Assignment 6.2: Preparing Data for Final Project

Team 2: Leonid Shpaner, Jose Luis Estrada, Christopher Robinson

This notebook implements a text mining sentiment analysis project

First we fetch the data from google drive

[1]:
```
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score, roc_curve
from sklearn.pipeline import Pipeline
import nltk
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\lshpaner\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[1]:
```
True
```

Read data

[2]:
```
import re
import subprocess

def download_gdrive(id, print_stout=True):
    coomand = 'gdown https://drive.google.com/uc?id={}'.format(id)
    returned_value = subprocess.run(coomand, shell=True, stdout=subprocess.PIPE,
    stderr=subprocess.STDOUT)
    if print_stout: print(returned_value.stdout.decode("utf-8"))
    else: print("Download Complete")

train_data = download_gdrive("10rDgl5zAvUdVgSoVngHfwJnf8I1tdpZi", print_stout=True)
test_data = download_gdrive("10qeDcgwdJC76Nv5cCj6WsUYjD6846fEL", print_stout=True)
```

```
Downloading…
From: https://drive.google.com/uc?id=10rDgl5zAvUdVgSoVngHfwJnf8I1tdpZi
To: c:\Users\lshpaner\Documents\Github
Repositories\msads509_final_project\twitter_emotions\notebook\train_data.csv
100%|      | 239M/239M [03:09<00:00, 1.26MB/s]

Downloading…
From: https://drive.google.com/uc?id=10qeDcgwdJC76Nv5cCj6WsUYjD6846fEL
To: c:\Users\lshpaner\Documents\Github
Repositories\msads509_final_project\twitter_emotions\notebook\test_data.csv
100%|      | 74.3k/74.3k [00:00<00:00, 436kB/s]
```

[3]:
```
columns = ['polarity', 'tweetid', 'date', 'query_name', 'user', 'text']
dftrain = pd.read_csv('train_data.csv',
                      header = None,
                      encoding ='ISO-8859-1')
dftest = pd.read_csv('test_data.csv',
                     header = None,
                     encoding ='ISO-8859-1')
dftrain.columns = columns
dftest.columns = columns
```

[4]:
```
dftrain.sample(10)
```

[4]:

	polarity	tweetid	date	query_name	user	text
676838	0	2248511881	Fri Jun 19 20:44:31 PDT 2009	NO_QUERY	jawnahthin	I am exhausted… and I think I left my brain …
703009	0	2255579257	Sat Jun 20 11:28:32 PDT 2009	NO_QUERY	wendy_munro	@monkeycoco They are all repeats here now. No…
620728	0	2228512707	Thu Jun 18 14:31:59 PDT 2009	NO_QUERY	miriamjlee	Had a lovely nap!~ time to get ready for psych…
204185	0	1972555803	Sat May 30 09:26:48 PDT 2009	NO_QUERY	AKLSweets41	is mad about her new dress
1011756	4	1881136628	Fri May 22 03:43:24 PDT 2009	NO_QUERY	camden_girl	@mattconfusion hi teo! yes here I am…quite s…
886151	4	1686661949	Sun May 03 06:19:11 PDT 2009	NO_QUERY	sanniu	@allmae yea it is ! can't wait to c next episode
2507	0	1468390805	Tue Apr 07 01:22:48 PDT 2009	NO_QUERY	dionbp	Morning!! I'm baggered! Been the gym then off …
1509077	4	2174752210	Sun Jun 14 23:14:04 PDT 2009	NO_QUERY	Ashhh_	today i brought the coolest Hannah Montana nec…
1260870	4	1998455170	Mon Jun 01 18:34:36 PDT 2009	NO_QUERY	Sicklillovesong	@RobCusella This year…I want summer like now
676515	0	2248421087	Fri Jun 19 20:35:51 PDT 2009	NO_QUERY	DanaJ12	has a broken car. An extremely broken car

Text Pre-Processing

[5]:
```
user_pat = '(?<=^|(?<=[^a-zA-Z0-9-_\.]))@([A-Za-z]+[A-Za-z0-9]+)'
http_pat = '(https?:\/\/(?:www\.|(?!www))[^\s\.]+\.[^\s]{2,}|www\.[^\s]+\.[^\s]{2,})'
repeat_pat, repeat_repl = "(.)\\1\\1+",'\\1\\1'

