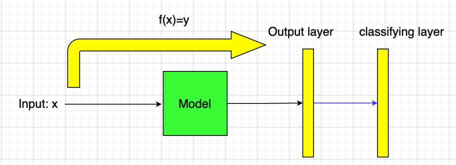
人工智慧專題

劉瓊如

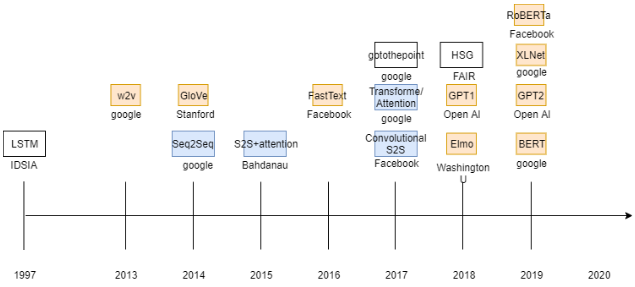
October 14, 2022

Langurage Models

- word2v
  - 優點 (vector similarity)
  - 缺點新字要重新訓練
- GloVe 字頻
- ELMo (next word)
- BERT (克漏字填空)
  - 優點：character level，可以組新字，組 sentence vector
  - 缺點：wors similarity 較差，例如有重複字："雪白"、"亮白" word similarity 接近，"白目" 字義不接近，但 word similarity 接近
- GPT: 適合生成模型
- XLNet
- Roberta

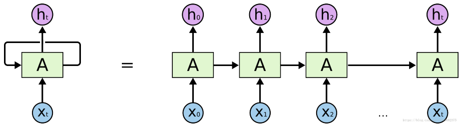


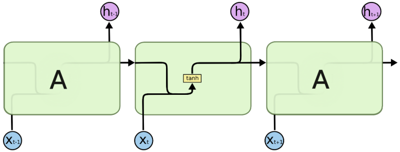
NLP History



RoBERTa
Facebook
gotothepoint
google
HSG
FAIR
XLNet
google
w2v
google
GloVe
Stanford
FastText
Facebook
Transforme/
Attention
google
GPT1
Open AI
GPT2
Open AI
LSTM
IDSIA
Seq2Seq
google
S2S+attention
Bahdanau
Convolutional
S2S
Facebook
Elmo
Washington
U
BERT
google
1997
2013
2014
2015
2016
2017
2018
2019
2020

Recurrent Neural Networks (RNNs)





The repeating module in a standard RNN contains a single layer.

$$h_t(x_t) = \sigma(h_{t-1}, x_t) = \sigma(W_{hh}h_{t-1} + W_{hx}x_t + b)$$
$$\hat{y}_t(x_t) = W_{hy}h_t(x_t) + b_y$$

Next word prediction

$$\mathbf{p}_t = \mathrm{softmax}(\hat{y}) \quad \in \mathbb{R}^{|V|}$$

$$L = -\frac{1}{T}\sum_{t=1}^{T} \mathbf{y}_t \log \mathbf{p}_t$$

$\mathbf{p}_t$: output probability distribution of the next word
h_T: sentence representation

$$\frac{\partial L_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{p}_t}\frac{\partial \mathbf{p}_t}{\partial \mathbf{h}_t}\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k}\frac{\partial \mathbf{h}_k}{\partial W}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \Pi_{j=k+1}^{t}\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}} = \Pi_{j=k+1}^{t}\mathrm{diag}(\sigma'(Wh_{j-1} + \cdots)) \times W$$

Gradient estimation

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T}\sum_{k=1}^{t}\frac{\partial L_t}{\partial \mathbf{p}_t}\frac{\partial \mathbf{p}_t}{\partial \mathbf{h}_t}(\Pi_{j=k+1}^{t}\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}})\frac{\partial \mathbf{h}_k}{\partial W}$$

If $\|\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}\| \leq \|\mathrm{diag}(\sigma'(Wh_{j-1}))\|\|W\| \leq \beta_h\beta_W$ by taking their L_2-norm

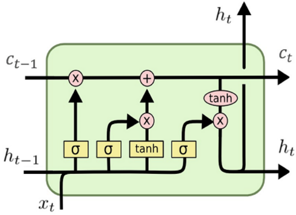
$$\|A\|_2 = \sup_{\mathbf{x}\neq 0}\frac{\|A\mathbf{x}\|_2}{\|x\|_2} = \sigma_{\max}(A)$$

$$\|\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k}\| \leq \|\Pi_{j=k+1}^{t}\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}})\| \leq (\beta_h\beta_W)^{t-k}$$

is easily large or small if $t-k$ is sufficiently large. This could cause the Gradient Explosion Problem or Gradient Vanishing Problem

