AI Final Project

Group members : 0816080許舒茵、0816116黃思瑜、0816123吳應玉
Group name : whisper nigastella
team id:14

Spicy Chatting Robot

我們實作了一個在telegram平台上的聊天機器人，基於NLP先訓練一個可以辨識語句情緒的model，讓使用者可以在telegram上輸入字句，機器人就會根據後台model判斷的情緒關鍵字給出回應。

Dataset

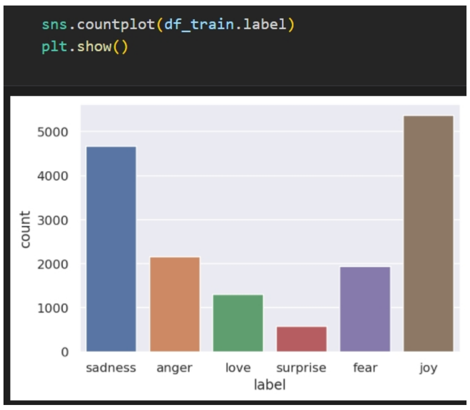
> 有分中英文版本

#1 English version

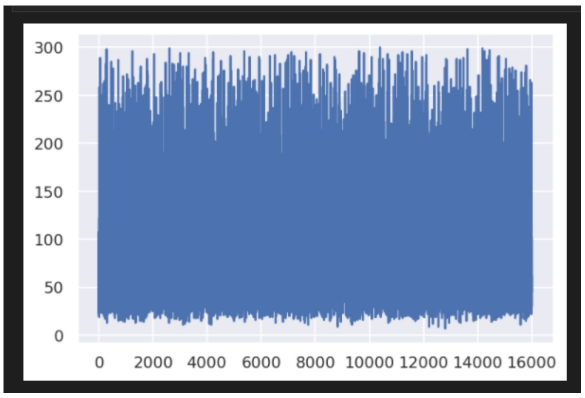
- 來源: Kaggle dataset

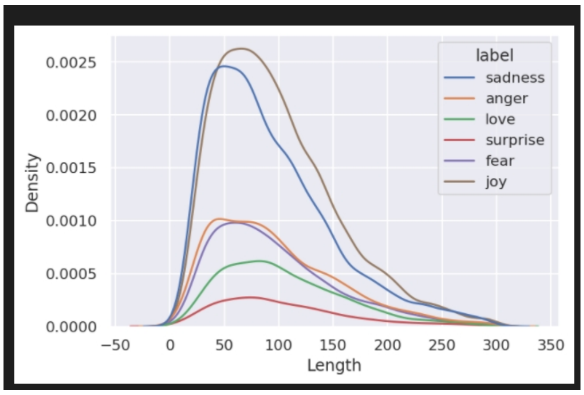
Dataset Analysis

- training data visualization

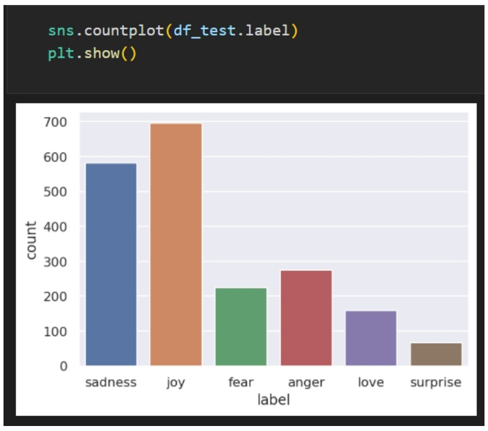


```
sns.countplot(df_train.label)
plt.show()
```

