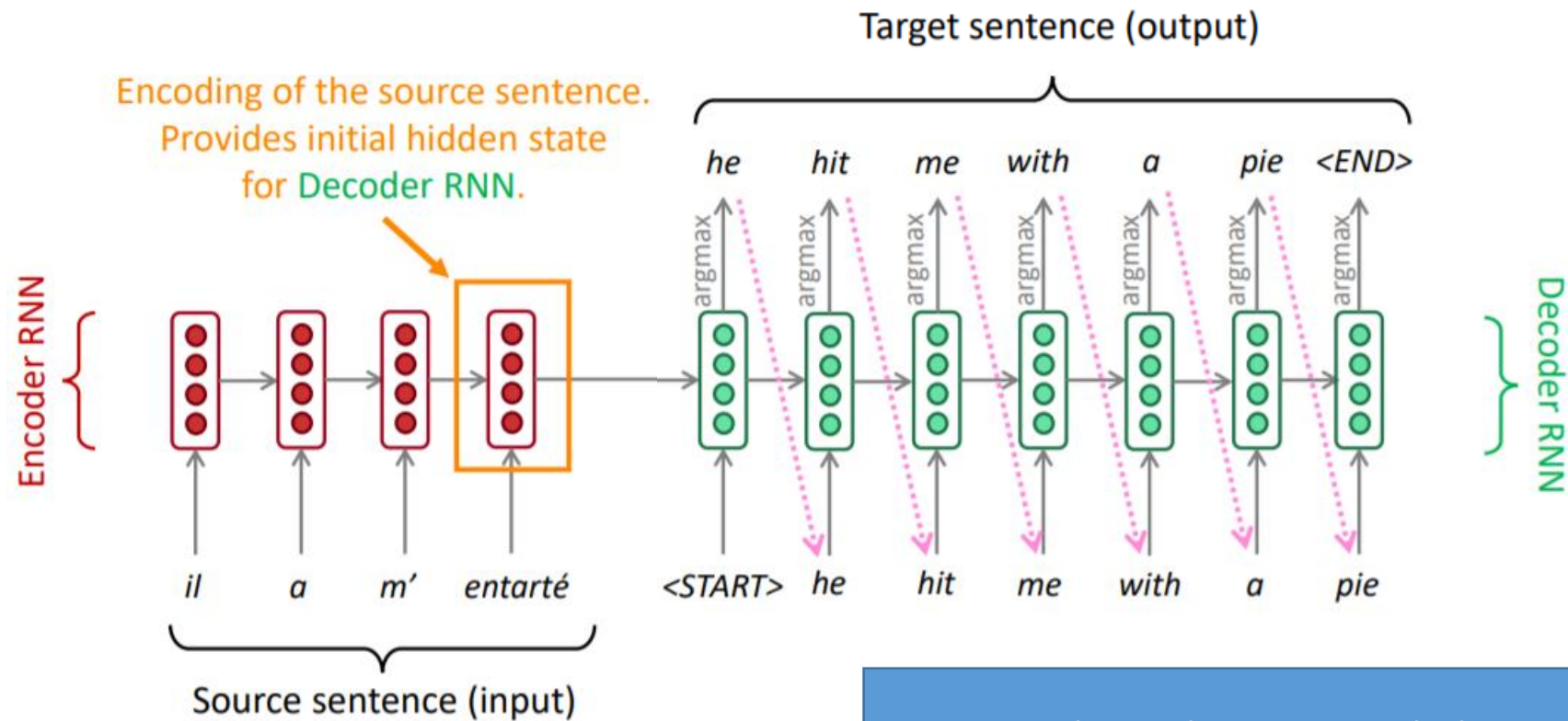
Bidirectional RNN

- 雙向RNN的假設
 - 需要有完整的句子
 - 並不適合用在語言模型 (會看到自己本身)
- 由於雙向RNN可以學習到雙向的資訊，因此雙向RNN是一個不錯的預設網路。



Neural Machine Translation (NMT)

NMT: Sequence-to-Sequence Model



Seq2Seq is versatile!!

