題目：conversations in TV shows

隊伍名稱：NTU_b03201031_K

成員：數學四，王楷

工作分工：獨自完成

一、 Preprocessing/Feature Engineering：

首先，關於 training data，設定 stopwords，及想要濾掉的標點符號及字，例如：” A:” 、” 「” …，將這些 stopwords 過濾掉。

關於 testing data，亦做上述相同的步驟，而若對話中有不同人的對話，則把人物符號及空白去掉，併做同一個句子。

之後利用 Jieba 套件做斷詞，Jieba 載入的辭典為 Jieba 官方提供的繁體辭典。

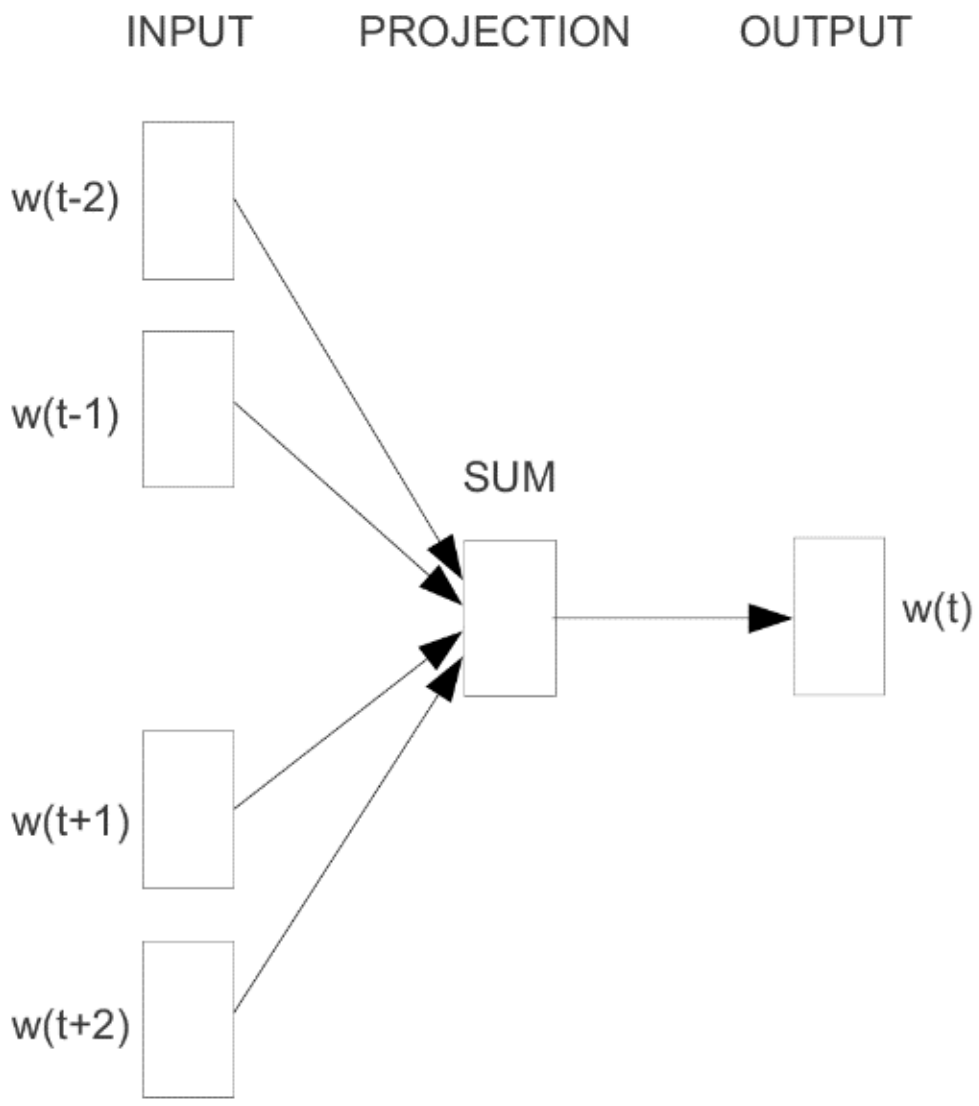
而當在做不同實驗時，利用 Jieba 套件中不同的模式，例如全模式、一般模式、搜尋引擎模式，產生不同的斷詞結果，進而達到不同的實驗效果。

