TTS 程式實作進度及步驟整理

Sian-Yi Chen

Advisors : Tay-Jyi Lin and Chingwei Yeh

Outline

Action item

- TTS 程式實作進度及步驟整理，整理我的處理過程、方法

Status report

- TTS 程式實作進度
- 舉兩個實作過程，以及方法
  - Positional Encoding
    - input、output
    - python 版本步驟、各功能
    - 實作過程、方法
  - Multi-Head Attention
    - input、output
    - python 版本步驟、各功能
    - 實作過程、方法

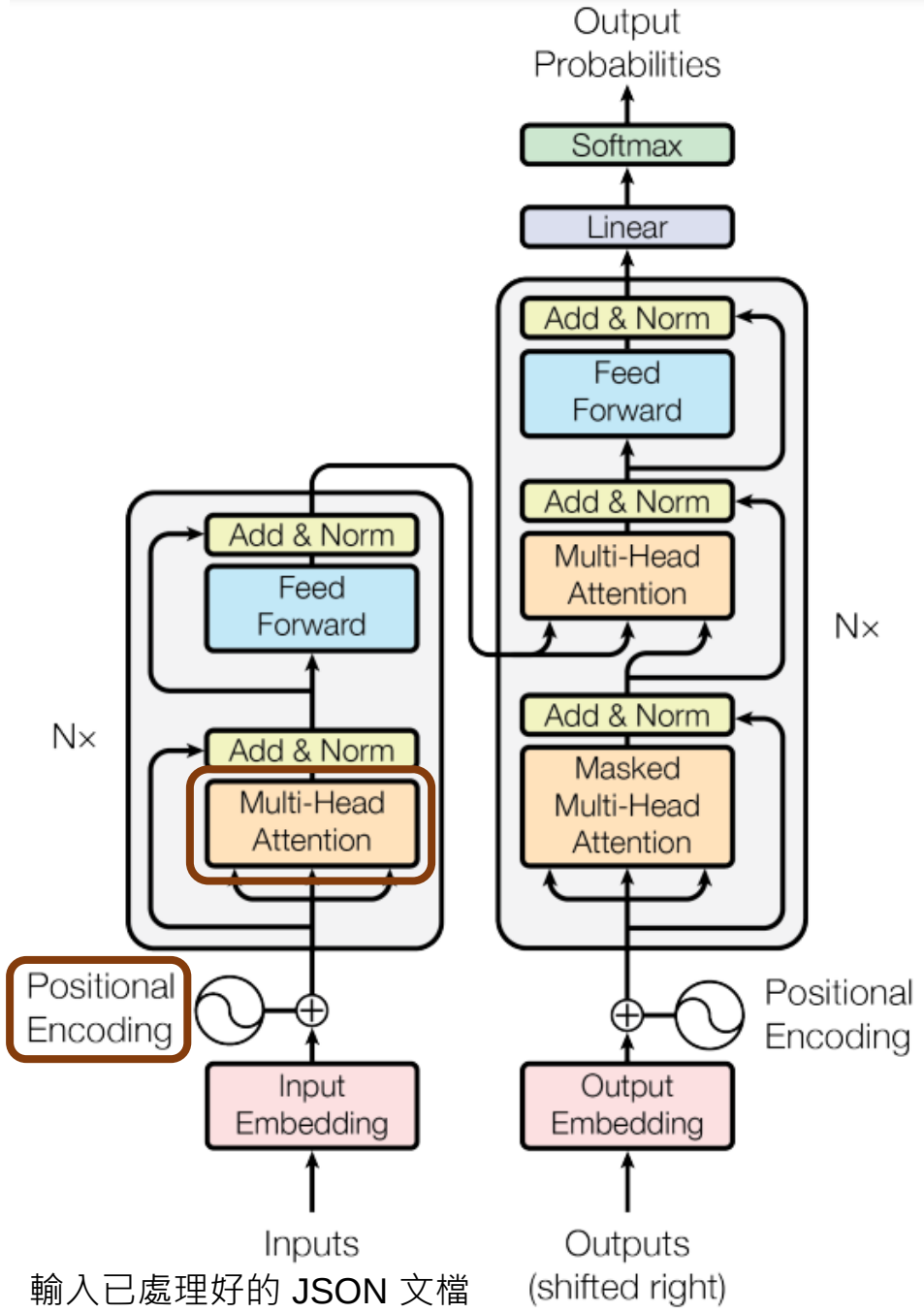


Figure 1: The Transformer - model architecture.

