Introduction to Bag of Words for Binary Classification

Motivation: In this problem we provide an introduction to a real world application of the Bag of Words Model: sentiment analysis and binary classification. The student will perform binary classification on two datasets: one Yelp Review dataset partitioned by low and highly rated reviews and another dataset of Airplane Tweets classified into positive and negative sentiment. The student starts off with data exploration. Afterwards, they create a simple logistic regression model with the only feature being word count, then implement bag of words features, then explore a number of modifications to the model in order to evaluate the tangible impact of using different variations and better understand the nuances of the model.

In [67]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
import sklearn
import json

from collections import Counter
```

1. Exploring the data set

Here we explore the dataset of Yelp Reviews and Airplane Tweets. The Yelp dataset has a star rating of 1 to 5, so we extract the most polar reviews with 1 and 5 stars and classify them as -1 and 1 respectively. For Airplane Tweets, we translate the 'negative' and 'positive' sentiment labels into classes of -1 and 1. We initially load the data and do some exploratory analysis in order to learn more about the data we will be classifying and gain some insight into how to do so.

In [68]:
```
### Uncomment to use Yelp Reviews dataset
# df = pd.read_csv('yelp_academic_dataset_review.csv')
###

### Uncomment this to use the Airplane Tweets dataset
df = pd.read_csv('Tweets.csv')
###
```

We simply grab all the one star and five star data from the dataset here.

In [69]:
```
### Uncomment to use Yelp Reviews dataset
# # Get one star reviews and label them with -1
# dfNegative = df[df['stars'] == 1]
# dfNegative = dfNegative.head(10000)
# dfNegative['stars'] = dfNegative['stars'].apply(lambda x: -1)

# # Get five star reviews and label them with 1
# print("Shape of the negative input: ")
# print(dfNegative.shape)
# dfPositive = df[df['stars'] == 5]
# dfPositive = dfPositive.head(10000)
# dfPositive['stars'] = dfPositive['stars'].apply(lambda x: 1)

# print("Shape of the positive input: ")
# print(dfPositive.shape)
# dfCombined = pd.concat([dfNegative, dfPositive], axis=0)
# dfCombined = dfCombined[['stars', 'text']]
# dfCombined=dfCombined.rename(columns = {'stars':'class'})
###


### Uncomment this to use the Airplane Tweets dataset
dfCombined = df[['airline_sentiment', 'text']]
dfCombined = dfCombined[dfCombined.airline_sentiment != 'neutral']
dfCombined['airline_sentiment'] = dfCombined['airline_sentiment'].replac
e(['positive','negative'],[1, -1])
dfCombined = dfCombined.rename(columns = {'airline_sentiment':'class'})
dfPositive = dfCombined[dfCombined['class'] == 1]
dfNegative = dfCombined[dfCombined['class'] == -1]
print("Shape of the negative input: ")
print(dfNegative.shape)
print("Shape of the positive input: ")
print(dfPositive.shape)
###



# Randomly shuffling the data then dividing it into train and test sets
dfCombined = dfCombined.sample(frac=1)
print("Shape of the dataframe: ")
print(dfCombined.shape)

dfTrainset = dfCombined.head(int(len(dfCombined.index) * .8))
dfTestset = dfCombined.tail(int(len(dfCombined.index) * .2))

trainX = np.asarray(dfTrainset['text'])
trainY = np.asarray(dfTrainset['class'])

testX = np.asarray(dfTestset['text'])
testY = np.asarray(dfTestset['class'])

print('Data Frame of reviews:')