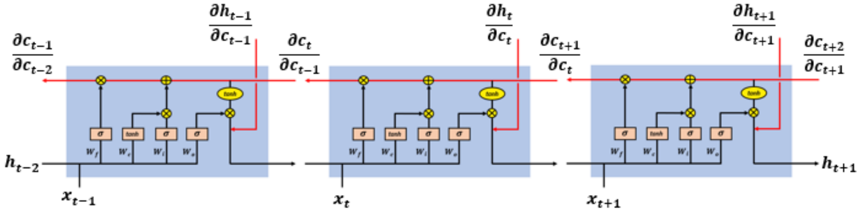
LSTM

論文：Long Short-Term Memory in Recurrent Neural Networks (Hochreiter and Schumidhuber)1997



$$i_t = \sigma(W_x^{(i)} x_t + W_h^{(i)} h_{t-1} + b_i) \quad \text{input gate}$$
$$f_t = \sigma(W_x^{(f)} x_t + W_h^{(f)} h_{t-1} + b_f) \quad \text{forget gate}$$
$$o_t = \sigma(W_x^{(o)} x_t + W_h^{(o)} h_{t-1} + b_o) \quad \text{output gate}$$
$$\tilde{c}_t = \tanh(W_x^{(\tilde{c})} x_t + W_h^{(\tilde{c})} h_{t-1} + b_{\tilde{c}}) \quad \text{temporary memory}$$
$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \quad \text{memory gate}$$
$$h_t = o_t \odot \tanh(c_t),$$

Backpropagation



$$
\frac{\partial L_t}{\partial W_{(\cdot)}} = \begin{cases} \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{p}_t} \frac{\partial \mathbf{p}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{c}_t} \frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_k} \frac{\partial \mathbf{c}_k}{\partial (\cdot)_k} \frac{\partial (\cdot)_k}{\partial W_{(\cdot)}} & (\cdot) = i, f, \tilde{c} \\ \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{p}_t} \frac{\partial \mathbf{p}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \mathbf{o}_k} \frac{\partial \mathbf{o}_k}{\partial W^{(o)}} & \text{otherwise} \end{cases}
$$

$$
\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_k} = \Pi_{t=k+1}^{t} \frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}}
$$

$$
\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \Pi_{t=k+1}^{t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \Pi_{t=k+1}^{t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{c}_t} \frac{\partial \mathbf{c}_t}{\partial \mathbf{h}_{t-1}}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} &= \frac{\partial}{\partial \mathbf{c}_{t-1}}[\mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t] \\
&= \mathrm{diag}(\mathbf{f}_t) + \mathrm{diag}(\mathbf{c}_{t-1})\frac{\partial \mathbf{f}_t}{\partial \mathbf{c}_{t-1}} + \mathrm{diag}(\tilde{\mathbf{c}}_t)\frac{\partial \mathbf{i}_t}{\partial \mathbf{c}_{t-1}} + \mathrm{diag}(i_t)\frac{\partial \tilde{\mathbf{c}}_t}{\partial \mathbf{c}_{t-1}} \\
&= A_t + B_t + C_t + D_t
\end{aligned}
$$

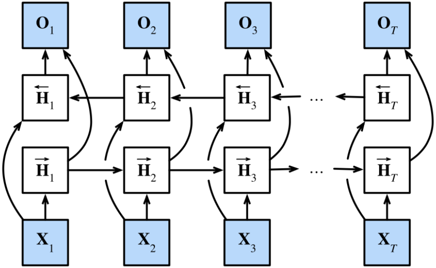
$$
\begin{aligned}
\frac{\partial \mathbf{f}_t}{\partial \mathbf{c}_{t-1}} &= \frac{\partial \mathbf{f}_t}{\partial \mathbf{h}_{t-1}}\frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{c}_{t-1}} = W_h^{(f)}\mathrm{diag}(\sigma'(W_h^{(f)}h_{t-1}))\mathrm{diag}(o_t)\mathrm{diag}(\tanh'(c_{t-1})) \\
\frac{\partial \mathbf{i}_t}{\partial \mathbf{c}_{t-1}} &= \frac{\partial \mathbf{i}_t}{\partial \mathbf{h}_{t-1}}\frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{c}_{t-1}} = W_h^{(i)}\mathrm{diag}(\sigma'(W_h^{(i)}h_{t-1}))\mathrm{diag}(o_t)\mathrm{diag}(\tanh'(c_{t-1})) \\
\frac{\partial \tilde{\mathbf{c}}_t}{\partial \mathbf{c}_{t-1}} &= \frac{\partial \tilde{\mathbf{c}}_t}{\partial \mathbf{h}_{t-1}}\frac{\partial \mathbf{h}_{t-1}}{\partial \mathbf{c}_{t-1}} = W_h^{(\tilde{c})}\mathrm{diag}(\tanh'(W_h^{(\tilde{c})}h_{t-1}))\mathrm{diag}(o_t)\mathrm{diag}(\tanh'(c_{t-1}
\end{aligned}
$$

Preventing the error gradients from vanishing

$$\frac{\partial L_t}{\partial W_{(\cdot)}} = \sum_{k=1}^{t} \frac{\partial L_t}{\partial \mathbf{p}_t} \frac{\partial \mathbf{p}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{c}_t} (\Pi_{t=k+1}^{t} [A_t + B_t + C_t + D_t]) \frac{\partial \mathbf{c}_k}{\partial W_{(\cdot)}}$$