A. Vaswani, et al., “Attention is all you need.,” arXiv e-prints:1706.03762, 2017

TTS 程式實作進度及步驟整理

目前已使用實際值確認無誤的部分

輸入已處理好的 input JSON 文檔

Transformer
- (embed)
  - **Embedding**
  - **ScaledPositionalEncoding**
- (encoder) **x 6**
  - (self_attention): Multi-Head Attention
    - (linear_q): **Linear()**
    - (linear_k): **Linear()**
    - (linear_v): **Linear()**
    - (linear_out): **Linear()**
  - (normalize):**LayerNorm()**
  - (feed forward):
    - (w_1): **Linear()**
    - (w_2): **Linear()**
    - **ReLU()**
  - (normalize):**LayerNorm()**

(embed)
- (prenet)
  - **Linear()**
  - **ReLU()**
  - **ScaledPositionalEncoding**

x 6
- (self_attention): Multi-Head Attention
  - (linear_q): **Linear()**
  - (linear_k): **Linear()**
  - (linear_v): **Linear()**
  - (linear_out): **Linear()**
- (normalize):**LayerNorm()**
- (src_attention): Multi-Head Attention
  - (linear_q): **Linear()**
  - (linear_k): **Linear()**
  - (linear_v): **Linear()**
  - (linear_out): **Linear()**
- (normalize):**LayerNorm()**
- (feed forward):
  - (w_1): **Linear()**
  - (w_2): **Linear()**
  - **ReLU()**
- (normalize):**LayerNorm()**

x 4
- (postnet)
  - **Conv1d()**
  - **BatchNorm1d()**
  - **Tanh()**

綠色：已完成底層功能，但仍需確認演算法，與運算結果
- Embedding
- ScaledPositionalEncoding
- Linear
- LayerNorm
- ReLU

紅色：尚未實作
- Conv1d()
- BatchNorm1d()
- Tanh()

Positional Encoding(PE)

1. 先使用cmd確認python版本中torch所使用的函數
   - input
   - output
   - 使用的運算式或公式
2. 用C實現各函數功能
3. 使用低維度測試C程式正確性
   - 目標使用 [19, 384]
   - 低維度使用 [2, 4] 測試
4. 功能完成再使用目標維度 [19, 384] 測試正確性
5. 將所有函數功能input接output確認結果
6. 如果看了document或是找完資料仍無法理解函數功能，才詢問睏飽學長

$$PE = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

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

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

Positional Encoding(PE) formula and table

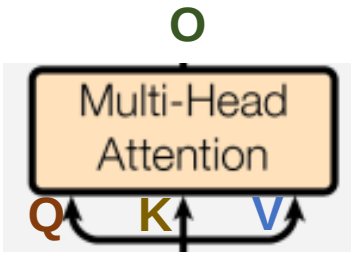
Scaled positional encoding

- input: Embedding output, dimension[19, 384]
- output: dimension[19, 384]

步驟、功能

- 產生一張根據input的第一維使用0填充的表
- position資訊(被除數)
  1. 一張1維值從1-384的表，間隔1，維度：[384]
  2. 擴展維度至2維，維度：[384, 1]
- 除數
  1. 一張1維值從1-384的表，間隔2，維度：[192]
  2. 乘上-(math.log(10000)/384
  3. 取指數函數exp
- 創建positional encoding(PE)表
  1. 偶數列position * 除數取sin
  2. 奇數列position * 除數取cos
- input + alpha * PE
  1. 取得alpha值
  2. 確認結果

Multi-Head Attention



input

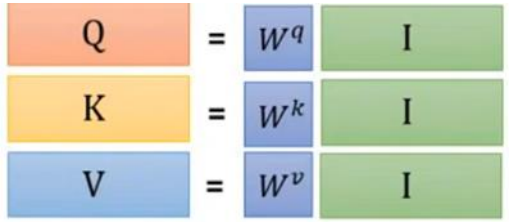
Multi-Head Attention layer

Multi-Head Attention

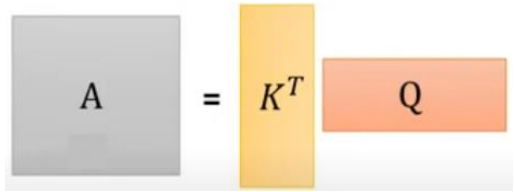
- input: scaled positional encoding output [19, 384] -> q, k, v
- output: [19, 384]

步驟、功能

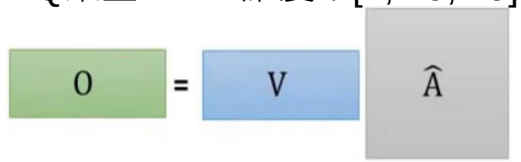
- scaled positional encoding output當作input輸入，並處理成q、k、v，
  使用torch指令linear實現
  - linear(query) = [input][weight$_q$]T + bias$_q$ = [19, 384] = q
  - 取得weight$_q$，先直接宣告值，後改成讀取txt檔的方式
  - linear(key) = [input][weight$_k$]^T + bias$_k$ = [19, 384] = k
  - linear(value) = [input][weight$_v$]^T + bias$_v$ = [19, 384] = k
- 拆分指定維度，2維轉換成3維
  - q.view = 2維轉換成3維，[19, 384] => [19, 4, 384/4] => [19. 4, 96]
  - k.view = 2維轉換成3維，[19, 384] => [19, 4, 384/4] => [19, 4, 96]
  - v.view = 2維轉換成3維，[19, 384] => [19, 4, 384/4] => [19, 4, 96]
- 指定維度轉置
  - q.view.transpose = [19, 4, 96] => [4, 19, 96]
  - k.view.transpose = [19, 4, 96] => [4, 19, 96]
  - v.view.transpose = [19, 4, 96] => [4, 19, 96]
- q與k^{T}相乘， k^T=(k.view.transpose)^T = [4, 19, 96] => [4, 96, 19]
  - q.view.transpose * (k.view.transpose)^T = [4, 19, 96]*[4, 96, 19] =
    [4, 19, 19]



