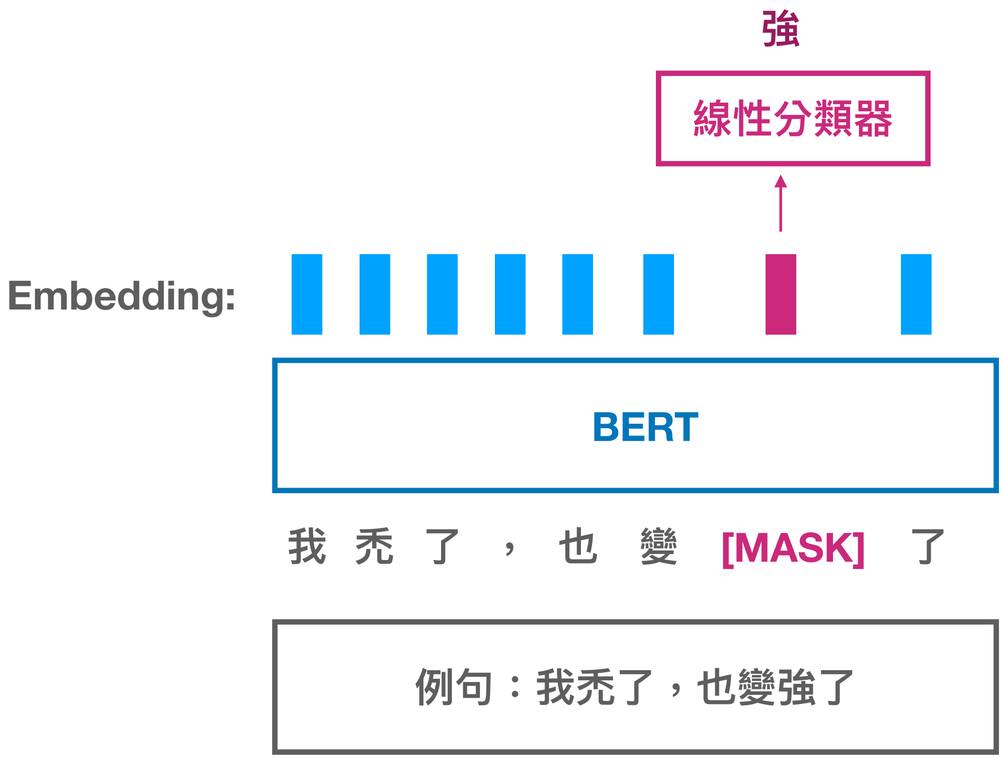
BERT預訓練

上一章節我們提到，BERT 其實就是把 Transformer 的 Encoder 部分拿出來，並且佛心的幫我們做好『預訓練』，那 BERT 的預訓練到底如何做的呢？

BERT訓練1

BERT預訓練方式有兩種，第一種我們叫做 『Masked LM』(Masked Language Model/MLM)，簡單來說就是『克漏字』測驗啦！再簡單一點來說就是 Word2Vec 的訓練方式