因為梯度包含 forget gates 的激活向量，可以控制梯度的大小（A_t 項乘的次方較少，值較大）．可以透過參數更新，讓神經元選擇遺忘或保留記憶．

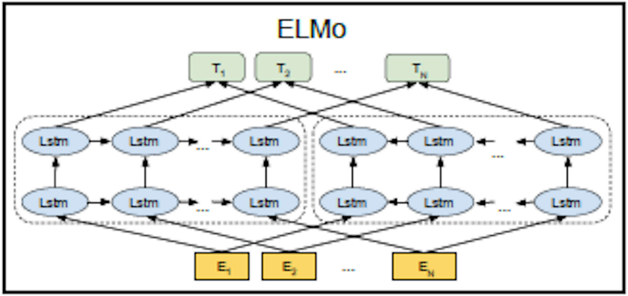
BLSTM



$\mathbf{O}_1$
$\mathbf{O}_2$
$\mathbf{O}_3$
$\mathbf{O}_T$
$\overleftarrow{\mathbf{H}}_1$
$\overleftarrow{\mathbf{H}}_2$
$\overleftarrow{\mathbf{H}}_3$
...
$\overleftarrow{\mathbf{H}}_T$
$\overrightarrow{\mathbf{H}}_1$
$\overrightarrow{\mathbf{H}}_2$
$\overrightarrow{\mathbf{H}}_3$
...
$\overrightarrow{\mathbf{H}}_T$
$\mathbf{X}_1$
$\mathbf{X}_2$
$\mathbf{X}_3$
$\mathbf{X}_T$

ELMo

論文：Deep contextualized word representations
(Allen Institute for Artificial Intelligence+University of Washington)
ELMo=(**E**mbeddings from **L**anguage **M**odels)



ELMo

Given a sentence of tokens $(t_1, t_2, \cdots, t_N)$

$$L = \sum_{k=1}^{N} \Bigg(\log p(t_k | t_1, \cdots, t_{k-1}; \vec{\Theta}_{\text{LSTM}}) + \log p(t_k | t_{k+1}, \cdots, t_N; \overleftarrow{\Theta}_{\text{LSTM}}) \Bigg),$$

For each token t_k, a L-layer biLM computes a set of $2L+1$ representations

$$\begin{aligned} R_k &= \{x_k^{LM}, \vec{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} | j = 1, \cdots, L\} \\ &= \{h_{k,j}^{LM}, j = 0, \cdots, L\}, \end{aligned}$$

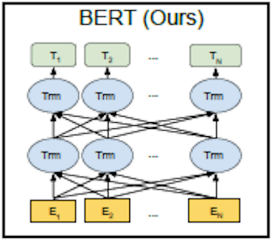
where $h_{k,0}^{LM}$ is the token layer and $h_{k,j}^{LM} = [\vec{h}_{k,j}^{LM}; \overleftarrow{h}_{k,j}^{LM}]$ for each biLM layer. For a downstream task, each word t_k has

$$\mathbf{ELMo}_k^{\text{task}} = E(R_k; \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} h_{k,j}^{LM}$$

for different tasks.

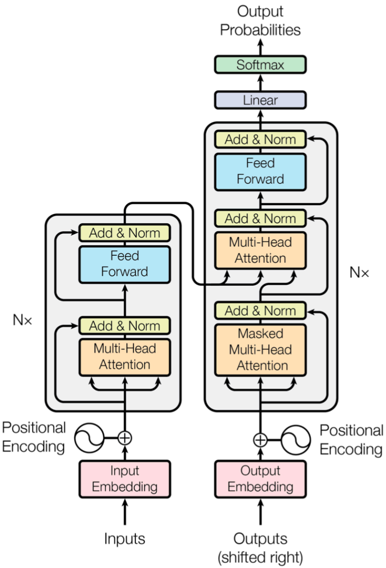
BERT

論文: Pre-training of Deep Bidirectional Transformers for Language Understanding
BERT=**B**idirection **E**ncoder **R**epresentation from **T**ransfomers 縮寫



- Encoder of Transformer
- Self-supervised Learning
- Cloze

Transformer



Transformer

- 捨棄 RNN 的序列式 input
- **Positional Encoding**: 由於一個句子可以同時進模型，為了可以區分順序

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

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

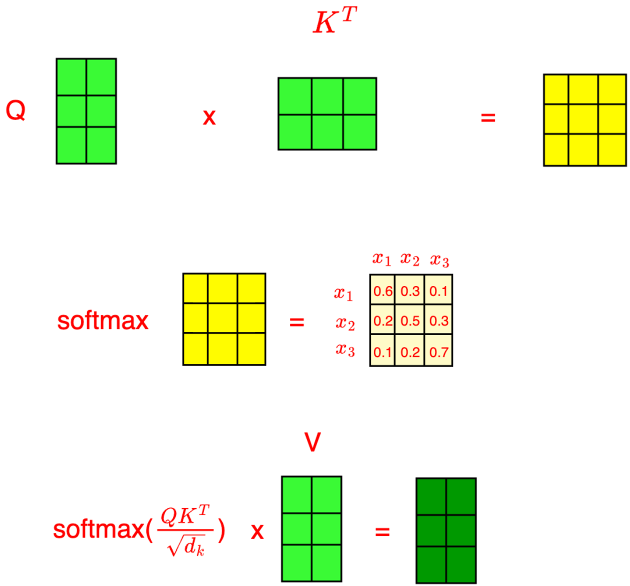
pos 代表的是位置，i 代表的是維度，偶數位置的文字會透過 sin 函數進行轉換，奇數位置的文字則透過 cos 函數進行轉換

