Problem Set 3

For this problem set, you will expand on PS2 to perform and evaluate various sentiment classification methods.

Your name: Rudy Martinez

You abc123: Lpe538

Submission Instructions

After completing the exercises below, generate a pdf of the code **with** outputs. After that create a zip file containing both the completed exercise and the generated PDF/HTML. You are **required** to check the PDF/HTML to make sure all the code **and** outputs are clearly visible and easy to read. If your code goes off the page, you should reduce the line size. I generally recommend not going over 80 characters.

Finally, name the zip file using a combination of your the assigment and your name, e.g., ps3_rios.zip

Exercise 1 (1 point)

For this step, you will load the training and test sentiment datasets "twitdata_TEST.tsv" and "allTrainingData.tsv". The data should be loaded into 4 lists of strings: X_txt_train, X_txt_test, y_test, y_train.

Note, when using csvreader, you need to pass the "quoting" the value csv.QUOTE_NONE.

In [31]:
```
import csv
import numpy as np
```

In [37]:
```
X_txt_train = []
y_train = []

with open('allTrainingData.tsv') as x_train_file:
    tsv_reader = csv.reader(x_train_file, delimiter = '\t', quoting = csv.QUOTE_NONE)

    for row in tsv_reader:
        X_txt_train.append(row[3])
        y_train.append(row[2])
```

In [36]:
```
X_txt_test = []
y_test = []

with open('twitdata_TEST.tsv') as x_test_file:
    tsv_reader = csv.reader(x_test_file, delimiter = '\t', quoting = csv.QUOTE_NONE)

    for row in tsv_reader:
        X_txt_test.append(row[3])
        y_test.append(row[2])
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

In [38]:
```
assert(type(X_txt_train) == type(list()))
assert(type(X_txt_train[0]) == type(str()))
assert(type(X_txt_test) == type(list()))
assert(type(X_txt_test[0]) == type(str()))
assert(type(y_test) == type(list()))
assert(type(y_train) == type(list()))
assert(len(X_txt_test) == 3199)
assert(len(y_test) == 3199)
assert(len(X_txt_train) == 8018)
assert(len(y_train) == 8018)
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

Exercise 2 (2 point)

This part is similar to HW2 (using the positive_words and negative_words variables). We will compare last homework's lexicon-based classification method with supervised models. Only make predictions on the test split and store all predictions in the list lex_test_preds. Next, calculate the **micro** precision, recall, and f1 scores using the lex_test_preds list.

You can learn more about lexicon-based classification in Chapter 19.6. If you are interested, the chapter is available online for free at the following link: [Speech and Language Processing]

In [16]:
```
# DO NOT MODIFY THE CODE IN THIS CELL
class LexiconClassifier():
    def __init__(self):
        self.positive_words = set()
        with open('positive-words.txt', encoding = 'utf-8') as iFile:
```

```
            for row in iFile:
                self.positive_words.add(row.strip())

        self.negative_words = set()
        with open('negative-words.txt', encoding='iso-8859-1') as iFile:
            for row in iFile:
                self.negative_words.add(row.strip())

    def predict(self, sentence):
        num_pos_words = 0
        num_neg_words = 0
        for word in sentence.lower().split():
            if word in self.positive_words:
                num_pos_words += 1
            elif word in self.negative_words:
                num_neg_words += 1

        pred = 'neutral'
        if num_pos_words > num_neg_words:
            pred = 'positive'
        elif num_pos_words < num_neg_words:
            pred = 'negative'

        return pred

    def count_pos_words(self, sentence):
        num_pos_words = 0
        for word in sentence.lower().split():
            if word in self.positive_words:
                num_pos_words += 1
        return num_pos_words

    def count_neg_words(self, sentence):
        num_neg_words = 0
        for word in sentence.lower().split():
            if word in self.negative_words:
                num_neg_words += 1
        return num_neg_words
```

