Data Mining Lab2

Kaggle competition report

- **Data preparation (code in “Second_part_data_preparation.ipynb”)**

首先在”tweets_DM.json”裡，我只保留”source”的部分，圖為其中一筆。


```
{'_score': 391,
 '_index': 'hashtag_tweets',
'_source': {'tweet': {'hashtags': ['Snapchat'],
  'tweet_id': '0x376b20',
  'text': 'People who post "add me on #Snapchat" must be dehydrated. Cuz man.... that\'s <LH>'}},
 '_crawldate': '2015-05-23 11:42:47',
 '_type': 'tweets'}
```


另外在”source”當中，也只保留”tweet_id”, “hashtags”, “text”的部分。

	hashtags	tweet_id	text
0	[Snapchat]	0x376b20	People who post "add me on #Snapchat" must be ...
1	[freepress, TrumpLegacy, CNN]	0x2d5350	@brianklaas As we see, Trump is dangerous to #...
2	[bibleverse]	0x28b412	Confident of your obedience, I write to you, k...
3	[]	0x1cd5b0	Now ISSA is stalking Tasha 😂😂😂 <LH>
4	[]	0x2de201	"Trust is not the same as faith. A friend is s...
...	...	...	...
1867530	[mixedfeeling, butimTHATperson]	0x316b80	When you buy the last 2 tickets remaining for ...
1867531	[]	0x29d0cb	I swear all this hard work gone pay off one da...
1867532	[]	0x2a6a4f	@Parcel2Go no card left when I wasn't in so I ...
1867533	[]	0x24faed	Ah, corporate life, where you can date <LH> us...
1867534	[Sundayvibes]	0x34be8c	Blessed to be living #Sundayvibes <LH>

透過”tweet_id”，利用 data_identification.csv 把資料分成 training set 和 test set。

	hashtags	tweet_id	text	identification
0	[Snapchat]	0x376b20	People who post "add me on #Snapchat" must be ...	train
1	[freepress, TrumpLegacy, CNN]	0x2d5350	@brianklaas As we see, Trump is dangerous to #...	train
2	[bibleverse]	0x28b412	Confident of your obedience, I write to you, k...	test
3	[]	0x1cd5b0	Now ISSA is stalking Tasha 😂😂😂 <LH>	train
4	[]	0x2de201	"Trust is not the same as faith. A friend is s...	test
...	...	...	...	...
1867530	[mixedfeeling, butimTHATperson]	0x316b80	When you buy the last 2 tickets remaining for ...	test
1867531	[]	0x29d0cb	I swear all this hard work gone pay off one da...	test
1867532	[]	0x2a6a4f	@Parcel2Go no card left when I wasn't in so I ...	test
1867533	[]	0x24faed	Ah, corporate life, where you can date <LH> us...	train
1867534	[Sundayvibes]	0x34be8c	Blessed to be living #Sundayvibes <LH>	train

也透過”tweet_id”，將 emotion.csv 併入 training set 裡標上 label。

	hashtags	tweet_id	text	identification	emotion
0	[Snapchat]	0x376b20	People who post "add me on #Snapchat" must be ...	train	anticipation
1	[freepress, TrumpLegacy, CNN]	0x2d5350	@brianklaas As we see, Trump is dangerous to #...	train	sadness
2	[]	0x1cd5b0	Now ISSA is stalking Tasha 😭😭😩 <LH>	train	fear
3	[authentic, LaughOutLoud]	0x1d755c	@RISKshow @TheKevinAllison Thx for the BEST TI...	train	joy
4	[]	0x2c91a8	Still waiting on those supplies Liscus. <LH>	train	anticipation
...	...	...	...	...	...
1455558	[NoWonder, Happy]	0x321566	I'm SO HAPPY!!! #NoWonder the name of this sho...	train	joy
1455559	[]	0x38959e	In every circumtance I'd like to be thankful t...	train	joy
1455560	[blessyou]	0x2cbca6	there's currently two girls walking around the...	train	joy
1455561	[]	0x24faed	Ah, corporate life, where you can date <LH> us...	train	joy
1455562	[Sundayvibes]	0x34be8c	Blessed to be living #Sundayvibes <LH>	train	joy

最後把 training set 和 test set 分別存成 pkl 檔，方便之後讀取。

- **Data preprocess**

在 training set 中，我發現資料有不平均的現象。

```
joy             516017
anticipation    248935
trust           205478
sadness         193437
disgust         139101
fear             63999
surprise         48729
anger            39867
Name: emotion, dtype: int64
```

我一開始有讓每個 **category** 用相同數量去訓練，但表現都變差，因此捨去此種方法。

```
In [14]: sample_train_data = train_data.groupby("emotion").sample(n=39867, random_state=1)

In [15]: sample_train_data.sample(frac=1)
         sample_train_data['emotion'].value_counts()

Out[15]: anger           39867
         anticipation    39867
         disgust         39867
         fear            39867
         joy             39867
         sadness         39867
         surprise        39867
         trust           39867
         Name: emotion, dtype: int64
```

接下來決定 model 的 input 樣式，我有嘗試三種:

1. TFIDF with stemmer processing
2. Self-training W2V
3. Pretrained W2V model (glove-twitter 27B 100d)

- TFIDF with stemmer processing:

Stemmer 可以讓 word 還原成最基本的樣子，例如 hid -> hide, children->child。
先將 sentence tokenize，再把每個 word 丟進 PorterStemmer 進行 stem。

```
porter = PorterStemmer()

def stemSentence(sentence):
    token_words=word_tokenize(sentence)
    token_words
    stem_sentence=[]
    for word in token_words:
        stem_sentence.append(porter.stem(word))
        stem_sentence.append(" ")
    return "".join(stem_sentence)
```

Stem 完後，再利用 TFIDF tokenizer 計算 TFIDF。

- Self-trainging W2V:

我是利用 keras 的 Tokenizer 進行，Tokenizer 會把所有 sentence 看過一遍，整理出由每個 unique word 組合而成的 list，再將 sentence 裡出現過的 word 轉換成對應的 index。

```
Now ISSA is stalking Tasha 😂😂😂 <LH>
[57, 614, 9, 6699, 2493, 892, 1]
```

但每個 sentence 各有不同長度，model 需要統一長度的 input，這邊利用 keras 的 pad_sequences 做 padding。我設定長度為 100，並在長度不滿 100 的句子後面補上 0。

```
[  56   59  572 1096   17   13 1173  292   18 1302  132  220    1    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0]
```

最後在 model 的第一層利用 embedding layer 進行 w2v 的訓練。因為我設 Tokenizer 的 num_word=10000，所以 input 維度為 10000，input_len 為 100，而 output 維度我則是設 50。

```
model.add(layers.Embedding(input_dim=10000,
                           output_dim=50,
                           input_length=maxlen))
```

- Pretrained W2V model (glove-twitter 27B 100d):

我利用 glove 的 glove-twitter 27B 100d 來做 W2V，裡面有高達 27 billion 的 word，而每個 word 會轉換成 100 維度的 vector。

首先讀取 glove 準備好的 txt 並處理成可應用的形式。

embeddings_index 是一個 dict，key 為 word，value 為其轉換後的 100d vectors。

```
embeddings_index = dict()
f = open('./glove/glove.twitter.27B.100d.txt', encoding='utf8')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.array(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
```

接下來先將每個 sentence 做 tokenized，並做 padding，如同之前所述。

最後我們要創造一個可以進行 W2V 的矩陣，row 為 tokenizer 算出來的 word size，column 為 100。依照每個 word，將他對應到的 vector 整理成矩陣。那如果真的沒有對應到 27 billion 裡的任何一個 word，就是 0。

```
# 建造可以轉換為GloVe 100維 詞向量的矩陣
embedding_matrix = np.zeros((vocab_size, 100))
for word, i in t.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

在 model 的 embedding layer 中，因為我們已經使用 pretrained 好的 W2V model，所以 trainable=False，而這邊多一個 weights 參數，就是剛剛整理出來的 W2V 矩陣。

```
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=100,
                           weights=[embedding_matrix],
                           input_length=maxlen,
                           trainable=False))
```

這三種 input 轉變模式，最終是 Self-trainging W2V 通常都會有比較好的結果，因此捨棄另外兩個。

轉變完後，進行 train test split 方便檢視模型表現，test size 為 0.03。
另外也需要對 label 進行 one hot encoding，把每個 label 變成 1*8 的矩陣。

- **Model training**

模型訓練的部分，由於資料集過大，傳統的機器學習方法都沒獲得較好的結果，這邊直接跳至 deep learning 進行。

```
model.add(layers.Embedding(input_dim=10000,
                           output_dim=50,
                           input_length=maxlen))
model.add(layers.Bidirectional(layers.LSTM(40, return_sequences=True)))
model.add(layers.Bidirectional(layers.LSTM(40, return_sequences=True)))
model.add(layers.Bidirectional(layers.LSTM(40)))
model.add(layers.Dense(8, activation='softmax'))
```

```
embedding (Embedding)        (None, 100, 50)           500000

bidirectional (Bidirectiona  (None, 100, 80)           29120
l)

bidirectional_1 (Bidirectio  (None, 100, 80)           38720
nal)

bidirectional_2 (Bidirectio  (None, 80)                38720
nal)

dense (Dense)                (None, 8)                 648
```

我是用**雙向的 LSTM** 進行訓練。LSTM 是一種 RNN 的變化，RNN 的特點是會考慮到循序序列的特徵，他會將 hidden layer 的 output 儲存在 memory，當下一筆進來的時候，可以將這個 output 再拿來一起計算，達到考慮循序關聯的特點。句子是由多個 word 組合而成，每個 word 都是互相關聯的，因此非常適合 RNN 這種模型進行訓練。而雙向 LSTM 是讓模型除了往前掃過句子，也可以將句子往反方向推論，這樣讓模型訓練到每個 word 的前後關係。
當 LSTM 訓練完成，最後套一層 dense network 來分類出 8 個 emotion。

下面為 hyper-parameter 的參數設定

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Epoch 為 20，batch_size 設 10。

- **Conclusion**

在這次的訓練中，有許多較為反常的前處理讓我非常訝異，像是讓模型 input 的 category 用相同數量做計算會有比較差的結果，或是利用 pretrained 好的 W2V model 進行也是，在模型的部分，我也有嘗試加 dropout 層讓 model 不要過度 overfit，表現也是降低，總總跡象讓我覺得其實稍微 overfit 是可以在競賽中取得較好的成績。

另外能夠大幅進步的方法也可以嘗試 BERT 進行訓練，並對 BERT 進行 fine-tune。