- **Attention Mechenism**

$$\mathrm{attention}(Q, K, V) = \mathrm{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V,$$

$Q = XW_Q,\ K = XW_K,\ V = XW_V,\ X \in \mathbb{R}^{n\times d},\ W_Q, W_K \in \mathbb{R}^{d\times d_k},$
$, W_V \in \mathbb{R}^{d\times d_v}$

Attention Score

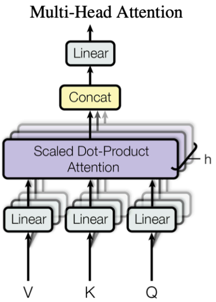


Self-Attention Sublayer

- **Multi-head attention**:

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_h)W^O$$
$$\mathrm{head}_i = \mathrm{attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$W_i^Q, W_i^K \in \mathbb{R}^{d_{\mathrm{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\mathrm{model}} \times d_v}$. Default $h = 8$, $d_k = d_v = d_{\mathrm{model}} = 64$.



Feed-Forward Sublayer

- **Position-wise Feed-Forward Networks**:

$$\mathrm{FFN}(\mathbf{x}) = \max\{0, \mathbf{x}W_1 + \mathbf{b_1}\} \cdot W_2 + \mathbf{b_2},$$

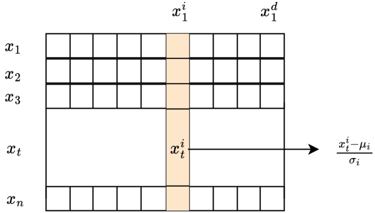
$W_1 \in \mathbb{R}^{d_{\mathrm{model}} \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_{\mathrm{model}}}$, $d_{ff} = 4d_{\mathrm{model}}$

- **Residual Connections**: We employ a residual connection around each of the two sub-layers, followed by layer normalization:

$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$$

防止 gradient vanishing

Layer Nomalization



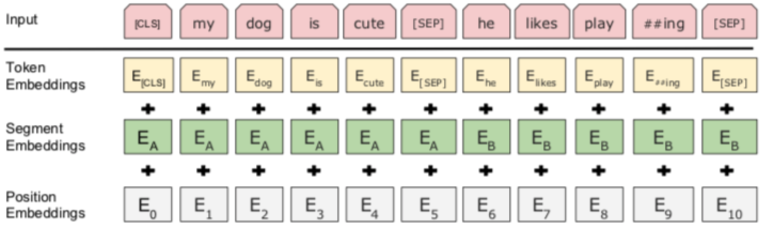
$$(\mathrm{LayerNorm}(\mathbf{x}))_t^i = \frac{(\mathbf{x})_t^i - \mu_i}{\sigma_i}$$

$$\mu_i = \frac{1}{n}\sum_{t=1}^{n}(\mathbf{x})_t^i$$

$$\sigma_i = \sqrt{\frac{1}{n}\sum_{t=1}^{n}((\mathbf{x})_t^i - \mu_i)^2}$$

Embedding

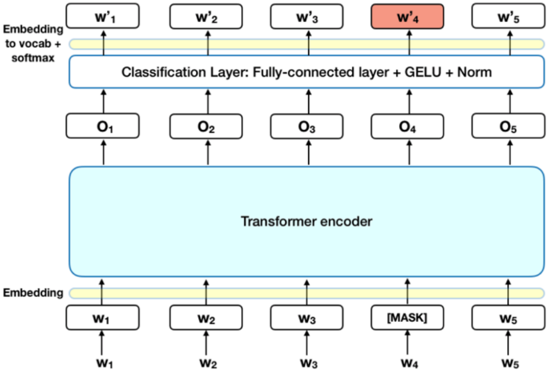
- Input embeddings: 三種 embedding 加在一起
  Token embedding+Segment embedding+Position embedding=word embedding+sentence embedding+ Position embedding



假定句字最長 512 字,不足補 0

Mask of BERT

使用 mask 推測當前的字.



$$L = -\log(p(w_t = \mathrm{Mask})|w_1, \cdots, w_{t-1}, w_{t+1}, \cdots, w_L)$$

Mask

克漏字原理

In all of our experiments, we mask 15% of all WordPiece tokens in each sequence at random. Rather than always replacing the chosen words with [MASK], the data generator will do the following:

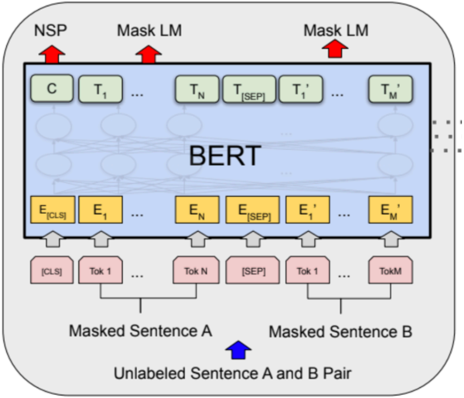
1. 80% of the time: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]
2. 10% of the time: Replace the word with a random word, e.g., my dog is hairy → my dog is apple
3. 10% of the time: Keep the word unchanged, e.g., my dog is hairy → my dog is hairy. The purpose of this is to bias the representation towards the actual observed word.