def transform(X):
    pp_text = X
    pp_text = pp_text.str.replace(pat = user_pat, repl = 'USERNAME')
    pp_text = pp_text.str.replace(pat = http_pat, repl = 'URL')
    pp_text.str.replace(pat = repeat_pat, repl = repeat_repl)
    return pp_text
```

Descriptive Statistics

[6]:
```
import string
from string import punctuation

def descriptive_stats(tokens, top_num_tokens = 5, verbose=True) :
    """
        Given a list of tokens, print number of tokens, number of unique tokens,
        number of characters, lexical diversity (https://en.wikipedia.org/wiki/
 ↪Lexical_diversity),
        and num_tokens most common tokens. Return a list with the number of tokens,␣
 ↪number
        of unique tokens, lexical diversity, and number of characters.
    """

    # Fill in the correct values here.

    num_tokens = len(tokens)
    num_unique_tokens = len(set(tokens))
    lexical_diversity = num_unique_tokens/num_tokens
    num_characters = len("".join(tokens))

    if verbose :
        print(f"There are {num_tokens} tokens in the data.")
        print(f"There are {num_unique_tokens} unique tokens in the data.")
        print(f"There are {num_characters} characters in the data.")
        print(f"The lexical diversity is {lexical_diversity:.3f} in the data.")

        # print the five most common tokens

        # use a list comprehension on a set to exclude repeating
        # tokens and empty strings
        unique_tokens = [token for token in set(tokens) if token]
        # use a unique tokens to check frequency of tokens in
        # original list
        counts = [tokens.count(token) for token in unique_tokens]
        result = []

        # iterate over the range of tokens to locate and find the
        # maximum count, then mutate both unique_tokens and counts
        # based on the associated position
        for _ in range(top_num_tokens):
            max_count = max(counts)
            max_count_pos = counts.index(max_count)
            most_common = unique_tokens.pop(max_count_pos)
            result.append(most_common)
            counts.pop(max_count_pos)

        print(result)

    return([num_tokens, num_unique_tokens,
```

```
            lexical_diversity,
            num_characters])
```

[7]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

#PRE-Process step
#feel free to add more items to the pre-processing step
dftrain['text'] = transform(dftrain['text'])
##word_tokenize
dftrain['text'] = dftrain.apply(lambda row: nltk.word_tokenize(row['text']), axis=1)

print("DESCRIPTIVE STATS ON TEXT: ")
all = []
#on 10k data
for li in dftrain['text'].sample(10000).iteritems(): all += li[1]
#on all data
# for li in dftrain['text'].iteritems(): all += li[1]

#feel free to add more items to analyze in descriptive_stats
descriptive_stats(all, verbose=True)
print("\n")

print("DESCRIPTIVE STATS ON SENTIMENT POLARITY:")
dftrain['polarity'].describe()
```

```
DESCRIPTIVE STATS ON TEXT:
There are 158217 tokens in the data.
There are 17778 unique tokens in the data.
There are 583544 characters in the data.
The lexical diversity is 0.112 in the data.
['!', '.', 'USERNAME', 'I', 'to']


DESCRIPTIVE STATS ON SENTIMENT POLARITY:
```

[7]:
```
count    1.600000e+06
mean     2.000000e+00
std      2.000001e+00
min      0.000000e+00
25%      0.000000e+00
50%      2.000000e+00
75%      4.000000e+00
max      4.000000e+00
Name: polarity, dtype: float64
```

TODO: Train and test model

[8]:
```
Xtest, ytest = dftest.text[dftest.polarity!=2], dftest.polarity[dftest.polarity!=2]
```