In [17]:

```
# WRITE CODE HERE
import numpy as np
from sklearn.metrics import precision_score, recall_score, f1_score

# 1. Instatiate that class
lex_class = LexiconClassifier()
```

```
lex_test_preds = [] # Initialize this as an empty list

# Loop over X_txt_test
#    for each string in X_txt_test (i.e., for each item in the list), pass it to LexiconClassifiers .predict()
#    append the prediction to lex_test_preds
for string in X_txt_test:
    lex_test_preds.append(lex_class.predict(string))

precision = precision_score(y_test, lex_test_preds, average = 'micro') # Get scores using lex_test_preds and y_
recall = recall_score(y_test, lex_test_preds, average = 'micro') # Get scores using lex_test_preds and y_test w
f1 = f1_score(y_test, lex_test_preds, average = 'micro') # Get scores using lex_test_preds and y_test with the
print("Precision: {:.4f}".format(precision))
print("Recall: {:.4f}".format(recall))
print("F1: {:.4f}".format(f1))
```

```
Precision: 0.5827
Recall: 0.5827
F1: 0.5827
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

In [18]:

```
assert(type(lex_test_preds) == type(list()))
assert(type(lex_test_preds[0]) == type(str()))
assert(set(lex_test_preds) == set(["positive", "negative", "neutral"]))
assert(len(lex_test_preds) == len(y_test))
assert(type(precision) == type(float()) or type(precision) == type(np.float64()))
assert(type(recall) == type(float()) or type(recall) == type(np.float64()))
assert(type(f1) == type(float()) or type(f1) == type(np.float64()))
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

Exercise 3 (1 point)

Again, using the LexiconClassifier, write code to generate a lists of lists where each sublist contains the number of positive words and negative words in a tweet. Assume we are give the train test datasets

```
X_txt_train = ["good good", "bad bad"]
X_txt_test = ["great", "bad bad great"]
```

you should write code that creates two lists of lists as follows:

```
X_train_lexicon_features = [[2, 0], [0,2]] # [2, 0] means the first tweet has 2 positive words and 0 negative words.
X_test_lexicon_features = [[1, 0], [1, 2]]
```

In [25]:
```
# WRITE CODE HERE

X_train_lexicon_features = [] # Initailize to an empty list. This will be a list of lists
X_test_lexicon_features = [] #  Initailize to an empty list. This will be a list of lists

# Loop over X_txt_test
#    for each string in X_txt_test (i.e., for each item in the list), pass it to LexiconClassifiers .count_pos_
#    append a list with the counts to X_test_lexicon_features
for string in X_txt_test:
    X_test_lexicon_features.append([lex_class.count_pos_words(string), lex_class.count_neg_words(string)])

# Loop over X_txt_train
#    for each string in X_txt_train (i.e., for each item in the list), pass it to LexiconClassifiers .count_pos
#    append a list with the counts to X_train_lexicon_features
for string in X_txt_train:
    X_train_lexicon_features.append([lex_class.count_pos_words(string), lex_class.count_neg_words(string)])
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

In [39]:
```
assert(type(X_train_lexicon_features) == type(list()))
assert(type(X_test_lexicon_features) == type(list()))
assert(type(X_test_lexicon_features[0]) == type(list()))
assert(len(X_train_lexicon_features) == len(X_txt_train))
assert(len(X_test_lexicon_features) == len(X_txt_test))
assert(len(X_train_lexicon_features[0]) == 2)
assert(len(X_test_lexicon_features[0]) == 2)
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

Exercise 4 (2 points)

For this task you should creat a feature matrix using CountVectorizer and train a LinearSVC model from scikit-learn. On the train split, use GridSearchCV to find the best LinearSVC C values (0.0001, 0.001, 0.001, 0.01, 0.1, 1, 10, or 100) based on the micro f1 scoring metric (hint: "micro" average) and set the cv parameter to 5. Also, with the CountVectorizer, only use unigrams (i.e., set ngram_range = (1,1)). Note that GridSearchCV will retrain the final classifier using the best parameters, so you don't need to do it manually.