雖然只 15% 的字需要預測，不知道哪個字被換掉，所以每個字都要關注．

Pre-training BERT: Next Sentence Prediction

- 判斷是否是下句，分割成對句，若全文有 40 句，拆成 20 對: pre-train a binarized next-sentence prediction.
  Note: Use a document-level corpus rather than a shuffled sentence-level corpus
  每兩句話做一個 input，label output:IsNext, NotNext
  訓練的時候，將 50% 的句子按原文前後順序放，50% 隨便亂放
- Special token: [CLS], [Mask], [Sep]
- 假設 A->B，若輸入 [CLS] A [Sep] B，則 [CLS] token 需猜測 [IsNext]
- 若輸入 [CLS] B [Sep] A，則 [CLS] token 需猜測 [NotNext]

Pre-training



Pre-training

Training

Task1 的 loss function: 假設 document 有 $2m$ 句，$s_i = \{w_1, \cdots, w_{N_i}\}$, N= 總字數：

$$\mathcal{L}_1 = \frac{1}{N}\sum_{i=1}^{m}\sum_{k=1}^{N}\log p(w_k = mask|w_1, \cdots, w_{k-1}, w_{k+1}, \cdots, w_{N_i}; \Theta),$$

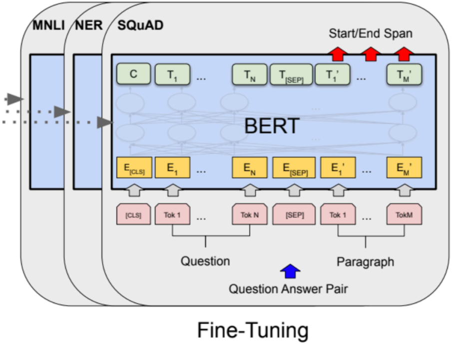
Task2 的 loss function: (假設分成 m 對）
document=$\{s_{1a}, s_{1b}, \cdots, s_{ma}, s_{mb}\}$:

$$\mathcal{L}_2 = \frac{1}{m}\sum_{k=1}^{m}\log p(y|s_{ka}, s_{kb})$$

The model is trained by the total loss

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$$

Down Stream Task



Down Stream Task

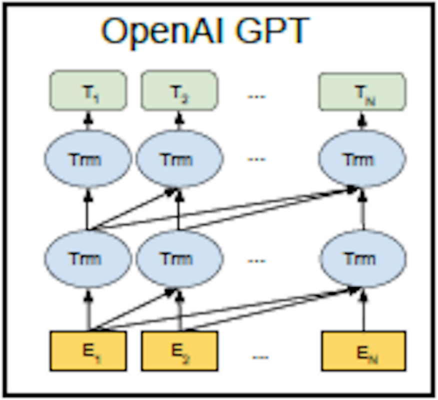
- Parameters transfer+ classifying layer
- Task: Text classification: NER, POS
- Task: Sentence Prediction: QA, NLI
  A: question, B: passage, S: start token embedding, E: end token
  - 第一個字：$\mathrm{argmax}_j P_i$, where $P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$: the probability of word i being the start of the answer span.
  - 接下來依序下個字的計算方式：The score of a candidate span from position i to position j is defined as $S \cdot T_i + E \cdot T_j$, and the maximum scoring span where $j \geq i$ is used as a prediction.
  - 最後一個字：$P_i = \frac{e^{E \cdot T_i}}{\sum_j e^{E \cdot T_j}}$ 最高的那個
- The training objective is the sum of the log-likelihoods of the correct start and end positions. (中間不算嗎?)

CoreNLP

Stanford CoreNLP

GPT

論文：Improving Language Understanding by Generative Pre-Training
使用 Transformer 的 decoder



Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, ..., u_n\}$

- Architecture:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_\ell &= \text{transformer}_{\text{block}}(h_{\ell-1}) \quad \forall i \in [1, n] \\ p(x) &= \text{softmax}(h_n W_e^T) \end{aligned}$$

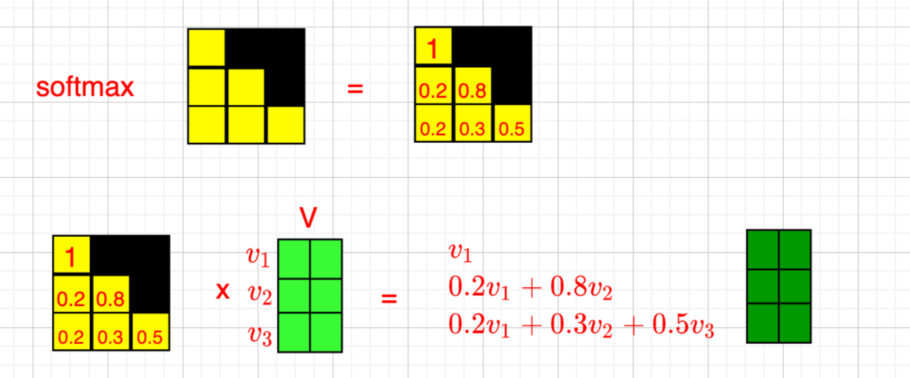
  where $U = (u_{-k}, ..., u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix and W_p is the position embedding matrix.

