Student Information

Name: 何思雅

Student ID: 113153036

GitHub ID: kaki7asmr

Kaggle name: shen yi0590

Kaggle private scoreboard snapshot:

79	↑ 3	shenyi0590	0.38945	5	2d

Instructions

1. Find **this page** in **week 10's** of your **grade**. Fill the **Identificate exercises** in the 0000000-ratio3-exam-Deps. You may need to copy some cells from the sub-notebook to this notebook.
2. Submit **this page** to **week 10's** of **your grade**. Find a place in Moodle to ask questions regarding Kaggle Competition. Partner by this link: https://moodle.ncku.edu.tw/mod/assign/view.php?id=1553146. The scoring will be given according to your place in the Private leaderboard ranking:
   - **Bronze 40%**: Get 0% of the 40% available for this section.
   - **Silver 70%**: Get 50% of the 70% available for this section.
   - **Gold 100%**: Get 100% of the 100% available for this section.

   for example, where for the total number of participants under a given name, (ie. if there are 300 participants and you rank first your score will be $100 - 0 \times 30 = 100$, if $100 \times 0.7 = 70$, if $100 \times 0.4 = 40$)
   Submit your score to Leaderboard **BEFORE this deadline (Dec. 25th, 11:59 pm, Tuesday)**. Make sure to take a screenshot of your position at the end of the competition and store it as 'pic_of_leaderboard.png' under the tag folder of this repository and record the URL **Student Information**.
3. Find **this page** in **week 10's** of **your grade**. A report of you work developing the model for the competition (You can use code and comments as it). This report should include what your preprocessing steps, the feature engineering steps and an explanation of your model. You can also mention difficulties? things you tried and? insights you gained.
4. Find **this page** in **week 10's** of **your grade**. It's the file ran to follow Exam code is fixing. i.e., you should **fully run your notebook**.

Upload your files to your repository. Make sure to file this in on the corresponding in repo as assignment.

Make sure to commit and save your changes to your repository **BEFORE the deadline (Dec. 25th, 11:59 pm, Tuesday)**

```
%%%% Write the language here
```

```
# Basic data processing
import numpy as np
import pandas as pd
import os, sys, ast
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import datetime, dateutil
import math, time, scipy, os, sklearn
import collections
import PIL, cv2
import keras, tensorflow
import torch
import torchvision
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torch.utils.data
import torchvision.transforms as T
import torchvision.utils as vutils
from torchvision.datasets import ImageFolder
from PIL import Image, ImageOps, ImageStat, ImageEnhance
import os, sys, numpy, PIL, matplotlib, pandas, sklearn, nltk, keras, cv2
print("Python version: {}".format(sys.version))
print("Numpy 版本: {}".format(numpy.__version__))
print("Pandas 版本: {}".format(pandas.__version__))
# Jupyter extension
%load_ext google.colaboratory.rpy2.ipynb
%load_ext rpy2.ipynb
%load_ext google.colab.drive
%load_ext google.colab.data_table
%matplotlib inline
%pylab inline
import os, sys, numpy, PIL, matplotlib, pandas, sklearn, nltk, keras, cv2
print("Python version: {}".format(sys.version))
print("Numpy 版本: {}".format(numpy.__version__))
print("Pandas 版本: {}".format(pandas.__version__))
# Keras extension
%load_ext google.colaboratory.rpy2.ipynb
%load_ext rpy2.ipynb
%load_ext google.colab.drive
%load_ext google.colab.data_table
%matplotlib inline
%pylab inline
```