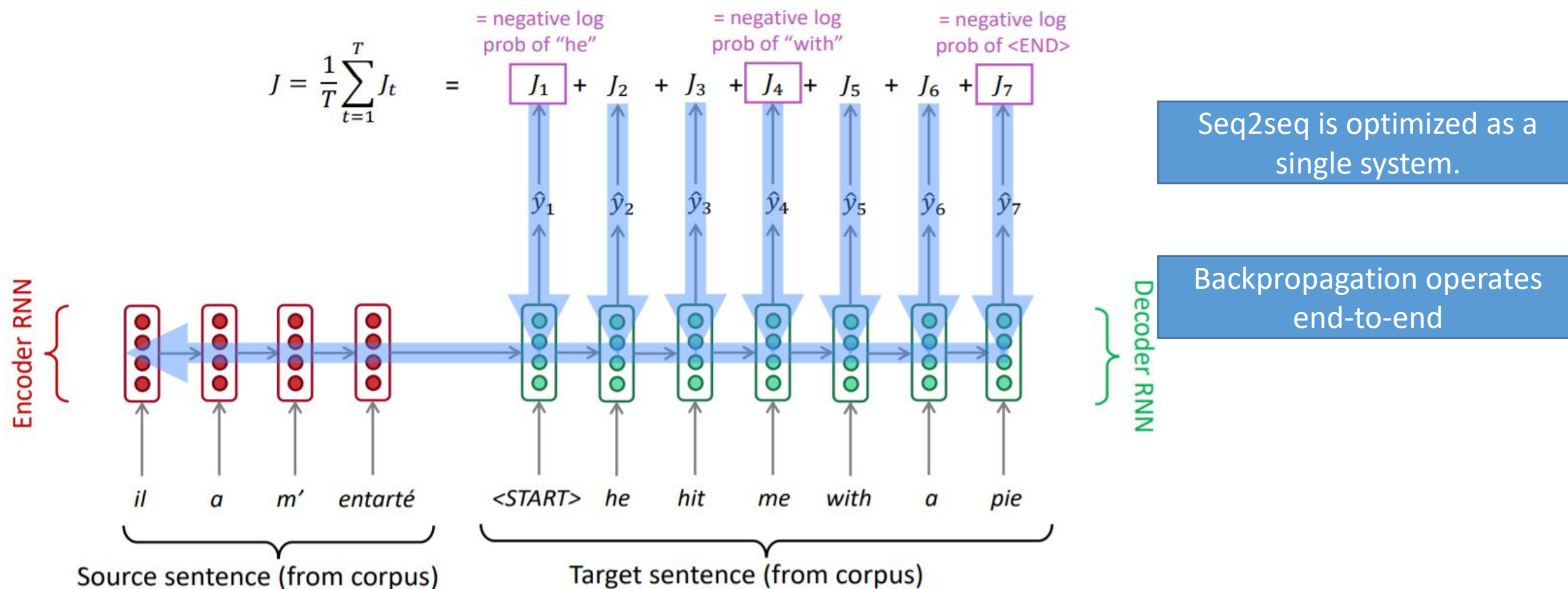
- 其他Seq2Seq模型可以運用的任務
 - 文本總結 (long text -> short text)
 - 對話系統 (previous utterance -> next utterance)
 - 文本描述影像(input image -> description of the image)
 - 文本分析 (input text -> output parse as sequence)

NMT: Sequence-to-Sequence Model

- Seq2Seq 為一個條件式語言模型的例子：
 - 語言模型：預測下一個所會出現的語詞
 - 條件式語言模型：根據輸入的句子預測下一個語詞
- Seq2Seq 會計算 $p(y|x)$:

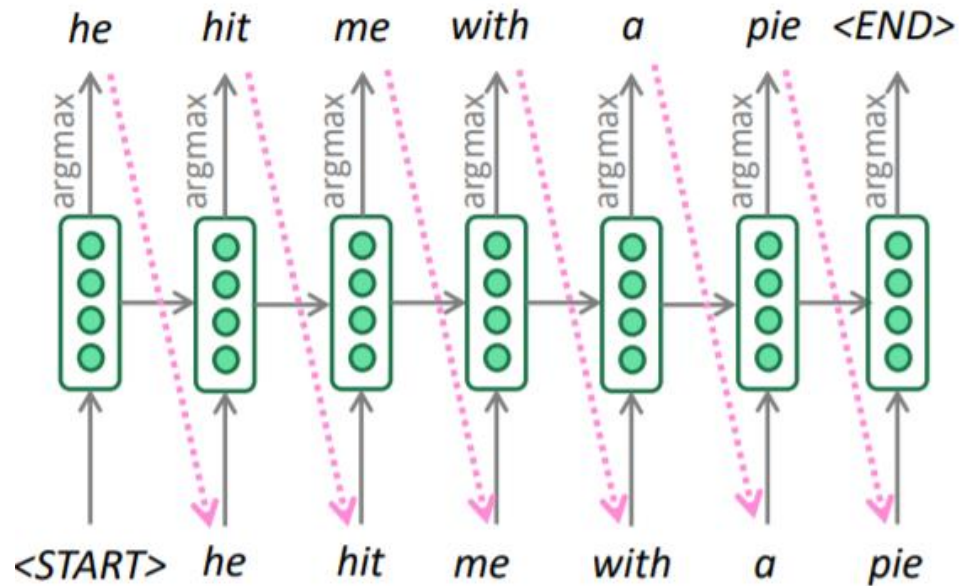
$$p(y|x) = p(y_1|x)p(y_2|y_1, x) \dots p(y_T|y_1 \dots y_{T-1}, x)$$