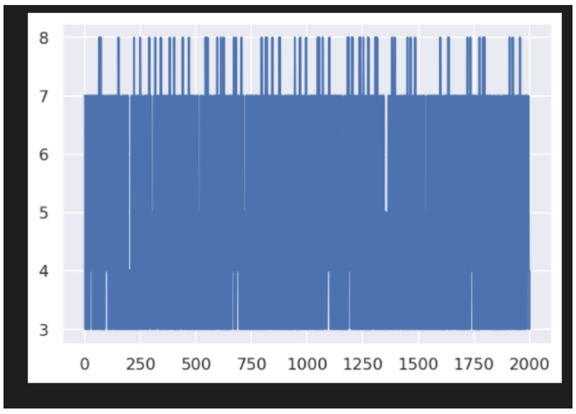




- testing data visualization



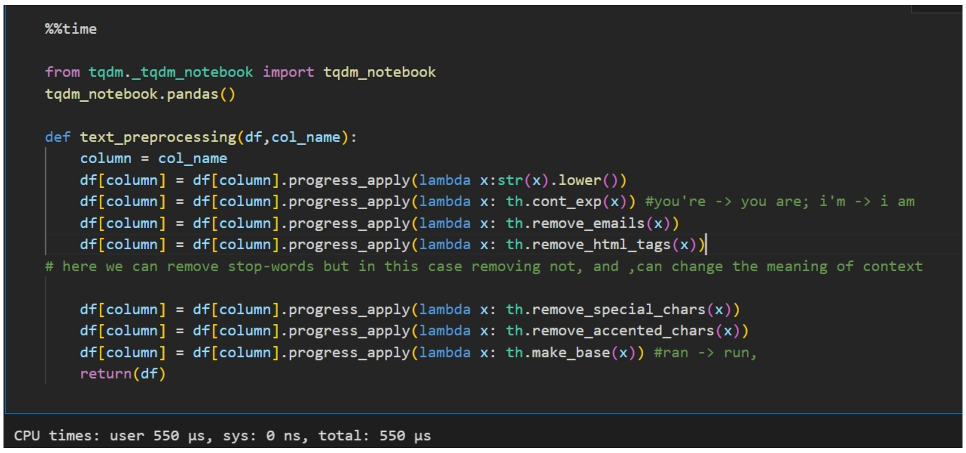
```
sns.countplot(df_test.label)
plt.show()
```



Data preprocessing

text hammer

- 使用`cont_exp` 把標點符號去掉
- 用`remove_special_chars` 等刪除特殊符號
- `make_base`可以把過去式單字等文法轉成統一的現在式



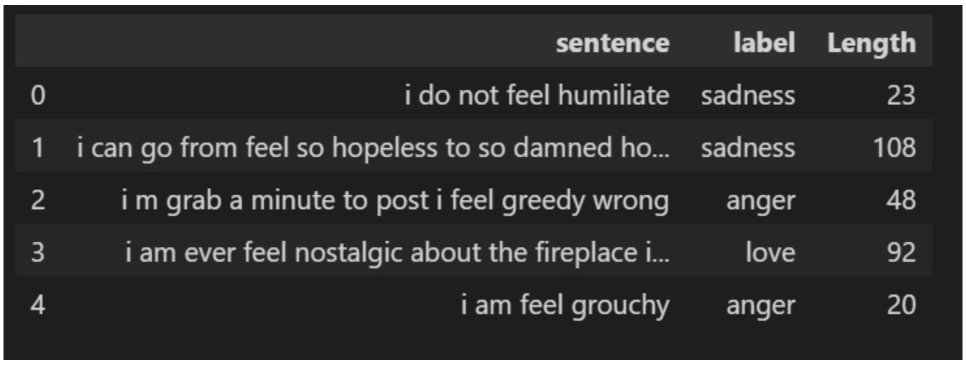
```
%%time

from tqdm._tqdm_notebook import tqdm_notebook
tqdm_notebook.pandas()

def text_preprocessing(df,col_name):
    column = col_name
    df[column] = df[column].progress_apply(lambda x:str(x).lower())
    df[column] = df[column].progress_apply(lambda x: th.cont_exp(x)) #you're -> you are; i'm -> i am
    df[column] = df[column].progress_apply(lambda x: th.remove_emails(x))
    df[column] = df[column].progress_apply(lambda x: th.remove_html_tags(x))
# here we can remove stop-words but in this case removing not, and ,can change the meaning of context

    df[column] = df[column].progress_apply(lambda x: th.remove_special_chars(x))
    df[column] = df[column].progress_apply(lambda x: th.remove_accented_chars(x))
    df[column] = df[column].progress_apply(lambda x: th.make_base(x)) #ran -> run,
    return(df)
```

```
CPU times: user 550 µs, sys: 0 ns, total: 550 µs
```