1. Data Preparation

1.1 Load data

put in the zipped file and print the specified files with (ls/cat/head/gzip/zcat/bzcat/xz)

```
%pylab inline
import os, sys
import numpy as np
import ast

path_train = "/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/train.jsonl"
path_test = "/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/test.jsonl"

def read_jsonl(path, limit=None):
    print(f"正在讀取 {os.path.basename(path)} ...")
    data = []
    with open(path, 'r', encoding='utf-8') as f:
        for line in f:
            if not line.strip:
                continue
            try:
                obj = ast.literal_eval(line.strip())
                data.append(obj)
                if len(data) % 100000 == 0:
                    print(f"已讀取 {len(data):,} 筆...")
                if limit and len(data) >= limit:
                    break
            except Exception as e:
                print(f"解析錯誤: {e}")
                continue
    print(f"完成！共讀取 {len(data):,} 筆")
    return data

print("\n=== 開始讀取 ===")
train_data = read_jsonl(path_train)

print("\n=== 訓練資料範例 ===")
if train_data:
    print("keys:", train_data[0].keys())
    print("\n第一筆:")
    for k, v in train_data[0].items():
        print(f"  {k}: {v}")

print("\n=== 開始讀取 ===")
test_data = read_jsonl(path_test)

print("\n=== 測試資料範例 ===")
if test_data:
    print("keys:", test_data[0].keys())
    print("\n第一筆:")
    for k, v in test_data[0].items():
        print(f"  {k}: {v}")
```

count the number of lines in the file

```
path_train = "/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/train.jsonl"
!wc -l "{path_train}"
!head -1 "{path_train}"
```

extract jsonl file to pandas dataframe and rename the column names.

```
import os
import numpy as np
import pandas as pd
import ast

path = "/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/train.jsonl"
df = pd.read_csv(path, sep='\t', header=None, quoting=3)

# 檢視前幾筆
print("原始 shape:", df.shape)
df.head(3)

# 解析 JSON
import ast
df = df[0].apply(ast.literal_eval).apply(pd.Series)

# 檢視欄位
print("欄位:", df.columns.tolist())
df.head(3)
```

save it to the drive

```
path_train = "/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/train.jsonl"
path_test = "/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/test.jsonl"

df_train = pd.read_csv(path_train)
df_test = pd.read_csv(path_test)

print("訓練集 shape:", df_train.shape)
print("測試集 shape:", df_test.shape)

df_train.head()
```

	conversation_id	video	timestamp	type	hashtags	tweet_id	text	identification	sentiment
0	1127	hashtag_tweets	2017-10-29T00:00:00Z	tweets	[blackfridaysale]	924,905,955,...	Grab a bargain before the sale ends! #blackfridaysale	train	antagonistic
1	1788	hashtag_tweets	2016-06-28T17:00:00Z	tweets	[antagonistic, flashpointgaming, ps4, ps4pro]	747,609,677,...	@FlashpointGaming Are PS4Pro Bundles coming?	train	neutral
2	1911	hashtag_tweets	2017-11-20T18:00:00Z	tweets	[mondaymotivation]	932,993,512,...	#mondaymotivation #Gratitude #Joy #Love #Peace #Light #Life	train	neutral
3	1911	hashtag_tweets	2017-11-20T18:00:00Z	tweets		932,993,512,...	Very thick crowding Stadio 🅱️🅾️🅾️🅻🅰️🅽🅾️	train	neutral
4	1911	hashtag_tweets	2017-11-20T18:00:00Z	tweets		932,993,512,...	"Trust is not the same as faith. A friend is a ...	train	neutral

separate the words identifications into 20 rows and all zero

```
df_train['identification'] = df_train['identification'].str.split('|')
df_train['identification'].head(3).tolist()
```

merge column sentiment to the identifications and take a look at the df_train

```
df_i = df_train[['conversation_id', 'identification']].explode('identification').rename(columns={'identification': 'text'})
df_i = df_i[df_i['text'].notna()].reset_index(drop=True)
df_i.shape, df_i.head(3)
```