In [45]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import precision_score, recall_score, f1_score

import numpy as np
np.random.seed(42)
import random
random.seed(42)

# WRITE CODE HERE

# Summary:
# 1. Convert X_txt_train and X_txt_test to matricies of numbers (i.e., use CountVectorizer)
vec = CountVectorizer(ngram_range = (1,1))

X_train = vec.fit_transform(X_txt_train) # This should be a matrix
X_test = vec.transform(X_txt_test) # This should be a matrix

# Initialize the classifier LinearSVC
svc = LinearSVC()

# Create the params with the C values
params = {"C": [0.0001, 0.001, 0.001, 0.01, 0.1, 1., 10., 100.]}

# Initialize GridSearchCV
clf = GridSearchCV(svc, params, cv = 5, scoring = 'f1_micro')

# "fit" the model  on X_train
clf.fit(X_train, y_train)

validation_score = clf.best_score_ # Get the score from the GridSearchCV "best score"
print("Validation F1: {:.4f}".format(validation_score))

svm_test_predictions = clf.predict(X_test) # "predict" on X_test

precision = precision_score(y_test, svm_test_predictions, average = 'micro') # Get scores using svm_test_predic
recall = recall_score(y_test, svm_test_predictions, average = 'micro')
f1 = f1_score(y_test, svm_test_predictions, average = 'micro')

print("Precision: {:.4f}".format(precision))
print("Recall: {:.4f}".format(recall))
print("F1: {:.4f}".format(f1))
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:976: ConvergenceWarning: Liblinear failed to co
nverge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
Validation F1: 0.6593
Precision: 0.6511
Recall: 0.6511
F1: 0.6511
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

In [46]:

```
from scipy.sparse import csr_matrix
assert(type(X_train) == type(csr_matrix(0)) or type(X_train) == type(np.array(0)))
assert(type(X_test) == type(csr_matrix(0)) or type(X_test) == type(np.array(0)))
assert(X_train.shape[0] == len(X_txt_train))
assert(X_test.shape[0] == len(X_txt_test))
assert(X_train.shape[1] == X_test.shape[1])
assert(type(precision) == type(float()) or type(precision) == type(np.float64()))
assert(type(recall) == type(float()) or type(recall) == type(np.float64()))
assert(type(f1) == type(float()) or type(f1) == type(np.float64()))
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

Exercise 5 (2 points)

Repeat the experiment from exercise 4, but include the lexicon features with the CountVectorizer features. Specifically, you need to concatenate the variables `X_train_lexicon_features` and `X_test_lexicon_features` with `X_train` and `X_test`, respectively. Intuitively, we are performing feature engineering by adding "lexicon features".

HINT: You will need to convert the lexicon features to numpy arrays then call hstack from the scipy.sparse library (https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.hstack.html)

In [49]:

```
import scipy.sparse as sp

import numpy as np
np.random.seed(42)
import random
random.seed(42)

# WRITE CODE HERE
```

```
# Summary:
# 1. Convert X_txt_train and X_txt_test to matricies of numbers (i.e., use CountVectorizer)
vec = CountVectorizer(ngram_range = (1,1))

X_train_w_lex = vec.fit_transform(X_txt_train) # This will be the matrix from CountVectorizer (X_txt_train)
X_test_w_lex = vec.transform(X_txt_test)

# Now we need to convert X_train_lexicon_features and X_test_lexicon_features to numpy arrays
# "hstack" X_train_lexicon_features with X_train_w_lex
# "hstack" X_test_lexicon_features with X_test_w_lex
X_train_lexicon_features = np.array(X_train_lexicon_features)
X_test_lexicon_features = np.array(X_test_lexicon_features)

X_train_w_lex = sp.hstack((X_train_lexicon_features, X_train_w_lex))
X_test_w_lex = sp.hstack((X_test_lexicon_features, X_test_w_lex))