NMT: Training Steps



Greedy Decoding

- 目前為止，Decoder主要將有最大機率的語詞輸出



- 這方法稱作Greedy Decoding

NMT: the Biggest Success Story of NLP Deep Learning

- 2014年前： NMT為較少研究的主題
- 2014年： Seq2Seq的論文被發表
- 2016年: Google翻譯從SMT轉成NMT
- 雖然SMT已經被許多研究員維護多年，但是NMT的表現仍然超越了SMT的表現



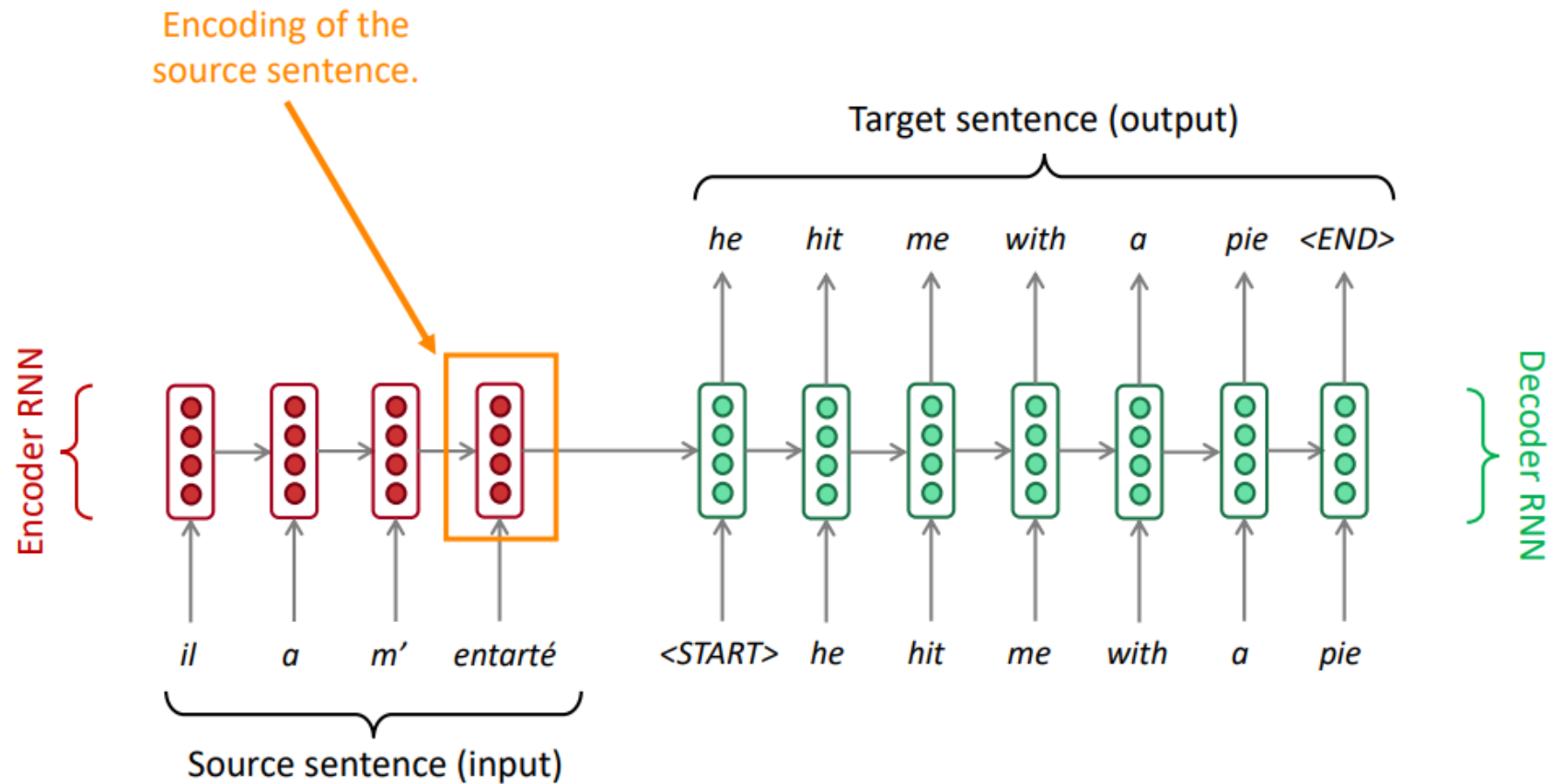
Attention

NMT Research Continues

- NMT在自然語言處理為一標記性的任務。
- NMT也促使了許多自然語言與深度學習的研究。
- 2019年： NMT 的研究仍然繼續進展
 - Seq2Seq也有了許多的演進
 - 然而：最核心的演進為