	conversation_id	video	timestamp	type	hashtags	tweet_id	text	identification	sentiment
0	1127	hashtag_tweets	2017-10-29T00:00:00Z	tweets	[blackfridaysale]	924,905,955,...	Grab a bargain before the sale ends! #blackfridaysale	train	antagonistic
1	1788	hashtag_tweets	2016-06-28T17:00:00Z	tweets	[antagonistic, flashpointgaming, ps4, ps4pro]	747,609,677,...	@FlashpointGaming Are PS4Pro Bundles coming?	train	neutral
2	1911	hashtag_tweets	2017-11-20T18:00:00Z	tweets	[mondaymotivation]	932,993,512,...	#mondaymotivation #Gratitude #Joy #Love #Peace #Light #Life	train	neutral
3	1911	hashtag_tweets	2017-11-20T18:00:00Z	tweets		932,993,512,...	Very thick crowding Stadio 🅱️🅾️🅾️🅻🅰️🅽🅾️	train	neutral
4	1911	hashtag_tweets	2017-11-20T18:00:00Z	tweets		932,993,512,...	"Trust is not the same as faith. A friend is a ...	train	neutral

take a look at the df_test

```
df_test.head()
```

	conversation_id	video	timestamp	type	hashtags	tweet_id	text	identification
0	1142	hashtag_tweets	2017-10-31T00:00:00Z	tweets	[democracy]	914,867,477,...	Vote!#Election #Democracy #Democracy #Democ...	test
1	1580	hashtag_tweets	2018-05-28T17:00:00Z	tweets		1,000,782,672,...	"Trust is not the same as faith. A friend is a ...	test
2	38	hashtag_tweets	2016-06-30T00:00:00Z	tweets	[hocoschool, closing, announcement]	748,778,634,...	Where did you have school? When did you last watch...	test
3	1580	hashtag_tweets	2018-05-28T17:00:00Z	tweets	[beautiful,sundaytoday]	1,000,782,672,...	#beautiful,sundaytoday https://t.co/ZNbxJ7BOoq #NoNoNoN...	test
4	1710	hashtag_tweets	2018-01-01T00:00:00Z	tweets		1,010,000,904,...	2018wishes:laugh:lines,wishes:fireflies:sun:a_ph...	test

check the shape of df and df.head(3)

```
df_test[df_test['text'].str.contains('#', na=False)]
```

	conversation_id	video	timestamp	type	hashtags	tweet_id	text	identification
9987	1747	hashtag_tweets	2017-12-27T23:59:59Z	tweets		1,000,747,...	@HobbsTim and two separate concerts of the a...	test

take a look at the shape of df.identification.unique()

```
print("Shape of training df: ", df_train.shape)
print("Shape of testing df: ", df_test.shape)

Shape of training df:  (200000, 9)
Shape of testing df:  (10000, 8)
```

1.2 Data Visualization

Plot word frequency for Top 30 words in both train and test dataset

```
from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer()
train_wordcount = count_vect.fit_transform(df_train['text'])
test_wordcount = count_vect.transform(df_test['text'])

freq_term_frequencies = np.asarray(train_wordcount.sum(axis=0))[0]
print("%d unique words" % len(freq_term_frequencies))

top30_ind = np.argsort(freq_term_frequencies)[-30:]
plt.figure(figsize=(12,6))
plt.bar(range(30), freq_term_frequencies[top30_ind], color="green")
plt.ylabel("Frequency")
plt.xlabel("Words")
plt.title("Top 30 Word Frequencies in Training Set")
plt.xticks(range(30), count_vect.inverse_transform(np.eye(30,30,dtype='int'))[:,top30_ind].ravel(), rotation=90)
plt.show()

freq_term_frequencies = np.asarray(test_wordcount.sum(axis=0))[0]
print("%d unique words" % len(freq_term_frequencies))

top30_ind = np.argsort(freq_term_frequencies)[-30:]
plt.figure(figsize=(12,6))
plt.bar(range(30), freq_term_frequencies[top30_ind], color="orange")
plt.ylabel("Frequency")
plt.xlabel("Words")
plt.title("Top 30 Word Frequencies in Testing Set")
plt.xticks(range(30), count_vect.inverse_transform(np.eye(30,30,dtype='int'))[:,top30_ind].ravel(), rotation=90)
plt.show()

%%file -a ./output.txt
%pwd
%ls -lh
%ls -lh ./output.txt
%cat ./output.txt
```