dfCombined
```

```
Shape of the negative input:
(9178, 2)
Shape of the positive input:
(2363, 2)
Shape of the dataframe:
(11541, 2)
Data Frame of reviews:
```

Out[69]:

	class	text
2271	-1	@united Flight Cancelled Flightled to BDL and ...
145	-1	@VirginAmerica I paid the premium to fly you a...
7486	-1	@JetBlue I shouldn't have to find them, they ...
7751	-1	@JetBlue no more than half an hour wait. It's ...
7462	-1	@JetBlue I waited 3 hrs for my bags and my fli...
13589	-1	@AmericanAir just want to thank you guys for t...
5166	-1	@SouthwestAir yep after two hours and thirty m...
2826	-1	@united with the purchase of my ticket i am en...
1636	1	@united you all do a wonderful job today. Got ...
5213	1	@SouthwestAir I made it! Heading to Denver, an...
3185	-1	@united stuck on Tarmac for the last hour. Can...
10426	-1	@USAirways FINALLY leaving 4 home from NC. Wil...
3855	-1	@united Can we get Any (free) upgrade because ...
3094	-1	@united 1hr delay at the start, huge queues at...
7415	-1	@JetBlue it seems almost inconceivable that si...
10432	-1	@USAirways waited for 3 hours NO LUGGAGE line ...
6741	-1	@SouthwestAir won't answer their phones #Horri...
12140	-1	@AmericanAir extremely upset that your baggage...
11757	-1	@USAirways I had a rep 10 min in who said she ...
9129	-1	@USAirways nah it's for my flight next week.. ...
10223	-1	@USAirways you can't control the weather but y...
12388	1	@AmericanAir Thanks, have emailed them. How lo...
1441	-1	@united either your staff or whoever you contr...
9239	-1	@USAirways 1,223.22 of unusable funds. Will no...
12685	-1	@AmericanAir This is getting out of hand. I ca...
2355	1	@united the upgrade to first class was a nice ...
14154	-1	@AmericanAir tomorrows flight Cancelled Flight...
2468	-1	@united bags arrived - I sure miss the custome...
13859	-1	@AmericanAir wow @AmericanAir still screw n st...
7587	-1	@JetBlue Appreciate the heads up at 10:45 that...
...	...	...
13042	1	@AmericanAir Thanks for asking On second plane...

	class	text
4596	-1	@SouthwestAir prove it, Cuz the southwest peop...
8567	1	@JetBlue Aw okay thanks
8094	1	@JetBlue you guys rock!! http://t.co/LA397zaoAY
8861	1	@JetBlue This could be the beginning of a BLUE...
12172	1	@AmericanAir Thanks so much!
12332	-1	@AmericanAir ...2/2 doesn't help me.
12750	-1	@AmericanAir how about you give us a number to...
5236	-1	@SouthwestAir, sev ppl im my office received a...
4023	-1	@united @annricord $856.81 the cost of one of ...
709	1	@united Thank you for the cheese platter and a...
14145	-1	@AmericanAir not enough push crews for JFK = 1...
1573	-1	@united missing a day of vacation to see my hu...
5612	-1	@SouthwestAir one of my bags didn't make it to...
12040	-1	@AmericanAir why would I even consider continu...
14087	1	@AmericanAir Great seats on this aircraft!
13119	-1	@AmericanAir we have been sitting on the plane...
1949	1	Very quick! TY. @united: @auciello I am sorry ...
4122	-1	@united @gg8929 so why did you Cancelled Fligh...
3155	-1	@united just curious, when are you going to to...
3217	-1	@united I'd rather have the truth about my fli...
11493	-1	@USAirways \nNot one to complain much but real...
8824	1	@JetBlue's CEO battles to appease passengers a...
11735	1	@USAirways that's why u guys are my #1 choice.
13184	1	.@AmericanAir @TyWinter it's really the small ...
2407	-1	@united I was insulted, disrespected and met w...
3778	-1	@united Nice "partners" you have. The delays k...
1523	-1	@united I'm really glad I just waited on the p...
5555	-1	@SouthwestAir just want the money I paid for e...
965	-1	@united #albanyairport delayed departure to ch...

11541 rows × 2 columns

Part A: Data Sampling

Try to run the below block multiple times to see different reviews and their respective class. Please comment below on what interesting aspects of the reviews you find associated with each class. What distinguishes between a classification of 1 and one of -1? Do so for both datasets.

In [70]:
```
sample = dfTrainset.sample()
print("Text: " + sample['text'].values[0]  + "\n")
print("Classification: " + str(sample['class'].values[0]))
```

```
Text: @united thank you for the reply. I emailed your customer care dep
artment about my experience.