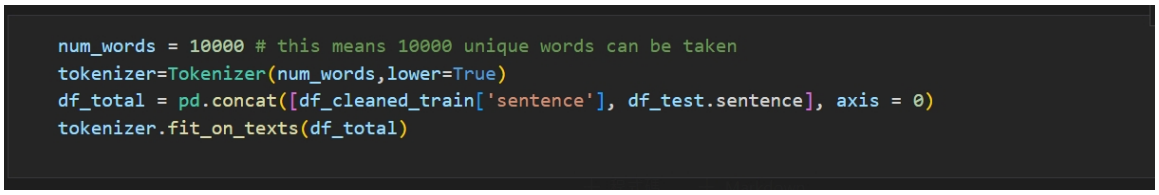


	sentence	label	Length
0	i do not feel humiliate	sadness	23
1	i can go from feel so hopeless to so damned ho...	sadness	108
2	i m grab a minute to post i feel greedy wrong	anger	48
3	i am ever feel nostalgic about the fireplace i...	love	92
4	i am feel grouchy	anger	20

處理後的結果

Tokenizer

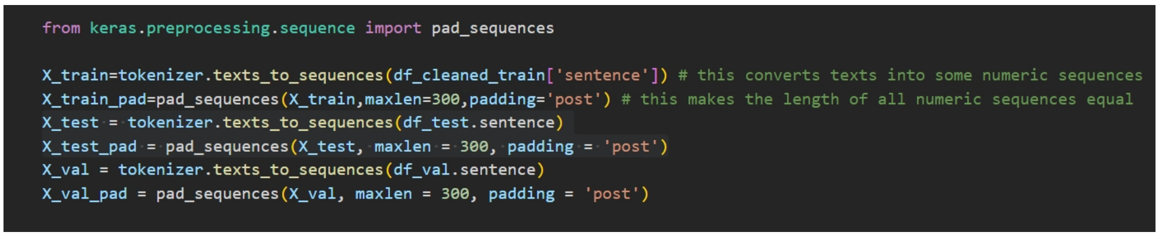
- 使用tokenizer把文本轉換為文檔辭典，每一元素為一文檔



```
num_words = 10000 # this means 10000 unique words can be taken
tokenizer=Tokenizer(num_words,lower=True)
df_total = pd.concat([df_cleaned_train['sentence'], df_test.sentence], axis = 0)
tokenizer.fit_on_texts(df_total)
```

Sequence

- 將向量填充到max_length，參數`'post'`使填充詞量補在文本後端

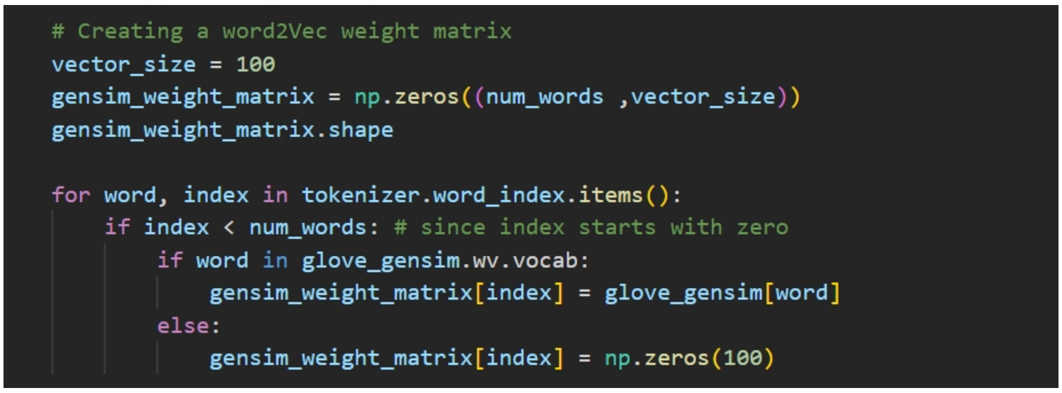


```
from keras.preprocessing.sequence import pad_sequences

X_train=tokenizer.texts_to_sequences(df_cleaned_train['sentence']) # this converts texts into some numeric sequences
X_train_pad=pad_sequences(X_train,maxlen=300,padding='post') # this makes the length of all numeric sequences equal
X_test = tokenizer.texts_to_sequences(df_test.sentence)
X_test_pad = pad_sequences(X_test, maxlen = 300, padding = 'post')
X_val = tokenizer.texts_to_sequences(df_val.sentence)
X_val_pad = pad_sequences(X_val, maxlen = 300, padding = 'post')
```