1.3 Save Data

save to pickle file

```
df_train.to_pickle('df_train.pkl')
df_test.to_pickle('df_test.pkl')
```

reading pickle file

```
df_train = pd.read_pickle('df_train.pkl')
df_test = pd.read_pickle('df_test.pkl')
```

1.4 Exploratory Data Analysis

Plot the distribution of the emotions in the training data

```
df_train.groupby('sentiment').count()['text']
```

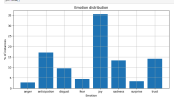
```
sentiment
anger          20000
disgust        20000
fear           20000
joy            20000
neutral        20000
sadness        20000
surprise       20000
trust          20000
Name: text, dtype: int64
```

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

# Text to display
text = df_train['sentiment']

# Prepare the text data for word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(' '.join(text))

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Sentiments')
plt.show()
```



2. Feature engineering

2.1 Text Pre-processing

```
import urllib

urls_train_path = []
urls_train = df_train[df_train['text'].str.contains('http')]['text']
for i, t in enumerate(urls_train):
    urls_train_path.append('/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/train_urls/'+str(i)+'.html')
    urlretrieve('http://textfiles.com/stuff/indexes/'+str(i)+'.html', '/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/train_urls/'+str(i)+'.html')
urls_test_path = []
urls_test = df_test[df_test['text'].str.contains('http')]['text']
for i, t in enumerate(urls_test):
    urls_test_path.append('/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/test_urls/'+str(i)+'.html')
    urlretrieve('http://textfiles.com/stuff/indexes/'+str(i)+'.html', '/content/drive/My Drive/113153036-Machine Learning/113153036-Machine Learning-Data/test_urls/'+str(i)+'.html')

%%file -a ./output.txt
%pwd
%ls -lh
%ls -lh ./output.txt
%cat ./output.txt
```

```
%%file -a ./output.txt
%%bash
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/train_urls | head -3
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/test_urls | head -3
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/train_urls | wc -l
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/test_urls | wc -l
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/train_urls | tail -3
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/test_urls | tail -3
%%file -a ./output.txt
%%bash
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/train_urls | head -3
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/test_urls | head -3
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/train_urls | wc -l
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/test_urls | wc -l
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/train_urls | tail -3
%ls -lh /content/drive/MyDrive/113153036-Machine Learning/113153036-Machine Learning-Data/test_urls | tail -3
```

```
%%file -a ./output.txt
import re, nltk, gensim, sklearn, numpy, scipy, scipy.stats, PIL, cv2
from gensim.models import KeyedVectors
from nltk.stem.porter import *
from nltk.stem.snowball import SnowballStemmer
import string
from nltk.corpus import stopwords
stemmer = SnowballStemmer("english")