1. 取得 Q、K、V



2. Q乘上K^T，A維度：[4, 19, 19]



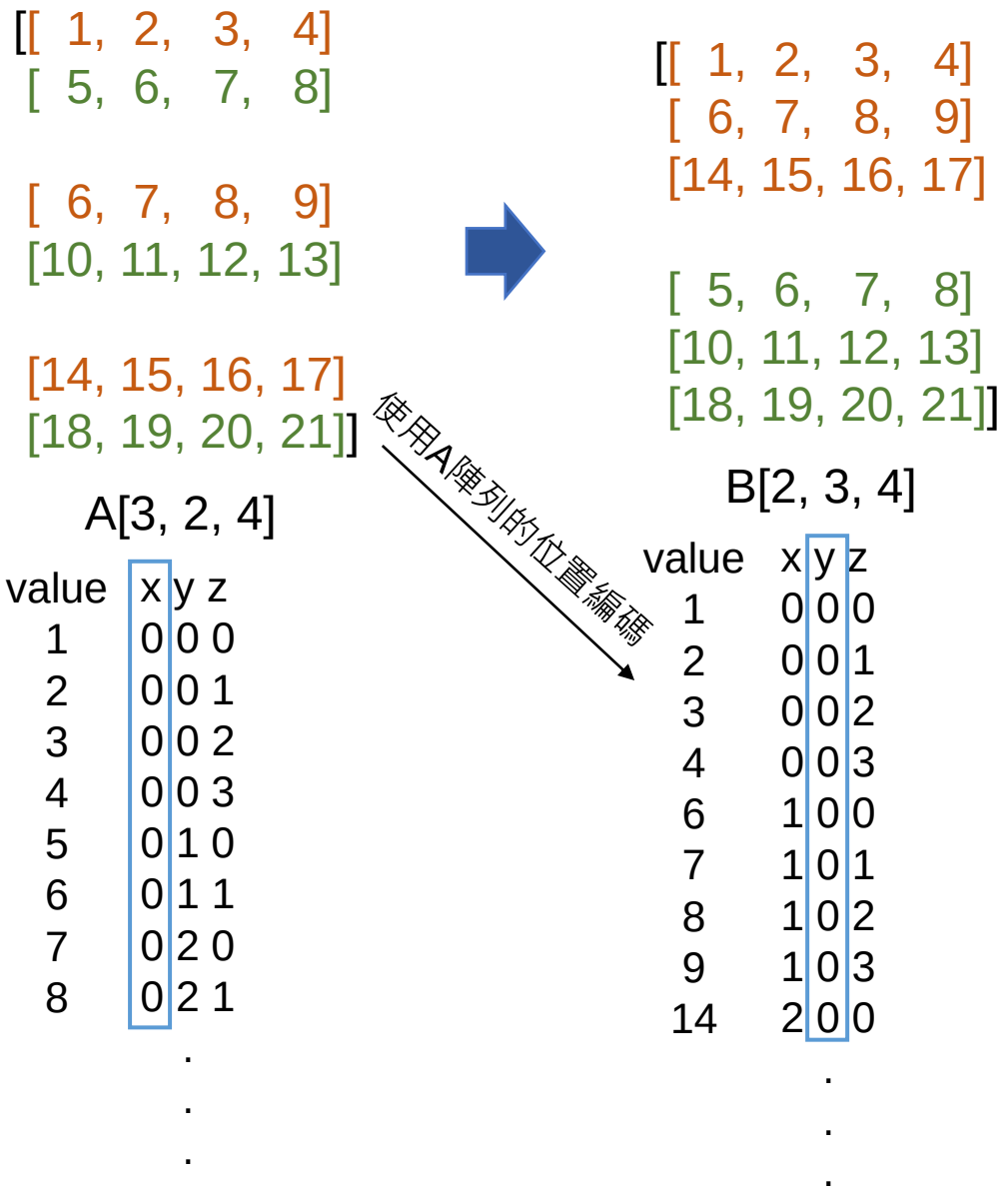
3. 最後在與V相乘，尚未實作

多維陣列debug方式

目標處理 [19, 4, 96] => [4, 19, 96]
預先使用低維度陣列實現 A[3, 2, 4] => B[2, 3, 4]

若程式錯誤，就在紙上先將過程想清楚，再重新寫程式

除了維度轉換，還有多維矩陣相乘、擴增維度等操作，執行程式時皆遇到最後運算數值不正確，因此在這個部分花了不少時間。



發現多維陣列，任一維度交換，可以直接改變x、y值即可