gensim word2vec

- 使用gensim轉換文本為向量

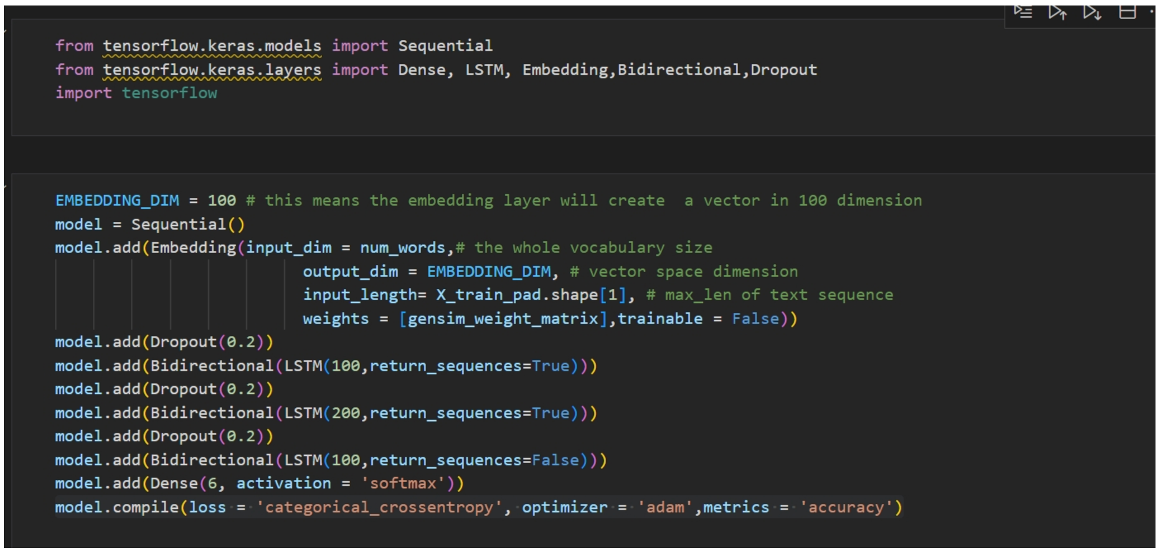


```python
# Creating a word2Vec weight matrix
vector_size = 100
gensim_weight_matrix = np.zeros((num_words ,vector_size))
gensim_weight_matrix.shape

for word, index in tokenizer.word_index.items():
    if index < num_words: # since index starts with zero
        if word in glove_gensim.wv.vocab:
            gensim_weight_matrix[index] = glove_gensim[word]
        else:
            gensim_weight_matrix[index] = np.zeros(100)
```

Build Model

- Use sequential model
- Layer選擇Bidirectional LSTM，增加辨識的效果

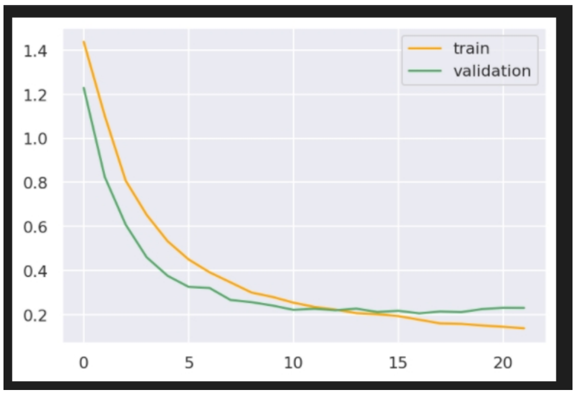


```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Embedding,Bidirectional,Dropout
import tensorflow
```