# Initialize the classifier LinearSVC
svc = LinearSVC()

# Create the params with the C values
params = {"C": [0.0001, 0.001, 0.001, 0.01, 0.1, 1., 10., 100.]}

# Initialize GridSearchCV
clf = GridSearchCV(svc, params, cv = 5)

# "fit" the model  on X_train_w_lex
clf.fit(X_train_w_lex, y_train)

validation_score = clf.best_score_
print("Validation F1: {:.4f}".format(validation_score))

svm_lex_test_predictions = clf.predict(X_test_w_lex) # Get predictions on X_test_w_lex

precision = precision_score(y_test, svm_lex_test_predictions, average = 'micro') # Get scores using svm_test_pr
recall = recall_score(y_test, svm_lex_test_predictions, average = 'micro')
f1 = f1_score(y_test, svm_lex_test_predictions, average = 'micro')

print("Precision: {:.4f}".format(precision))
print("Recall: {:.4f}".format(recall))
print("F1: {:.4f}".format(f1))
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:976: ConvergenceWarning: Liblinear failed to co
nverge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
Validation F1: 0.6729
Precision: 0.6555
```

```
Recall: 0.6555
F1: 0.6555
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

In [50]:
```
from scipy.sparse import csr_matrix
assert(X_train_w_lex.shape[0] == len(X_txt_train))
assert(X_test.shape[0] == len(X_txt_test))
assert(X_train_w_lex.shape[1] == X_test.shape[1] + 2)
assert(X_train_w_lex.shape[1] == X_test_w_lex.shape[1])
assert(type(precision) == type(float()) or type(precision) == type(np.float64()))
assert(type(recall) == type(float()) or type(recall) == type(np.float64()))
assert(type(f1) == type(float()) or type(f1) == type(np.float64()))
print("Asserts Completed Successfully!")
```

```
Asserts Completed Successfully!
```

Exercise 6 (2 points)

For this exercise, you will perform manual analysis of the predictions. Answer the questions below.

In [53]:
```
count_tweets = 1
num_tweets = 0
for text, svm_pred, svm_lex_pred, lex_pred, y  in zip(X_txt_test, svm_test_predictions, svm_lex_test_prediction
    print("{} Tweet: {}".format(count_tweets, text))
    print("Ground-Truth Class: {}".format(y))
    print("SVM Prediction: {}".format(svm_pred))
    print("SVM+Lexicon Prediction: {}".format(svm_lex_pred))
    print("Lexicon Model Prediction: {}".format(lex_pred))
    print()

    num_tweets += 1
    count_tweets += 1
    if num_tweets == 20:
        break
```

```
1 Tweet: Musical awareness: Great Big Beautiful Tomorrow has an ending, Now is the time does not
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: positive

2 Tweet: On Radio786 100.4fm 7:10 Fri Oct 19 Labour analyst Shawn Hattingh: Cosatu's role in the context of unr
```

```
est in the mining http://t.co/46pjzzl6
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: negative

3 Tweet: Kapan sih lo ngebuktiin,jan ngomong doang Susah Susah.usaha Aja blm udh nyerah,inget.if you never try
you'll never know.cowok kok gentle bgt
Ground-Truth Class: negative
SVM Prediction: neutral
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: positive

4 Tweet: Tomorrow come and hear @DavidWillettsMP&amp;@MASieghart debate "Navigating the new Higher Education ma
rket" 5.30pm, Jurys Inn #CPC12
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

5 Tweet: Excuse the connectivity of this live stream, from Baba Amr, so many activists using only one Sat Mode
m. LIVE http://t.co/U283IhZ5 #Homs
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: negative

6 Tweet: Show your LOVE for your local field &amp; it might win an award!  Gallagher Park #Bedlington current 4
th in National Award http://t.co/WeiMDtQt
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: positive

7 Tweet: @firecore Can you tell me when an update for the Apple TV 3rd gen becomes available? The missing updat
e holds me back from buying #appletv3
Ground-Truth Class: positive
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

8 Tweet: @Heavensbasement The Crown, Filthy McNastys, Katy Dalys or the Duke if York in Belfast! Can't wait to
catch you guys tomorrow night!
Ground-Truth Class: positive
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: negative

9 Tweet: Uncover the Eternal City! Return flights to Rome travel on the 21st January, for 3 nights Augustea, 3
```

```
star Hotel... http://t.co/tw0Jeh9g
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

10 Tweet: My #cre blog Oklahoma Per Square Foot returns to the @JournalRecord blog hub tomorrow. I will have so
me interesting local data to share.
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: positive

11 Tweet: "@bbcburnsy: Loads from SB; talks with Chester continue; no deals 4 out of contract players 'til Jan;
Dev t Roth ,Coops to Chest'ld #hcafc"
Ground-Truth Class: negative
SVM Prediction: negative
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

12 Tweet: Trey Burke has been suspended for the Northern Michigan game (exhibition) tomorrow. http://t.co/oefkA
ElW
Ground-Truth Class: negative
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

13 Tweet: W.O.W Wednesday!Marni lands this Lumberjack vest for the ladies looking to bring a little Tom boy tou
ghness  http://t.co/7NyCbdJR
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: negative

14 Tweet: Activists in Deir Ezzor captured this image of Musab Bin Umair Mosque after regime forces set it on f
ire Wednesday. http://t.co/MRcoprCE
Ground-Truth Class: negative
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

15 Tweet: @karaotr You will appreciate this.. Sunday brunch coffee: Normal cup in b/g and then the BOWL of jav
a. Yowza. http://t.co/XhbtaCvm
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: positive

16 Tweet: Join me Wed for a live webcast on cost optimization for IT, for the SMB crowd. http://t.co/tyJn4RES
```

```
&lt;&lt; send your questions in! #DellWebcast
Ground-Truth Class: positive
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

17 Tweet: Special THANKS to EVERYONE for coming out to Taboo Tuesday With DST tonight! It was FUN&amp;education
al!!! :) @XiEtaDST
Ground-Truth Class: positive
SVM Prediction: positive
SVM+Lexicon Prediction: positive
Lexicon Model Prediction: negative

18 Tweet: @fatimasule That was the revelation I mentioned on sunday evening. I am still in Abj. How are u &amp;
where have u been again?
Ground-Truth Class: positive
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: positive

19 Tweet: Kim Hyung Jun - Football Team the 2nd A Match at YeongDeungPo-gu DaeRimDong [12.10.27] Credit : tlxha
h #6 http://t.co/u7mPTl0X
Ground-Truth Class: neutral
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral

20 Tweet: The audio booth is ready to blow the roof off the Comcast Center tomorrow! Are you? #MDMadness htt
p://t.co/B19fECgY
Ground-Truth Class: positive
SVM Prediction: neutral
SVM+Lexicon Prediction: neutral
Lexicon Model Prediction: neutral
```