# set the maximum number of words
MAX_NUM_WORDS = 10000

# Replacement for the stop words
stoplist = stopwords.words('english')
replacement = {
    "ain't": "is not",
    "aren't": "are not",
    "can't": "cannot",
    "cannot": "cannot",
    "could've": "could have",
    "couldn't": "could not",
    "didn't": "did not",
    "doesn't": "does not",
    "don't": "do not",
    "gimme": "give me",
    "gonna": "going to",
    "gotta": "got to",
    "hadn't": "had not",
    "hasn't": "has not",
    "haven't": "have not",
    "he'd": "he would",
    "he'll": "he will",
    "he's": "he is",
    "i'd": "i would",
    "i'll": "i will",
    "i'm": "i am",
    "i've": "i have",
    "isn't": "is not",
    "it'd": "it would",
    "it'll": "it will",
    "it's": "it is",
    "let's": "let us",
    "mustn't": "must not",
    "shan't": "shall not",
    "she'd": "she would",
    "she'll": "she will",
    "she's": "she is",
    "shouldn't": "should not",
    "that's": "that is",
    "there's": "there is",
    "they'd": "they would",
    "they'll": "they will",
    "they're": "they are",
    "they've": "they have",
    "wasn't": "was not",
    "we'd": "we would",
    "we'll": "we will",
    "we're": "we are",
    "we've": "we have",
    "weren't": "were not",
    "what'll": "what will",
    "what're": "what are",
    "what's": "what is",
    "what've": "what have",
    "where's": "where is",
    "who'd": "who would",
    "who'll": "who will",
    "who're": "who are",
    "who's": "who is",
    "who've": "who have",
    "won't": "will not",
    "wouldn't": "would not",
    "you'd": "you would",
    "you'll": "you will",
    "you're": "you are",
    "you've": "you have",
    "rt": " ",
    "u": "you",
    "ur": "your",
    "n": "and"
}