Classification: -1
```

RESPONSE:

Yelp Reviews:

Airplane Tweets:

Any meaningful answer that discusses words that are more commonly used in positive reviews vs negative reviews or length analysis will suffice

Part B: Corpus Examination

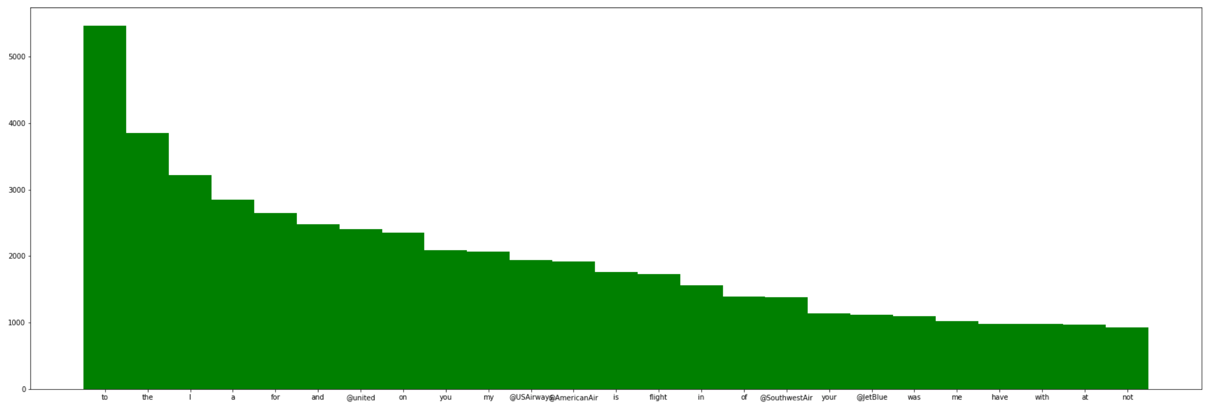
We will now look at all the text in our train dataset (corpus) in order to see what it contains. In the provided space below use a histogram to visualize the frequency of the 25 most common words. Then answer the questions that follow. Hint: The most_common function for Counters may come in handy.

In [90]:
```
allText = ' '.join(dfTrainset["text"])
words = allText.split()

wordCounts = Counter()
for word in words:
    wordCounts[word] += 1
```

In [91]:
```
print("Length of all text:")
print(len(allText))
print("Number of unique words:")
print(len(wordCounts))
### Begin Part B
mostCommon = dict(wordCounts.most_common(25))

fig, ax = plt.subplots(figsize=(30,10))
ax.bar(mostCommon.keys(), mostCommon.values(), 1, color='g')
### End Part B
```

```
Length of all text:
1007408
Number of unique words:
21770
```

Out[91]: <BarContainer object of 25 artists>



What do you notice about the most common words for both datasets? Do you think they are useful in classifying a review?

RESPONSE:

RESPONSE HERE

Look at some of the least common words below. Define the variable least common.

In [92]:
```
### Begin Part B
leastCommon = dict(wordCounts.most_common()[:-10-1:-1])
### End Part B
```

In [93]:
```
print(leastCommon)
```

```
{'ticket???': 1, 'KTA': 1, '#,': 1, 'List': 1, 'Why,': 1, 'worse...':
1, 'memphis.': 1, '1-2888155964': 1, 'request:': 1, 'son.': 1}
```

What do you notice about the least common words for both datasets? Do you think they are useful in classifying a review?

RESPONSE:

RESPONSE HERE

Part C: Identifying Unique Most Common Words of Each Classification

We now want to find the most common words in each class that are not included in the other. Basically, we find the most common words in positive reviews (class = 1) that are not in the most common set of words for negative reviews (class = -1) and vice versa. Fill out the below code and answer the following questions.

In [94]:
```
allTextPositive = ' '.join(dfPositive["text"])
wordsPositive = allTextPositive.split()

### Begin Part C
# Find the 100 most common words that are found in the five star reviews
wordCountsPositive = Counter()
for word in wordsPositive:
    wordCountsPositive[word] += 1

mostCommonPositive = dict(wordCountsPositive.most_common(100))
### End Part C



allTextNegative = ' '.join(dfNegative["text"])
wordsNegative = allTextNegative.split()

### Begin Part C
# Find the 100 most common words that are found in the one star reviews
wordCountsNegative = Counter()
for word in wordsNegative:
    wordCountsNegative[word] += 1
mostCommonNegative = dict(wordCountsNegative.most_common(100))
### End Part C

### Begin Part C
# Subtract sets in order to find the most common unique words for each s
et
positiveUnique = { k : mostCommonPositive[k] for k in set(mostCommonPosi
tive) - set(mostCommonNegative) }
negativeUnique = { k : mostCommonNegative[k] for k in set(mostCommonNega
tive) - set(mostCommonPositive) }
### End Part C

print("Most common words in negative reviews: ")
print(negativeUnique)
print()
print("Most common words in positive reviews: ")
print(positiveUnique)
```