Attention

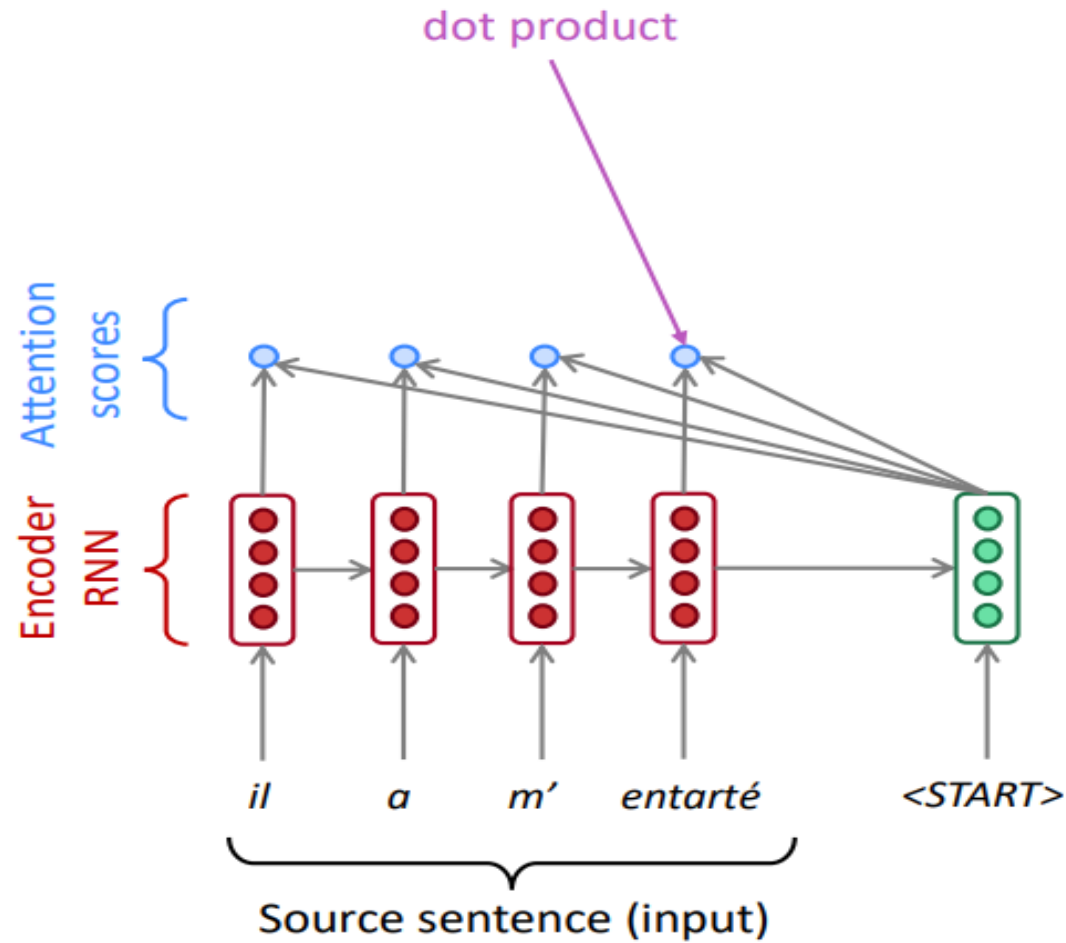
Seq2Seq: the Bottleneck Problem



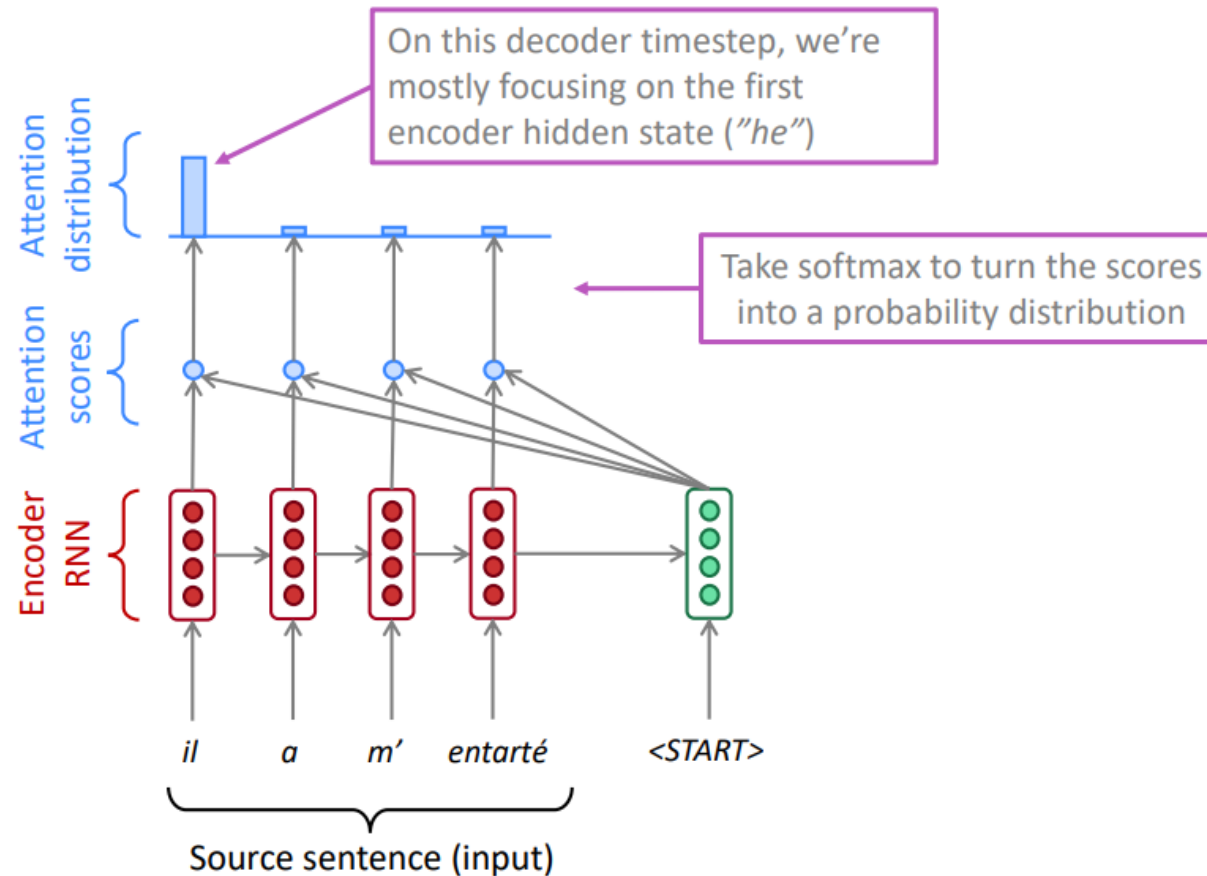
Attention

- Attention 解決了原本Seq2Seq訓練時所會遇到的問題
- 想法：將解碼的隱含狀態注意到每一個編碼的隱含狀態

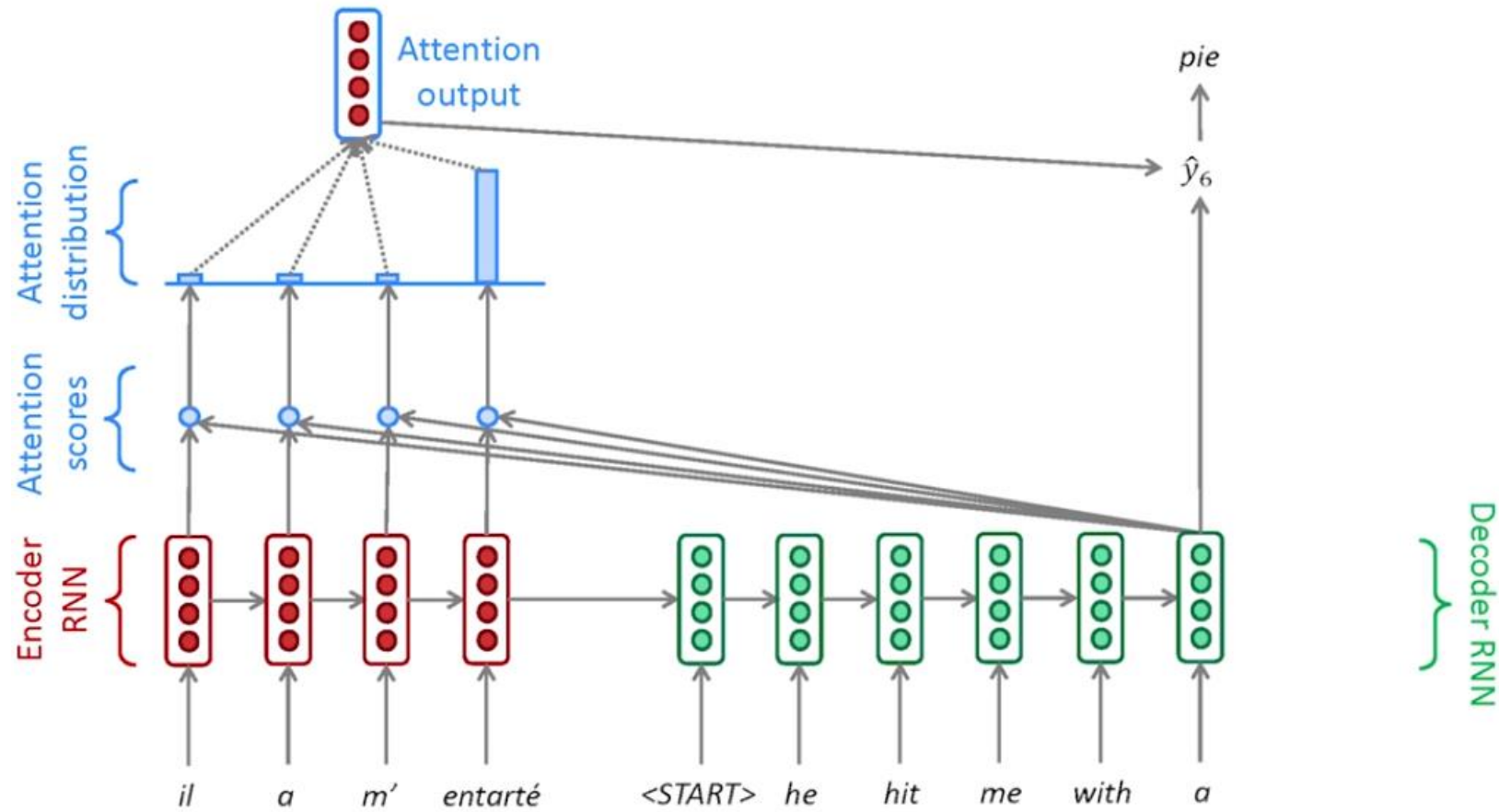
Seq2Seq with Attention



Seq2Seq with Attention



Seq2Seq with Attention



Attention: Equations

- 假設編碼器的隱含狀態如下： $h_1, \dots, h_N \in \mathbb{R}^h$