- Minimizing

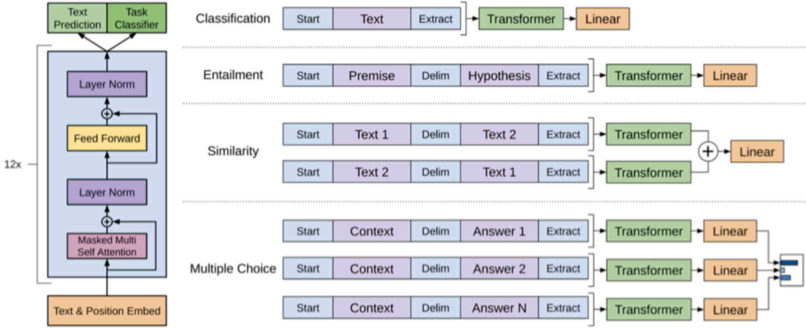
$$L_1(\mathcal{U}) = -\log p(u_i | u_{i-k}, \cdots, u_{i-1}; \Theta),$$

  where k is the size of the context window, and the conditional probability p is modeled using a neural network with parameters Θ.

Masked Attention



GPT Task



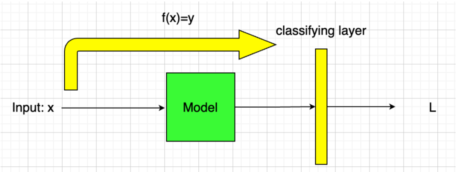
Text Prediction
Task Classifier
Layer Norm
Feed Forward
Layer Norm
Masked Multi Self Attention
12x
Text & Position Embed
Classification
Start
Text
Extract
Transformer
Linear
Entailment
Start
Premise
Delim
Hypothesis
Extract
Transformer
Linear
Similarity
Start
Text 1
Delim
Text 2
Extract
Transformer
Start
Text 2
Delim
Text 1
Extract
Transformer
Linear
Multiple Choice
Start
Context
Delim
Answer 1
Extract
Transformer
Linear
Start
Context
Delim
Answer 2
Extract
Transformer
Linear
Start
Context
Delim
Answer N
Extract
Transformer
Linear

OpenAI playgound

OpenAI

Classification

- Text Classification
- Image Classification



- Model output feature $f(\mathbf{x}) = \mathbf{z} \in \mathbb{R}^d$
- Classifying Layer $\hat{\mathbf{y}} = W\mathbf{z} + \mathbf{b}$, $\mathbf{p} = \mathrm{softmax}(\hat{\mathbf{y}})$, $W \in \mathbb{R}^{C \times d}$
  or $\mathbf{p} = \sigma(\hat{\mathbf{y}}) = (\sigma(\hat{y}_1), \cdots, \sigma(\hat{y}_C))^T$

Metric of Performance

- Binary classification: Confusion matrix

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

- Recall: $\frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$
  亦稱：sensitivity; hit rate, true positive rate
- Precision $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$
- Accuracy$=\frac{\mathrm{TP+TN}}{\mathrm{TP+FN+TN+FP}}$

F scores

$$F_\beta = (1+\beta^2)\frac{p\cdot r}{\beta^2 p + r}$$

Proof.

$$\frac{\partial F_\beta}{\partial \beta} = \frac{2\beta p r^2}{(\beta^2 p + r)^2} > 0 \quad \text{if } \beta > 0.$$

Therefore F_β is increasing. Furthermore, if $r > p$, then

$$\frac{\beta^2}{r} + \frac{1}{r} < \frac{\beta^2}{r} + \frac{1}{p} < \frac{\beta^2}{p} + \frac{1}{p}$$

Thus their inverse have the following relation

$$r = \frac{1+\beta^2}{\frac{\beta^2}{r} + \frac{1}{r}} > \frac{1+\beta^2}{\frac{\beta^2}{r} + \frac{1}{p}} = F_\beta > \frac{1+\beta^2}{\frac{\beta^2}{p} + \frac{1}{p}} = p.$$

也就是 $p < F_{1/2} < F_1 < F_2 < \cdots < r$ $\square$

Multiple Classification

Confusion matrix

	P_1	P_2	P_3
T_1	a	b	c
T_2	d	e	f
T_3	g	h	i

- 若有 none，假設 class 1=none
  - Recall $r = \frac{e+f}{d+e+f+g+h+i}$
  - Precision $p = \frac{e+f}{b+e+h+c+f+i}$
- 若沒有 none，分別統計