MASK

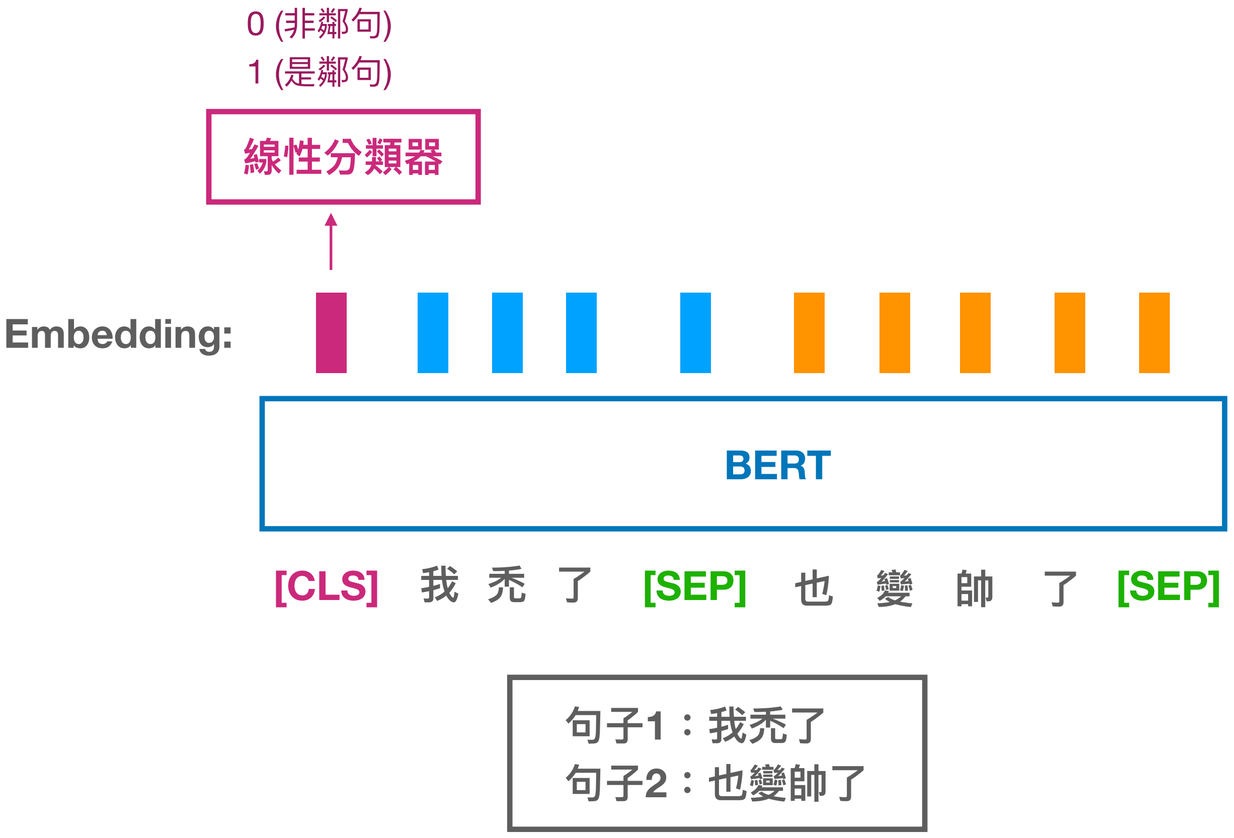
這裡和 word2vec 唯一的差別就是我們定義了個特別的 token， [MASK] 而且我們會挑選文章中 『固定趴數』的 字詞替換成 [MASK] ，被替換的字詞就必須通過線性分類器預測遮蔽詞是什麼

線性分類器

這裡特別的是，我們在預測的時候會選一個『弱分類器』，其實就是一個『線性分類器』，為什麼不選好一點呢？因為我們希望 BERT 得到的 Attention 向量是最好的，即便用弱一點的分類器也能正確預

BERT訓練2

第二種我們叫做 『Next Sentence Prediction』 ，剛剛 MLM 我們考慮『詞的關係』來做出向量，現在我們考慮『句子的關係』來做出向量



句子1：我禿了
句子2：也變帥了

[CLS]

[CLS] 你可以想像成『總體向量』，由於我們是 Attention機制，所以你可以想像成這個 Query，就是針對所有帶入的詞的 Query，得到的就是關於整段的向量，接著我們用它來做一個關於整段的判斷

[SEP]

[SEP] 是拿來分隔句子的，在 BERT 原始的訓練中，我們帶入input就是兩個句子，輸出就是一個分類

BERT使用姿勢

這裡我們配合 Keras 的易於使用，我們使用 keras-bert

https://github.com/CyberZHG/keras-bert

預訓練的模型，我們去 Google BERT 的 github裡找

https://github.com/google-research/bert

The links to the models are here (right-click, 'Save link as...' on the name):

- `BERT-Large, Uncased (Whole Word Masking)` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Large, Cased (Whole Word Masking)` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Uncased` : 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Large, Uncased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Cased` : 12-layer, 768-hidden, 12-heads , 110M parameters
- `BERT-Large, Cased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Multilingual Cased (New, recommended)` : 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Multilingual Uncased (Orig, not recommended)` **(Not recommended, use** `Multilingual Cased` **instead)**: 102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Chinese` : Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters

模型挑選

The links to the models are here (right-click, 'Save link as...' on the name):

- `BERT-Large, Uncased (Whole Word Masking)` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Large, Cased (Whole Word Masking)` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Uncased` : 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Large, Uncased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Cased` : 12-layer, 768-hidden, 12-heads , 110M parameters
- `BERT-Large, Cased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Multilingual Cased (New, recommended)` : 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Multilingual Uncased (Orig, not recommended)` **(Not recommended, use** `Multilingual Cased` **instead)**: 102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Chinese` : Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters

Base or Large

差在層數的多寡，如果需要更準確的正確率，就選 Large，不過通常我們為了比較快的訓練速度和使用速度，會選擇 Base

Cased or Uncased

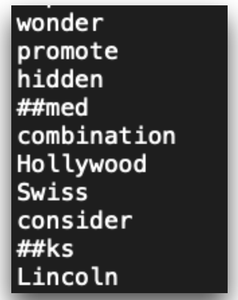
是否視大小寫為不同的詞，如果想將大小寫視為不同的詞就必須選擇 Cased，沒有特別大小寫需求可以選擇 Uncased

中文 / 多語系

中文請選 Chinese，對於中文字彙有更好的收錄，如果需要多語言的分析請選 multilingual

字彙表

要了解 whole word masking，我們就要了解 BERT 的字典是如何構建的，你可以從下載下來模型裡的 vocab.txt 裡面看



vocab.txt

開始為 ## 的不是一個詞，而是一個 subword

避免 OOV 問題的另外一個方法就是把有意義的subword分開

這樣分解的話，我們還可以大量的減少詞彙表的大小

還記得我們說過，要盡量地避免OOV問題吧？

譬如:
loved 變成
##lov ## ed
loves 變成
lov ## es

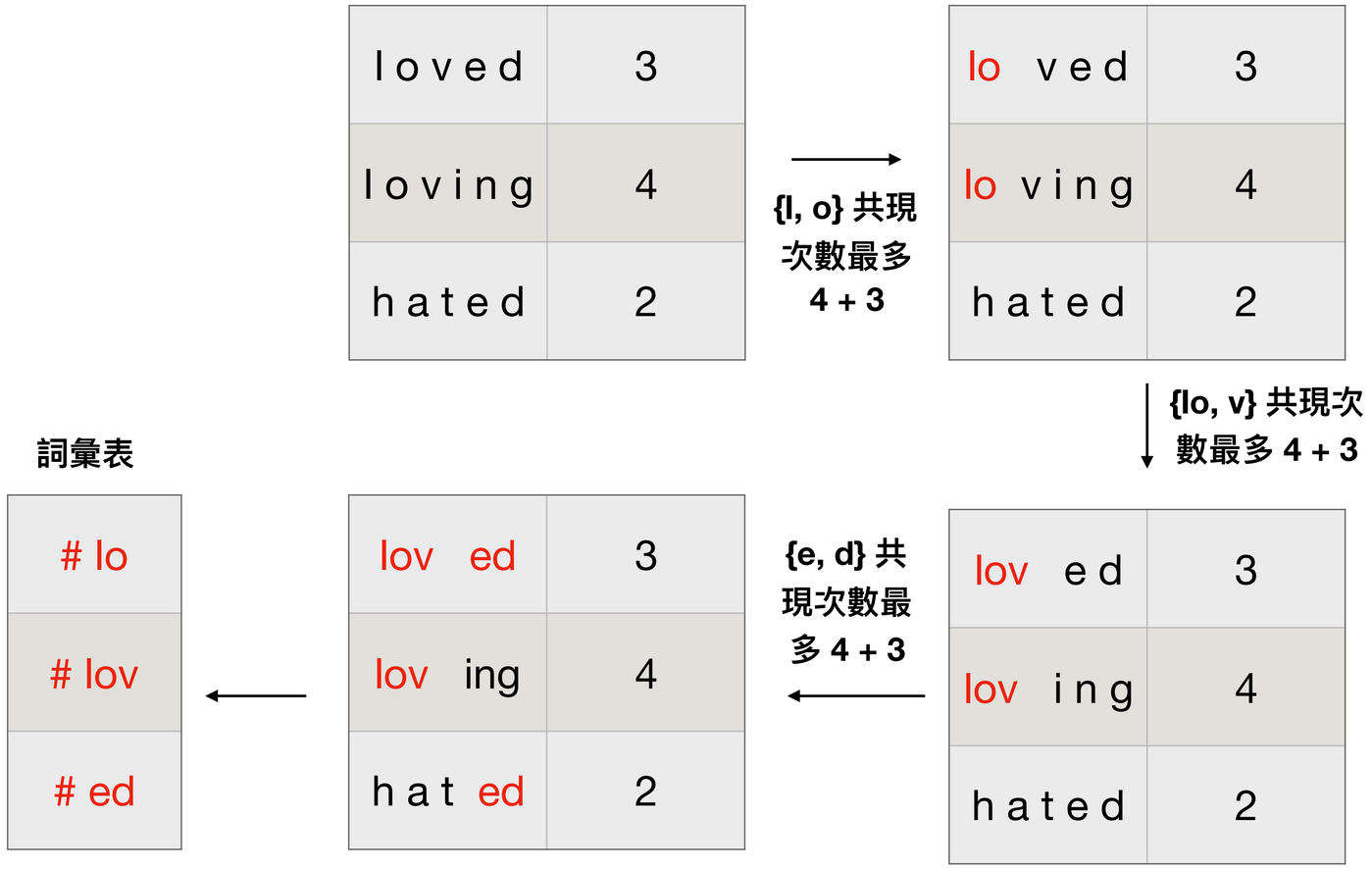
當然中文就沒這麼好康了，因為中文一字就是一個基本單位，所以中文直接以字做切割

Subword挑選

這裡挑選的方式是根據 BPE(Byte-Pair Encoding) 做出的改良，簡單來說就是把同時出現次數很多的字組合變一個subword

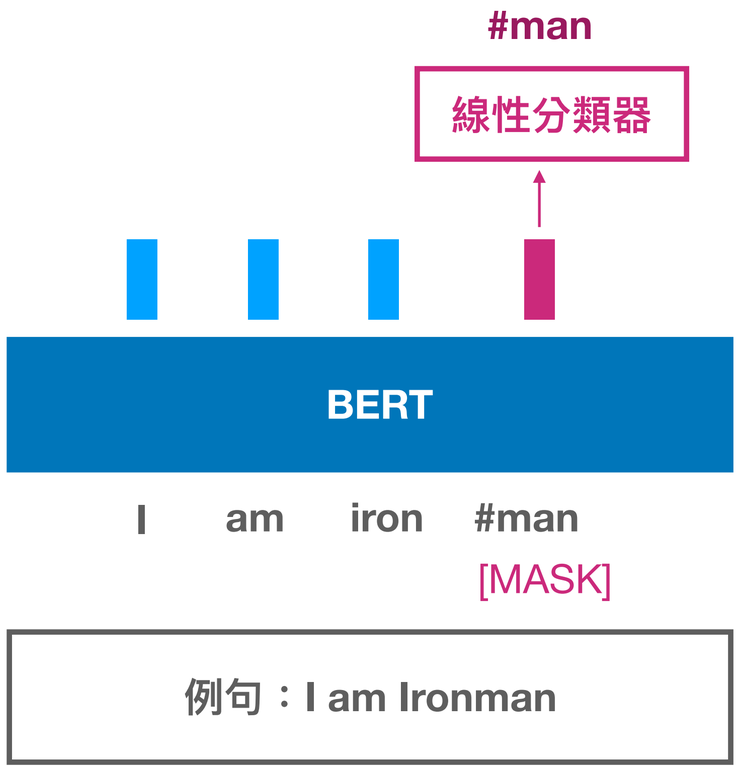
詳細步驟

1. 先把所有的詞彙以字母切割開
2. (BPE) 找出出現次數最多的 bi-word
   (BERT) 找出加到模型裡可以讓準確率提升最高的 bi-word
3. 直到字彙表達到設定的最大容量或者 (BPE) 再也找不到任何共現出現次數 > 1 (BERT) 提升率沒超過 threshold 時候停止