# Preprocessing function
def process_text(text):
    global replacement
    global stoplist
    global stemmer
    text = text.lower().split()
    stext = []
    for word in text:
        if word in replacement:
            word = replacement[word]
        if word[0:4] == 'http':
            continue
        if len(word) > 3 and word[0:2] == '@@':
            word = '@@'
        if len(word) > 3 and word[0:2] == '@':
            word = '@'
        if len(word) > 3 and word[0:2] == '&&':
            word = '&&'
        if len(word) > 3 and word[0:2] == '##':
            word = '##'
        if len(word) > 3 and word[0:2] == '$$':
            word = '$$'
        if len(word) > 3 and word[0:2] == '%%':
            word = '%%'
        if len(word) > 3 and word[0:2] == '^^':
            word = '^^'
        if len(word) > 3 and word[0:2] == '**':
            word = '**'
        if len(word) > 3 and word[0:2] == '((':
            word = '(('
        if len(word) > 3 and word[0:2] == '))':
            word = '))'
        if len(word) > 3 and word[0:2] == '--':
            word = '--'
        if len(word) > 3 and word[0:2] == '__':
            word = '__'
        if len(word) > 3 and word[0:2] == '++':
            word = '++'
        if len(word) > 3 and word[0:2] == '==':
            word = '==('
        if len(word) > 3 and word[0:2] == '[[':
            word = '[['
        if len(word) > 3 and word[0:2] == ']]':
            word = ']]'
        if len(word) > 3 and word[0:2] == '<<':
            word = '<<'
        if len(word) > 3 and word[0:2] == '>>':
            word = '>>'
        if len(word) > 3 and word[0:2] == '||':
            word = '||'
        if len(word) > 3 and word[0:2] == '//':
            word = '//'
        if len(word) > 3 and word[0:2] == '/*':
            word = '/*'
        if len(word) > 3 and word[0:2] == '*/':
            word = '*/'
        if len(word) > 3 and word[0:2] == '/>':
            word = '/>'
        if len(word) > 3 and word[0:2] == '</':
            word = '</'
        if len(word) > 3 and word[0:2] == '-[':
            word = '-['
        if len(word) > 3 and word[0:2] == ']-':
            word = ']-'
        if len(word) > 3 and word[0:2] == '-(':
            word = '-('
        if len(word) > 3 and word[0:2] == ')-':
            word = ')-'
        if len(word) > 3 and word[0:2] == '-{':
            word = '-{'
        if len(word) > 3 and word[0:2] == '}-':
            word = '}-'
        if len(word) > 3 and word[0:2] == '-|':
            word = '-|'
        if len(word) > 3 and word[0:2] == '|-':
            word = '|-'
        if len(word) > 3 and word[0:2] == '-/':
            word = '-/'
        if len(word) > 3 and word[0:2] == '/-':
            word = '/-'
        if len(word) > 3 and word[0:2] == '-*':
            word = '-*'
        if len(word) > 3 and word[0:2] == '*-':
            word = '*-'
        if len(word) > 3 and word[0:2] == '-+':
            word = '-+'
        if len(word) > 3 and word[0:2] == '+-':
            word = '+-'
        if len(word) > 3 and word[0:2] == '-=':
            word = '-='
        if len(word) > 3 and word[0:2] == '=-':
            word = '=-'
        if len(word) > 3 and word[0:2] == '-_':
            word = '-_'
        if len(word) > 3 and word[0:2] == '_-':
            word = '_-'
        if len(word) > 3 and word[0:2] == '-~':
            word = '-~'
        if len(word) > 3 and word[0:2] == '~-':
            word = '~-'
        if len(word) > 3 and word[0:2] == '-%':
            word = '-%'
        if len(word) > 3 and word[0:2] == '%-':
            word = '%-'
        if len(word) > 3 and word[0:2] == '-^':
            word = '-^'
        if len(word) > 3 and word[0:2] == '^-':
            word = '^-'
        if len(word) > 3 and word[0:2] == '-&':
            word = '-&'
        if len(word) > 3 and word[0:2] == '&-':
            word = '&-'
        if len(word) > 3 and word[0:2] == '-$':
            word = '-$'
        if len(word) > 3 and word[0:2] == '$-':
            word = '$-'
        if len(word) > 3 and word[0:2] == '-#':
            word = '-#'
        if len(word) > 3 and word[0:2] == '#-':
            word = '#-'
        if len(word) > 3 and word[0:2] == '-@':
            word = '-@'
        if len(word) > 3 and word[0:2] == '@-':
            word = '@-'
        if len(word) > 3 and word[0:2] == '-!':
            word = '-!'
        if len(word) > 3 and word[0:2] == '!-':
            word = '!-'
        if len(word) > 3 and word[0:2] == '-?':
            word = '-?'
        if len(word) > 3 and word[0:2] == '?-':
            word = '?-'
        if len(word) > 3 and word[0:2] == '-;':
            word = '-;'
        if len(word) > 3 and word[0:2] == ';-':
            word = ';-'
        if len(word) > 3 and word[0:2] == '-:':
            word = '-:'
        if len(word) > 3 and word[0:2] == ':-':
            word = ':-'
        if len(word) > 3 and word[0:2] == '-.':
            word = '-.'
        if len(word) > 3 and word[0:2] == '.-':
            word = '.-'
        if len(word) > 3 and word[0:2] == '-,':
            word = '-,'
        if len(word) > 3 and word[0:2] == ',-':
            word = ',-'
        if len(word) > 3 and word[0:2] == '-\':
            word = '-\'
        if len(word) > 3 and word[0:2] == '\-':
            word = '\-'
        if len(word) > 3 and word[0:2] == '-`':
            word = '-`'
        if len(word) > 3 and word[0:2] == '`-':
            word = '`-'
        if len(word) > 3 and word[0:2] == '-"':
            word = '-"'
        if len(word) > 3 and word[0:2] == '"-':
            word = '"-'
        if len(word) > 3 and word[0:2] == '-\'':
            word = '-\'
        if len(word) > 3 and word[0:2] == '\'-':
            word = '\'-'
        if len(word) > 3 and word[0:2] == '-<':
            word = '-<'
        if len(word) > 3 and word[0:2] == '<-':
            word = '<-'
        if len(word) > 3 and word[0:2] == '->':
            word = '->'
        if len(word) > 3 and word[0:2] == '>-':
            word = '>-'
        if len(word) > 3 and word[0:2] == '->':
            word = '->'
        if len(word) > 3 and word[0:2] == '<-':
            word = '<-'
        if len(word) > 3 and word[0:2] == '-<':
            word = '-
```