  $r_1 = \frac{a}{a+b+c}$, $p_1 = \frac{a}{a+d+g}$, $r_2 = \frac{e}{d+e+f}$, $p_2 = \frac{e}{b+e+h}$, $r_3 = \frac{i}{g+h+i}$, $p_3 = \frac{i}{c+f+i}$

F scores for multiple classifications

Compute

$$p_i = \frac{TP_i}{TP_i + FP_i} \quad r_i = \frac{TP_i}{TP_i + FN_i}$$

for each class

- Micro F1:
  precision $p = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} TP_i + \sum_{i=1}^{C} FP_i}$, and recall $r = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} TP_i + \sum_{i=1}^{C} FN_i}$.
  Notice that $p = r$. Micro $F_1 = p = r$
- F scores for each class

$$F_{\beta,i} = (1 + \beta^2)\frac{p_i \cdot r_i}{\beta^2 p_i + r_i}$$

  Macro $F_\beta = \frac{1}{C}\sum_{i=1}^{C} F_{\beta,i}$
- 類別不平衡時，Macro F scores 較公平
- Accuracy$=\frac{\sum_{i=1}^{C} TP_i}{N}$

Good metric?

評估指標依產業而不同

- $F_1 = \frac{2}{\frac{1}{r}+\frac{1}{p}}$ harmonic mean，均衡 recall 與 precision。一般分類任務採用 F_1
- Image classification: The top-1, top-k error
  Top-1 error= 沒猜中的比例（1-accuracy），Top-5 error=GT 不在前五名猜測範圍。
- 與安全性相關的產業：醫療、指紋辨識、人臉辨識解鎖
  高 precision
- 廣告、關鍵字
  高 recall
- 翻譯任務：
  - Perplexity $2^{-\ell}$, $\ell = \frac{1}{N}\sum_{i=1}^{N}\log(w_i|w_{i-1})$

機器翻譯指標:BLEU(Bilingual Evaluation Understudy) score

- $$BLEU = BP \cdot (\Pi_{i=1}^{4} p_i)^{1/4},$$

  where

  $$p_i = \frac{\sum_{snt \in \mathrm{Cand-Corpus}} \sum_{i \in snt} \min(m^i_{cand}, m^i_{ref})}{w^i_t = \sum_{snt' \in \mathrm{Cand-Corpus}} \sum_{i' \in snt'} \min(m^i_{snt'}, m^i_{cand})}$$

- m^i_{cand} is the count of i-gram in candidate matching the reference translation
- m^i_{ref} is the count of i-gram in the reference translation
- w^i_t is the total number of i-grams in candidate translation
- $BP = \min(1, \exp(1 - rl/ol))$: Brevity penalty (懲罰因子) ol(output-length= 機器譯文長度，rl (reference-length)= 參考翻譯長度
  - BLEU 需要計算翻譯後 p_n: n-gram 精確率
  - BP：若翻譯後長度小於參考譯文，則 BP 小於 1
  - BLEU 的 1-gram 精確率表示譯文終於原文 (loss)，其他 n-gram 表示翻譯流暢程度

假設

Reference: the cat is on the mat

Candidate: the the the cat mat (機器翻譯)

Unigram	m^1_{cand}	m^1_{ref}	$\min(m^1_{cand}, m^1_{ref})$
the	3	2	2
cat	1	1	1
is	0	1	0
on	0	1	0
mat	1	1	1

$w^1_t = 5,\ p_1 = 2 + 1 + 1/5 = 0.8,\ BP = \min(1, e^{1-6/5}) = e^{-0.2}$

2-gram precision

2-gram	m^2_{cand}	m^2_{ref}	$\min(m^2_{cand}, m^2_{ref})$
the cat	1	1	1
cat is	0	1	0
is on	0	1	0
on the	0	1	0
the mat	0	1	0

$w^2_t = 5, \; p_2 = 1/5 = 0.2$

3-gram precision

3-gram	m^3_{cand}	m^3_{ref}	$\min(m^3_{cand}, m^3_{ref})$
the cat is	0	1	0
cat is on	0	1	0
is on the	0	1	0
on the mat	0	1	0

$w^3_t = 4,\ p_3 = 0/5 = 0$
BLEU=0

作業五

Calculating the BLEU score

Reference: The NASA Opportunity rover is battling a massive dust storm on Mars.

Candidate 1: The Opportunity rover is combating a big sandstorm on Mars.

Candidate 2: A NASA rover is fighting a massive storm on Mars.

作業六

- 找一個 dataset 打造一個以 RNN 為基礎，做情意分析
- 炎龍老師 Mooc 課程 https://github.com/yenlung/Deep-Learning-MOOC/blob/master/04-1.%20 RNN .ipynb
- 選一個自己喜歡的 dataset，或者IMDB
- 觀察 dateset，正樣本多？還是負樣本多？有無資料不平衡問題？如何解決？
- 模型：RNN, LSTM, BERT, GPT? 幾層？hidden dimension 多少才適合？
- Loss 的選擇
- 訓練到什麼時候才算訓練好？評判指標是什麼
- 準確率如何？採用何種指標？是適合這個 dataset 的評判嗎？