二、 Model Description

在做實驗時，主要利用兩種 Word embedding Model，再搭配不同的 data preprocessing 做 training，這裡 word embedding 的 model，是用 gensim 中提供的 word2vec 來做訓練，最終將句子中的 word vector 相加，以 cosine 值判斷相似度，而 word embedding 的模型包

含了 CBOW 以及，skip-gram 兩種不同的模型。

第一種，word2vec 參數，iter=20, mincount=1, sg=0，模型架構圖如下:

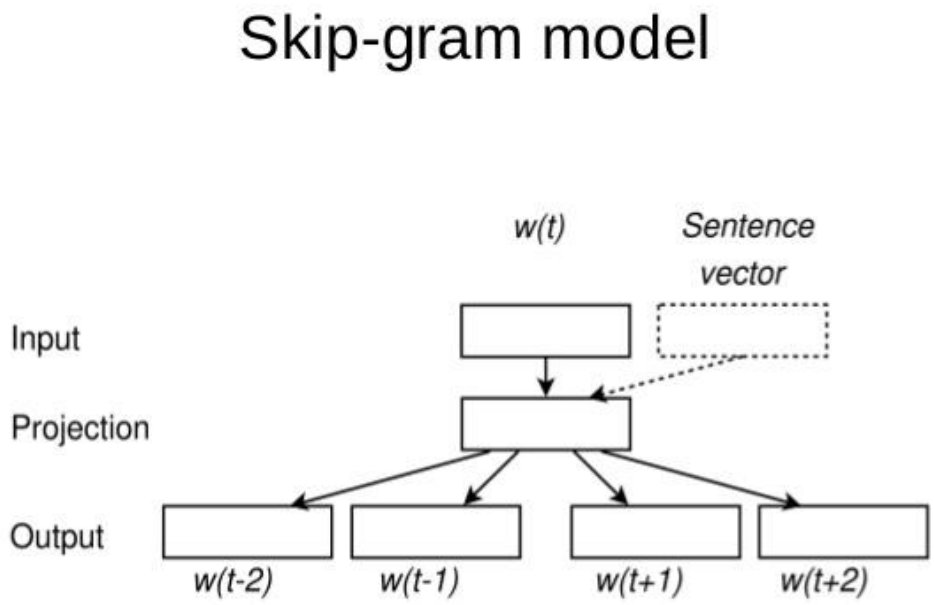


CBOW

此模型為一個三層 NN 結構，模型的第一層為輸入層，輸入已知上下文的詞向量，中間層為穎含層，輸入詞向量的累加，第三層是一個樹狀結

構，樹的 leaves 與語料庫中的單詞一一對應，且將所有出現過一次的單字皆當作 training data，做20次的 iteration

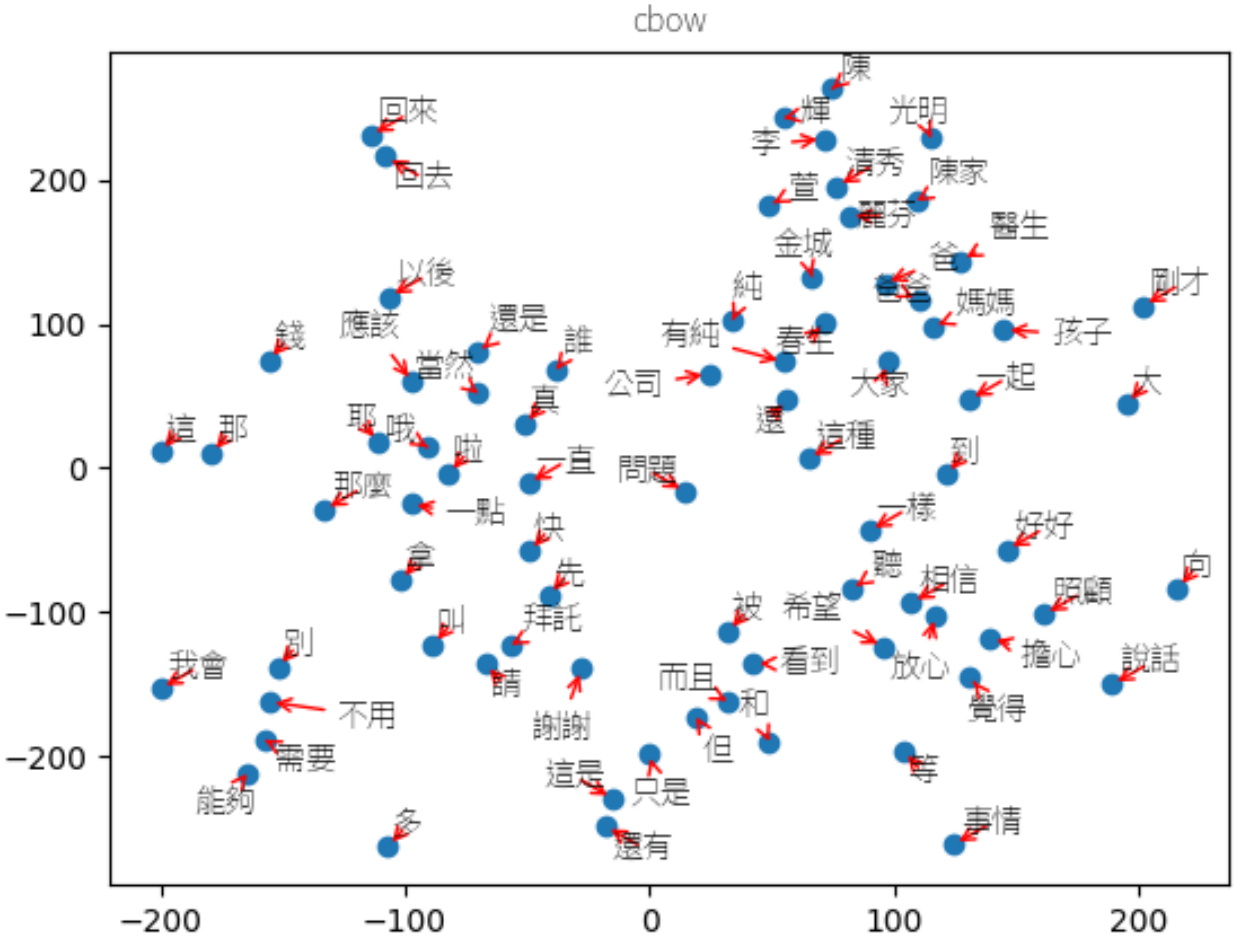
第二種，Skip-gram，word2vec 參數，iter=20, mincount=1, sg=1，架構圖如下:

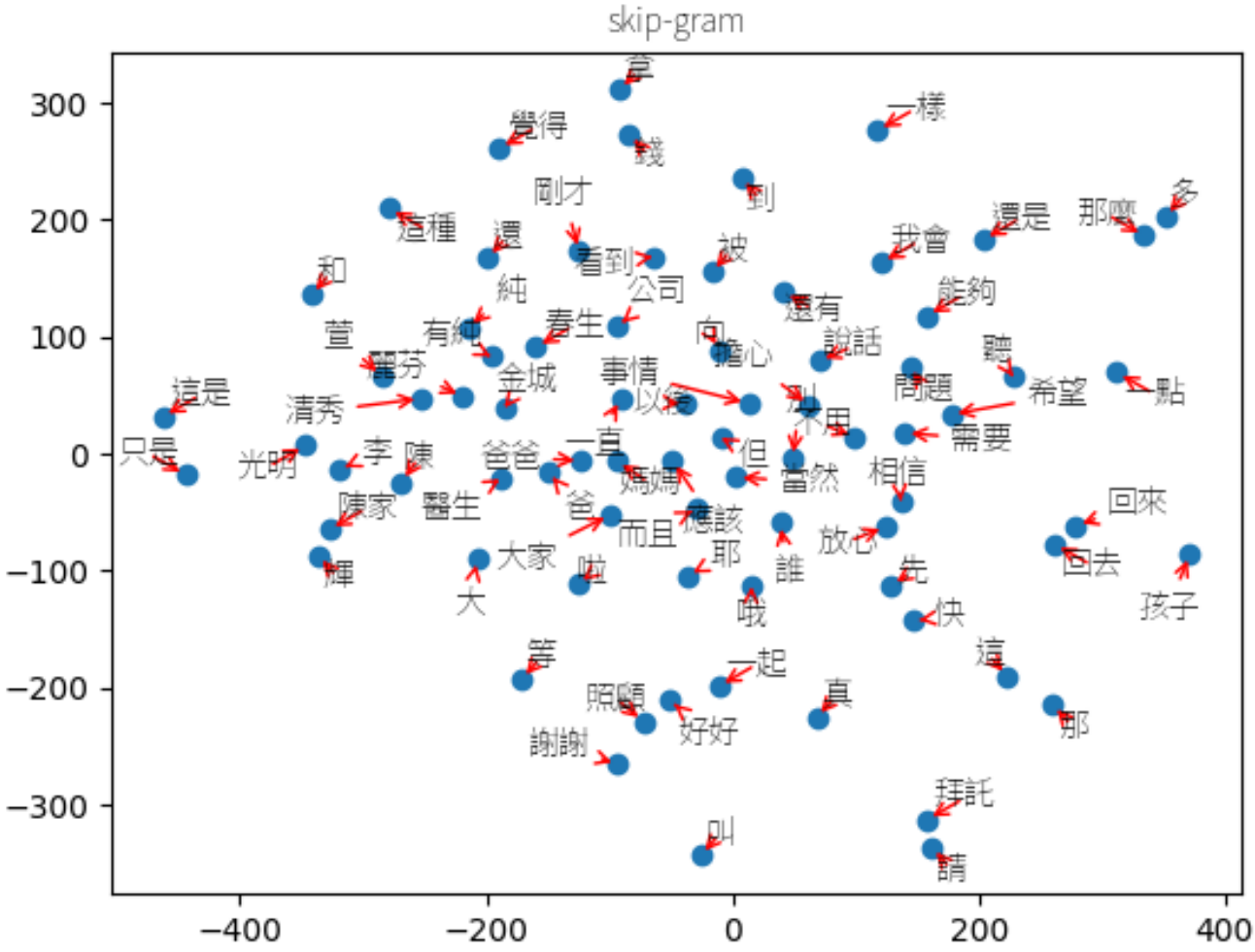


此模型亦為一個三層 NN 結構，樹入一個單字，輸出隊上下文的預測，此結構的核心亦為樹狀結構，其中把所有出現過一次的單字皆當作 training data，做20次 iteration

三、 實驗

a. 在同樣的 preprocess 方法下，對 skip-gram 模型與 CBOW 模型做 TSNE 降維，觀察 visualization 後的差異





單看視覺化效果的話，CBOW 的效果比 skip gram 明顯，但兩者皆有呈現出先觀程度較高的詞語會較靠近的效果，有可能是因為只取出現次數介於3000到6000的詞語，所以細節差異表現不明顯。

b. 觀察 CBOW 與 Skip-gram 的準確率差異

在 iter=20, dim=256的情況下，

CBOW 的準確率為30%

Skip-gram 的準確率為46%

在 iter=20, dim=192

CBOW 的準確率為32%

Skip-gram 的準確率為45%

由此可知，在此問題中 skip-gram 模型所得準確率會比 CBOW 模型高。

c. 在不同的 preprocessing 方法下，觀察 skip-gram 模型對不一樣的 preprocessing 所產生的準確率變化

此處 word2vec 的參數皆固定為 dim=256, iter=20, mincount=1, sg=1

若 Traing data 與 Testing data 相同模式

1. 全模式，準確率 40.8%
2. 正常模式，準確率46%
3. 搜尋引擎模式 ，準確率 41%

d. 觀察 skip-gram 模型在不同的 iteration 次數所產生準確率的變化