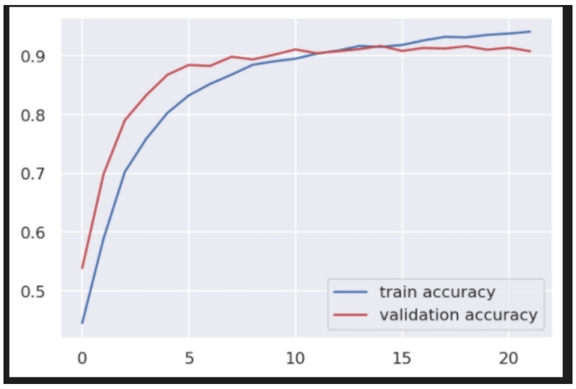
```python
EMBEDDING_DIM = 100 # this means the embedding layer will create  a vector in 100 dimension
model = Sequential()
model.add(Embedding(input_dim = num_words,# the whole vocabulary size
                          output_dim = EMBEDDING_DIM, # vector space dimension
                          input_length= X_train_pad.shape[1], # max_len of text sequence
                          weights = [gensim_weight_matrix],trainable = False))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(100,return_sequences=True)))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(200,return_sequences=True)))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(100,return_sequences=False)))
model.add(Dense(6, activation = 'softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam',metrics = 'accuracy')
```

Result

Loss and accuracy curves

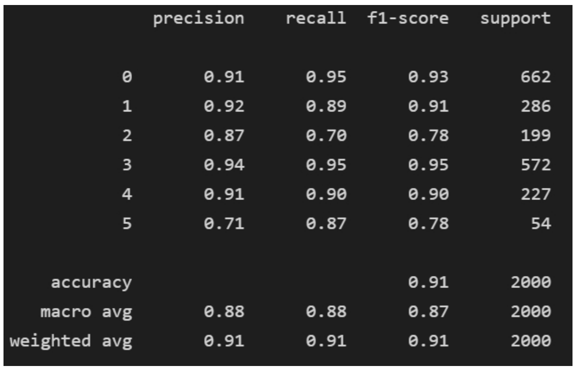


Loss curve



accuracy curve

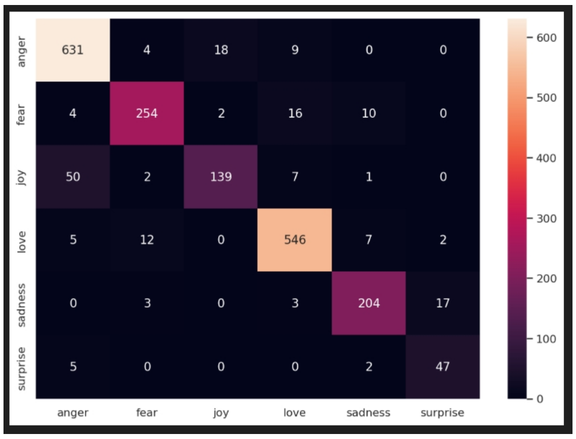
classification report & confusion matrix



```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93       662
           1       0.92      0.89      0.91       286
           2       0.87      0.70      0.78       199
           3       0.94      0.95      0.95       572
           4       0.91      0.90      0.90       227
           5       0.71      0.87      0.78        54

    accuracy                           0.91      2000
   macro avg       0.88      0.88      0.87      2000
weighted avg       0.91      0.91      0.91      2000
```



	anger	fear	joy	love	sadness	surprise
anger	631	4	18	9	0	0
fear	4	254	2	16	10	0
joy	50	2	139	7	1	0
love	5	12	0	546	7	2
sadness	0	3	0	3	204	17
surprise	5	0	0	0	2	47