- 假設時序 t 的解碼器隱含狀態為 $s_t \in \mathbb{R}^h$

- 時序 t 的注意力分數為 e^t ：

$$e^t = [S_t^T h_1, \dots, S_t^T h_N] \in \mathbb{R}^N$$

- 運用Softmax函數將時序 t 的注意力分數轉成機率分布 α^t

$$\alpha^t = \text{softmax}(e^t) \in \mathbb{R}^N$$

- 利用注意力的機率分布與時序 t 之解碼器隱含狀態計算計算總合 a_t

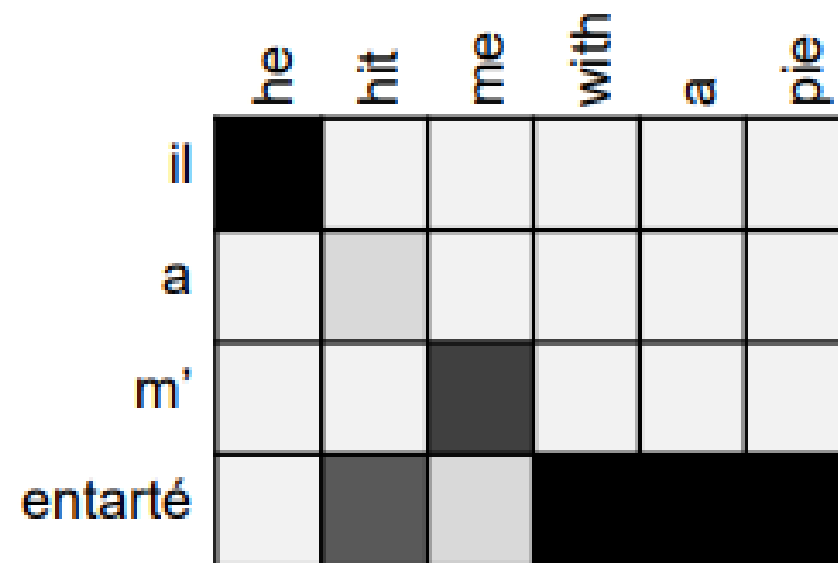
$$a_t = \sum_{i=1}^N \alpha_i^t h_i \in \mathbb{R}^h$$

- 最後將 a_t 與解碼器隱含狀態做Concat

$$[a_t, s_t] \in \mathbb{R}^{2h}$$

Attention is Great

- Attention
 - 改進了NMT的表現
 - 解決了訓練Seq2Seq模型所會遇到的問題
 - 幫忙解決梯度消失的問題
 - 讓我們更容易理解模型的狀態
 - 藉由注意力分布來了解解碼的所注意的位置





Transformer

Transformer Models

ULMfit
Jan 2018
Training:
1 GPU day



All of these models are Transformer architecture models ... so maybe we had better learn about Transformers?