```
Most common words in negative reviews:
{'how': 309, 'why': 341, 'phone': 267, 'Late': 374, 'No': 256, 'bag': 2
87, 'Flightled': 318, '3': 265, "can't": 379, 'hour': 355, 'or': 318,
'hold': 516, 'need': 314, 'one': 371, 'more': 266, 'what': 309, 'plan
e': 392, 'after': 342, 'delayed': 358, 'now': 521, 'over': 310, 'call':
384, 'hours': 467, '2': 513, 'because': 280, "I've": 265, 'Cancelled':
912, 'when': 422, 'still': 395, 'waiting': 286, "don't": 333, 'there':
287, 'if': 328}

Most common words in positive reviews:
{'thanks!': 47, 'Thank': 231, 'good': 75, 'you.': 77, 'love': 85, 'bes
t': 63, 'were': 46, 'thank': 204, 'you!': 129, 'great': 144, 'know': 4
7, 'fly': 54, 'appreciate': 46, 'new': 45, "I'll": 39, 'airline': 46,
'You': 62, 'Great': 48, 'Thanks': 177, 'flying': 59, '@VirginAmerica':
138, 'made': 55, 'very': 55, 'Just': 44, ':)': 96, 'Thanks!': 69, 'se
e': 44, 'guys': 76, 'really': 42, 'crew': 51, 'make': 39, 'much': 54,
'thanks': 218}
```

What do you notice about these words above? Are they more respresentative of each classification? What words do you think are good indicators of each review class? What words are not so good? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

2. Constructing and Evaluating Different Models

Part D: Baseline Model

To see the effect of the bag of words model, we first build a naive baseline model that tries to simply classification of the model purely based on the length of the review. Complete the code below and answer the following questions.

In [76]:
```
def baseline_featurize(review):
    ### Begin Part D
    # Featurize the data based on the length of the review. Hint: There
 should only be one feature.
    return np.asarray([len(review)])
    ### End Part D

def trainModel(X_featurized, y_true):
    ### Begin Part D
    # Return a logistic regression model
    model = LogisticRegression()
    model.fit(X_featurized, y_true)
    return model
    ### End Part D

def accuracyData(model, X_featurized, y_true):
    ### Begin Part D
    # Predict the data given the model and corresponding data. Return th
e accuracy
    # as the percentage of values that were correctly classified. Also p
rint a confusion
    # matrix to help visualize the error. Hint: Look at sklearn.metrics.
confusion
    y_predict = model.predict(X_featurized)
    total_num = len(y_true)
    total_correct = np.sum([1 if y_predict[i] == y_true[i] else 0 for i
in range(len(y_predict))])
    total_incorrect = total_num - total_correct
    accuracy = total_correct / total_num
    print(sklearn.metrics.confusion_matrix(y_true, y_predict, labels=[-1
, 1]))
    print(accuracy)
    ### End Part D
    return accuracy
```

In [77]:
```
### Begin Part D
# Featurize the training data and then train a model on it.
# Afterwards, featurize the test data and evaluate the model on it.
# Use the functions you made above to do so
print("Beginning Train Featurization")
featurized_data = np.array(list(map(baseline_featurize, trainX)))
print("Beginning Training")
model = trainModel(featurized_data, np.asarray(dfTrainset["class"]))
print("Beginning Test Featurization")
testFeaturized_data = np.array(list(map(baseline_featurize, testX)))
print("Accuracy:")
accuracyData(model, testFeaturized_data, np.asarray(dfTestset["class"]))
### End Part D
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1766   51]
 [ 420   71]]
0.7959272097053726
```

Out[77]:
```
0.7959272097053726
```

What did you get as your accuracy? Does that surprise you? Why or why not? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

Part E: Bag of Words Model

Now implement the bag of words featurization below based on the provided lecture. Please complete the following code segments and answer the following questions.

In [78]:
```
# We create a wordsOrdered list that contains all words in the train dat
a that show up more
# than one time. Each word count should be in its respective place in th
e feature vector.

modifiedCounter = Counter(el for el in wordCounts.elements() if wordCoun
ts[el] > 1)
wordsOrdered = [key for key, _ in modifiedCounter.most_common()]