Binary Cross-Entropy

The binary cross entropy is

$$L = -\left(y \log p + (1-y)\log(1-p)\right)$$

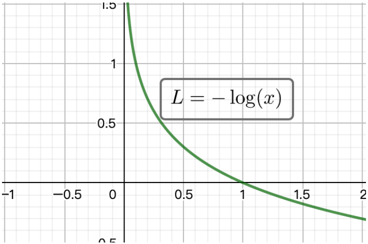
where $p \in [0,1] \subset \mathbb{R}^1$ and $y \in \{0,1\}$.
Categories cross-entropy is given by

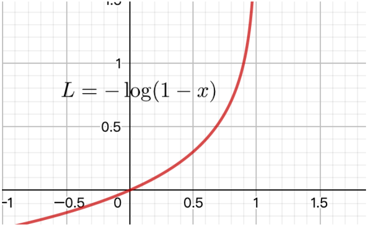
$$L = -\mathbf{y} \log \mathbf{p}$$

Suppose there are two categories. That is, $\mathbf{y}, \mathbf{p} \in \mathbb{R}^2$, where $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ and $\mathbf{p} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$. Since $\mathbf{y}$ is the ground truth and $\mathbf{p}$ represents the probability of predicts, we have $y_1 + y_2 = 1$ and $p_1 + p_2 = 1$. Then

$$L = -\mathbf{y}\log\mathbf{p} = -(y_1 \log p_1 + y_2 \log p_2) = -(y_1 \log p_1 + (1-y_1)\log(1-p_1))$$

Loss

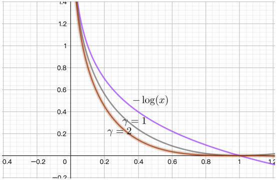




Focal Loss

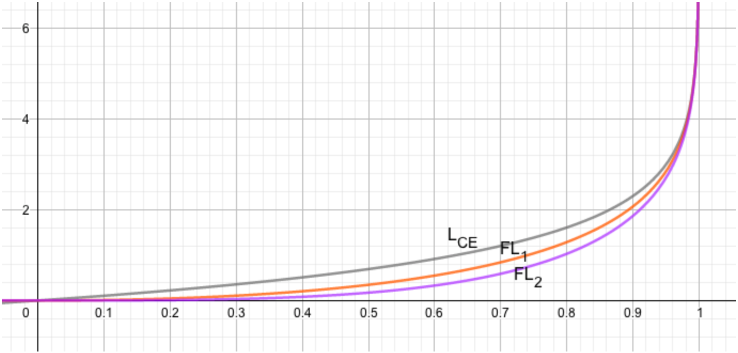
論文：Focal Loss for Dense Object Detection

$$FL = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i (1-\mathbf{p}_i)^{\gamma} \log(\mathbf{p}_i)$$



	分錯	分對但預測差	分很對
Cross Entropy	$\frac{L_{CE}(0.4)}{L_{CE}(0.8)} = 4.11$	$\frac{L_{CE}(0.5)}{L_{CE}(0.9)} = 6.58$	$\frac{L_{CE}(0.8)}{L_{CE}(0.9)} = 2.12$
Focal Loss(1)	$\frac{L_{FL1}(0.4)}{L_{FL1}(0.8)} = 12.32$	$\frac{L_{FL1}(0.5)}{L_{FL1}(0.9)} = 32.89$	$\frac{L_{FL1}(0.8)}{L_{FL1}(0.9)} = 4.24$
Focal Loss(2)	$\frac{L_{FL2}(0.4)}{L_{FL2}(0.8)} = 36.96$	$\frac{L_{FL2}(0.5)}{L_{FL2}(0.9)} = 164.47$	$\frac{L_{FL2}(0.8)}{L_{FL2}(0.9)} = 8.47$

實際值 $y=0$ 時：



	分錯	分對但預測差	分很對
Cross Entropy	$\frac{L_{CE}(0.6)}{L_{CE}(0.2)}=4.11$	$\frac{L_{CE}(0.49)}{L_{CE}(0.1)}=6.39$	$\frac{L_{CE}(0.2)}{L_{CE}(0.1)}=2.12$
Focal Loss(1)	$\frac{L_{FL1}(0.6)}{L_{FL1}(0.2)}=12.32$	$\frac{L_{FL1}(0.49)}{L_{FL1}(0.1)}=31.32$	$\frac{L_{FL1}(0.2)}{L_{FL1}(0.1)}=4.24$
Focal Loss(2)	$\frac{L_{FL2}(0.6)}{L_{FL2}(0.2)}=36.96$	$\frac{L_{FL2}(0.5)}{L_{FL2}(0.9)}=153.44$	$\frac{L_{FL2}(0.2)}{L_{FL2}(0.1)}=8.47$

Focal Loss 特色

- 著重在處理預測差的問題 (hard, misclassified examples)
- 類別不均衡問題：通常 object detection 採用 OHEH
- α-balanced variant

$$FL(p_t) = -\alpha(1 - p_t)^\gamma \log(p_t)$$

default $\gamma = 2$, $\alpha = 0.25$ (γ 增加，α 減少). In practice, α may be set by inverse class frequency or treated as a hyperparameter to set by cross-validation.