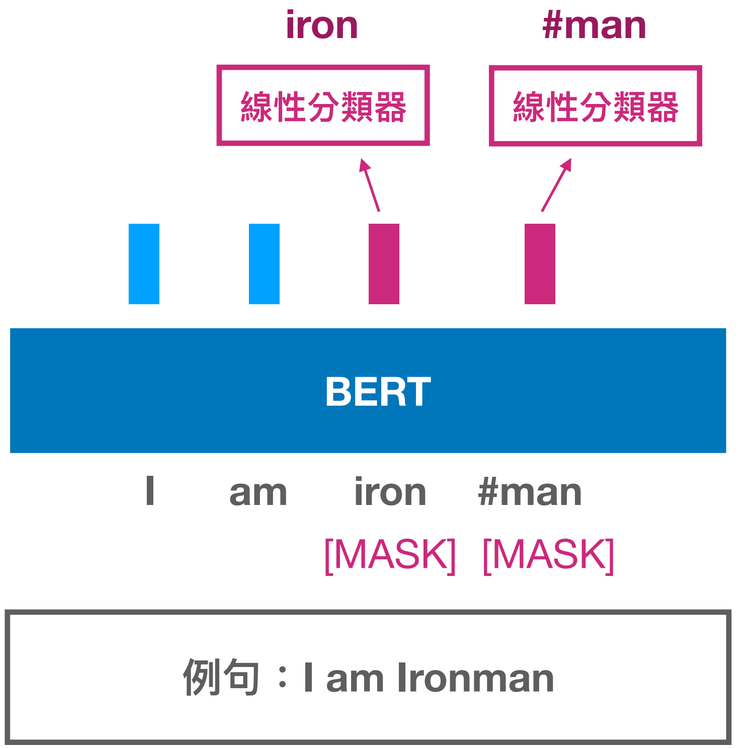


Whole Word Masking

沒有 whole word masking 太好猜，因為你有旁邊的提示，這樣詞向量的訓練會不精準，於是我們把同一個詞一起 MASK，中文做出這改動也會有很大的幫助！



沒有 whole word masking

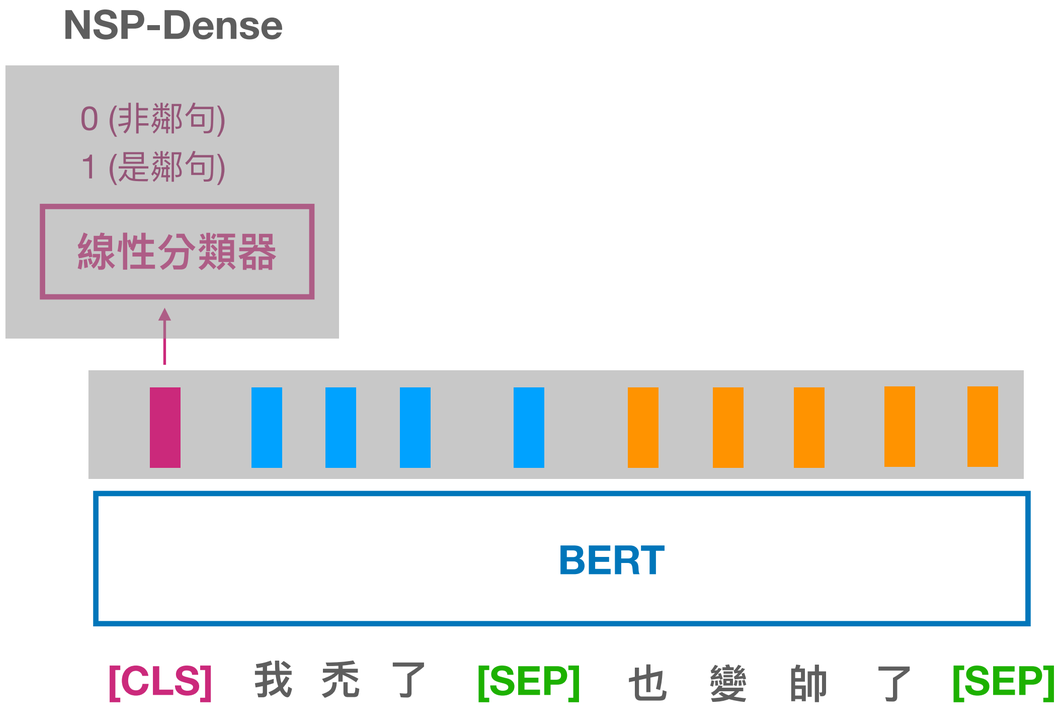


有 whole word masking

BERT輸入

我們的 BERT 總共會有 三個你可以調整的輸入
(位置編碼不算進去，固定不調)

BERT輸出



拿取每一個詞的Attention向量

keras-bert: Training = False

如果你想要以詞為 Level 加上你的下游任務，你就可以直接拿每一個詞的向量

拿取整個輸入的 Attention

keras-bert: Training = True

如果你想要以整個輸入的向量作為下游任務，建議只要是『帶入兩段的任務』的時候必須使用，例如：『問答系統』，判定問題和答案是不是一個配對