此處 word2vec 的參數皆固定為 dim=256, mincount=1, sg=1

Training data 與 testing data preprocessing 的方法皆為正常模式

1. Iteration = 20，準確率46.6%

2. Iteration =30，準確率 45.8%
3. Iteration =40，準確率 46.2%
4. Iteration = 50，準確率 45.8%

由此可知，iteration 越多未必能得到越好的 performance，在此 case 中，最佳的 iteration 次數約為

e. 觀察 skip-gram 模型在不同的 embedding dimension 所產生的準確率變化

此處 word2vec 的參數皆固定為 iter=20, mincount=1, sg=1

1. dim = 64，準確率為 45.5%
2. dim = 128，準確率為 43.5%
3. dim = 192，準確率為45.8%
4. dim = 256，準確率為46.6%
5. dim = 320，準確率為 46.4%

從這邊可以觀察到，dim=64時為 underfitting，dim = 320為 overfitting，最適合的 dimension 應介於200~300之間

f. 比較當把 word vector 相加時有無把單字出現頻率考慮進去

此處此處 word2vec 的參數皆固定為 dim =256, iter=20, mincount=1, sg=1

1. 不考慮時，準確率為46%
2. 考慮時，準確率為30%

由此可知 frequency 對準確率可能造成負面的影響，個人猜測原因為有些出現次數非常頻繁，但實際並無太大意義的單字，例如:就、為...等會因為過多的權重影響實際的準確率。