Complete the following tasks:

- Manually annotate all of the tweets printed above:

    1. Tweet 1 Annotation Here **Positive**
    2. Tweet 2 Annotation Here **Neutral**
    3. Tweet 3 Annotation Here **Neutral**
    4. Tweet 4 Annotation Here **Neutral**
    5. Tweet 5 Annotation Here **Neutral**
    6. Tweet 6 Annotation Here **Positive**
    7. Tweet 7 Annotation Here **Negative**

8. Tweet 8 Annotation Here **Positive**
9. Tweet 9 Annotation Here **Neutral**
10. Tweet 10 Annotation Here **Neutral**
11. Tweet 11 Annotation Here **Neutral**
12. Tweet 12 Annotation Here **Negative**
13. Tweet 13 Annotation Here **Positive**
14. Tweet 14 Annotation Here **Negative**
15. Tweet 15 Annotation Here **Positive**
16. Tweet 16 Annotation Here **Positive**
17. Tweet 17 Annotation Here **Positive**
18. Tweet 18 Annotation Here **Neutral**
19. Tweet 19 Annotation Here **Neutral**
20. Tweet 20 Annotation Here **Positive**

- How many of your annotations match the ground truth labels? Do you think the datasets labels are correct? (Use your intuition)
  - **15**
- How many of your annotations match the lexicon-based model's predictions?
  - **7**
- How many of your annotations match the SVM's predictions?
  - **12**
- How many of your annotations match the SVM+Lexicon's predictions?
  - **10**
- Do you see any major limitations of the linear SVM model? Use your intuition, I will accept most answers, as long as it makes some sense. Please describe and provide examples below:

Answer Here

A major limitation that I notice with the linear SVM model is its inability to pick up on punctuation within the tweets that can drastically change the true sentiment around specific wording. Additionally, I noticed that the model takes some time to process and achieve a final result.