GPT
June 2018
Training
240 GPU days



BERT
Oct 2018
Training
256 TPU days
~320–560
GPU days

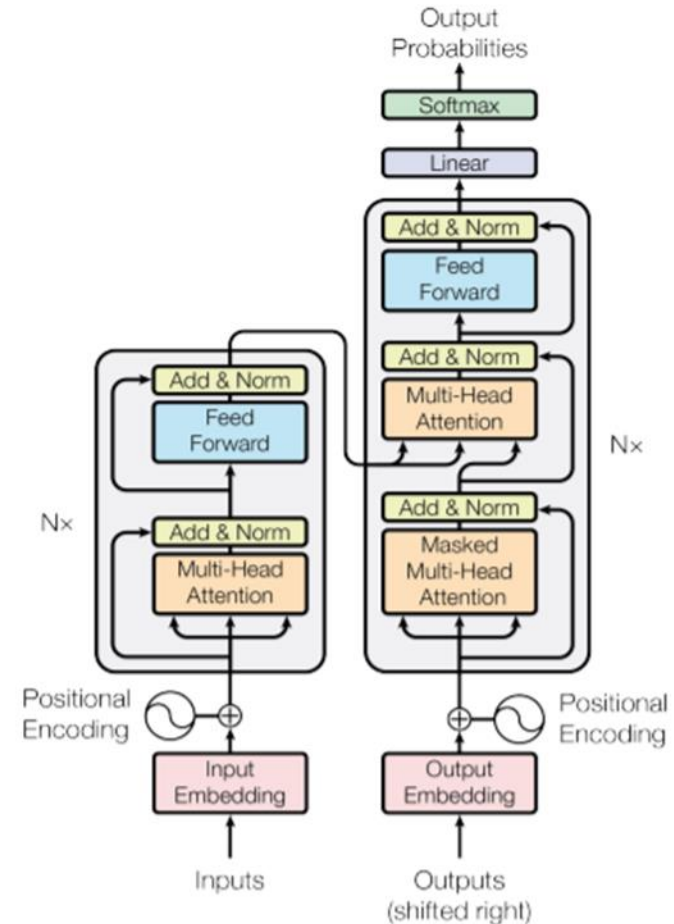


GPT-2
Feb 2019
Training
~2048 TPU v3
days according to
[a reddit thread](#)



Transformer Overview

- Non-Recurrent sequence-to-sequence encoder-to-decoder model.
- Task : Machine Translation with parallel corpus
- Predict each translated word
- Loss Function
 - Cross-Entropy loss on top of the softmax classifier



Dot-Product Attention

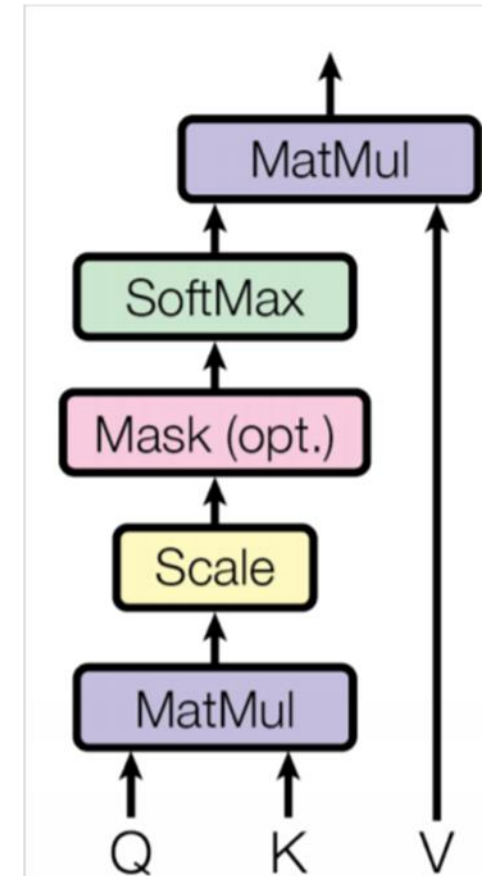
- Inputs: a query q , and a set of key-value (k-v), where
 - q , k , and v are vector.
- Output is a weighted sum of values, where
 - Dimension of queries and key are the same which are d_k
 - Dimension of values are d_v

$$A(Q, K, V) = \text{softmax}(Q^T K)V$$

Scaled Dot-Product Attention

- Problem: As d_k gets large, the variance of $q^T k$ increases -> some values inside Softmax increases -> Softmax becomes very peaked -> As a result, Gradient becomes smaller
- Solution : Scaled by $1 / \sqrt{d_k}$

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

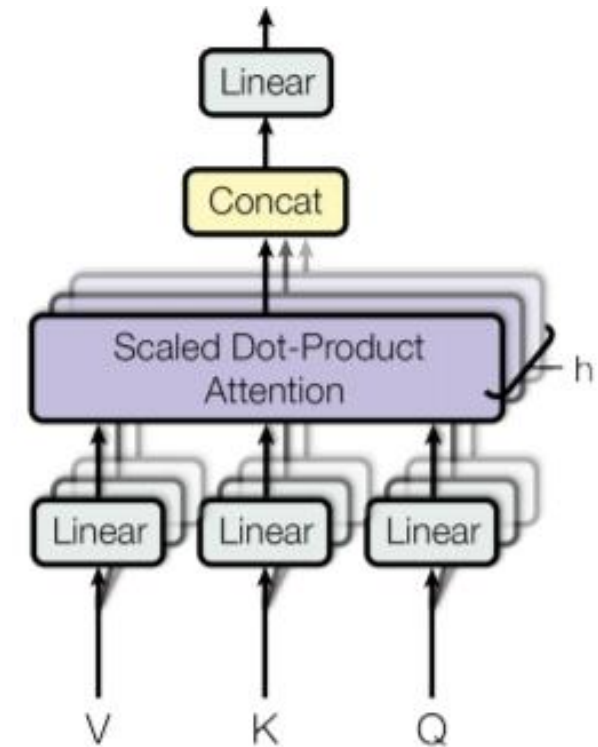


Multi-head attention

- Problems with simple attention:
 - Cannot have multiple attention
- Solution: Multi-head attention
 1. Feed Q, K and V into linear projection layer
 2. Apply attention
 3. Concatenate output
 4. Pipe through linear layer

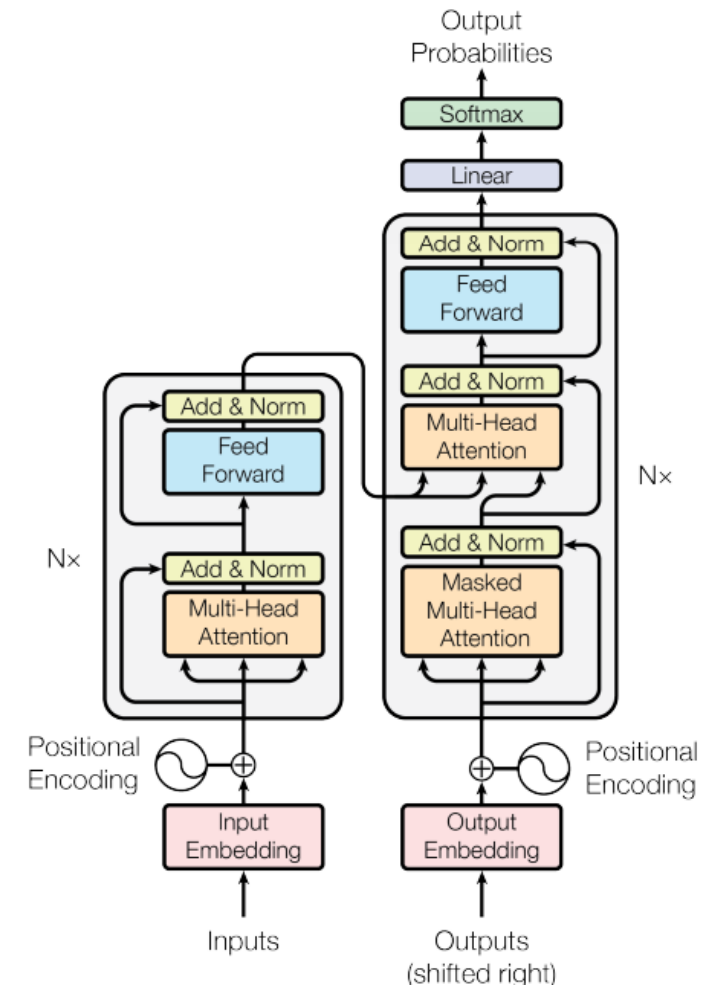
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$



Transformer: Complete Encoder and Decoder

- Blocks are repeated.
- Encoder: 6 layers
- Decoder: 6 layers



References

- 齋藤康毅, Deep Learning 2: 用Python進行自然語言處理的基礎理論實作.
- Manning et al., CS224n Natural Language Processing with Deep Learning, Stanford University.
- Vaswani et al., Attention Is All You Need.