def bag_of_words_featurize(review):
    ### Begin Part E
    # Code the featurization for the bag of words model. Return the corr
esponding vector
    reviewWords = review.split()
    vec = np.zeros(len(modifiedCounter))
    for word in reviewWords:
        if word in wordsOrdered:
            vec[wordsOrdered.index(word)] += 1
    return vec
    ### End Part E
```

Run the below script and see how well the bag of words model performs. Warning: this block may around 10 minutes to run.

In [79]:
```
print("Beginning Train Featurization")
currBagFeaturized_data = np.array(list(map(bag_of_words_featurize, train
X)))
print("Beginning Training")
currBagModel = trainModel(currBagFeaturized_data, np.asarray(dfTrainset[
"class"]))
print("Beginning Test Featurization")
testFeaturizedBag_data = np.array(list(map(bag_of_words_featurize, testX
)))
print("Accuracy:")
accuracyData(currBagModel, testFeaturizedBag_data, np.asarray(dfTestset[
"class"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1745   72]
 [ 132  359]]
0.9116117850953206
```

Out[79]:
```
0.9116117850953206
```

What was your accuracy? Does that surprise you? Why did it perform as it did? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

In [80]:
```
intermed = dict(enumerate(wordsOrdered))
wordPosition = {y:x for x,y in intermed.items()}
```

Part F: Examining Bag of Words Weights

We have provided a function that gets the weight of a word feature below in the weight vector generated from the logistic regression model with bag of words featurization. Answer the question below.

In [81]:
```
def weightOfWords(word):
    if word not in wordPosition.keys():
        print("Word does not exist in model, no weight is assigned to i
t")
        return
    return currBagModel.coef_[0][wordPosition[word]]
```

In [82]:
```
# Try different words here
weightOfWords('good')
```

Out[82]: 1.4699087976401013

List three words that have positive weights. List three that have negative weights. Explain why that makes sense. Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

Any set of words that works is sufficient. Ex: 'him', 'her', 'bad' are negatively weighted ...

Part G: Binary Bag of Words

There are times when we only want to identify whether a word is in a review or not and disregard the number of times it has shown up in the review. In this case, we find binary bag of words more useful that our regualar bag of words model. Hypothesize which model should run better given the examination of the dataset. Complete the code below and answer the questions below.

In [83]:
```
def bag_of_words_binary_featurize(review):
    ### Begin Part G
    reviewWords = review.split()
    vec = np.zeros(len(modifiedCounter))
    for word in reviewWords:
        if word in wordsOrdered:
            vec[wordsOrdered.index(word)] = 1
    return vec
    ### End Part G
```

Run the below script and see how well the bag of words model performs. Warning: this block may around 10 minutes to run.

In [84]:
```
print("Beginning Train Featurization")
currBinBagFeaturized_data = np.array(list(map(bag_of_words_binary_featur
ize, trainX)))
print("Beginning Training")
currBinBagModel = trainModel(currBinBagFeaturized_data, np.asarray(dfTra
inset["class"]))
print("Beginning Test Featurization")
testFeaturizedBinBag_data = np.array(list(map(bag_of_words_binary_featur
ize, testX)))
print("Accuracy:")
accuracyData(currBinBagModel, testFeaturizedBinBag_data, np.asarray(dfTe
stset["class"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1750   67]
 [ 129  362]]
0.9150779896013865
```

Out[84]: 0.9150779896013865

What was your accuracy percentage? Was it what you expected? How did it compare to the regular Bag of Words model? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

Part H: Bag of Words Negative Features

There are times where we also want to identify negative words as separate features instead of regular features. For example if we get a review: "The food is not good", the word "good" is used in a negative connotation and should be treated as such. Thus we make new features for the negative of each of our chosen words. Complete the code below and answer the following questions. Hint: Try doubling the size of the feature vector.

In [85]:
```
def bag_of_words_neg_featurize(review):
    ### Begin Part H
    reviewWords = review.split()
    vec = np.zeros(len(modifiedCounter)*2)
    isNegative = False
    for word in reviewWords:
        if word in wordsOrdered:
            if isNegative:
                vec[wordsOrdered.index(word)+len(modifiedCounter)] += 1
            else:
                vec[wordsOrdered.index(word)] += 1
            isNegative = False
        if "n't" in word or word == "not":
            isNegative = True
    return vec
    ### End Part H
```

Run the below script and see how well the bag of words model performs. Warning: this block may around 10 minutes to run.

In [86]:
```
print("Beginning Train Featurization")
neg_data = np.array(list(map(bag_of_words_neg_featurize, trainX)))
print("Beginning Training")
negModel = trainModel(neg_data, np.asarray(dfTrainset["class"]))
print("Beginning Test Featurization")
testFeaturizedNeg_data = np.array(list(map(bag_of_words_neg_featurize, t
estX)))
print("Accuracy:")
accuracyData(negModel, testFeaturizedNeg_data, np.asarray(dfTestset["cla
ss"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1745   72]
 [ 133  358]]
0.9111785095320624
```

Out[86]: 0.9111785095320624

How did this model perform? Is it as expected? Why did it perform this way? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

Part I: Negative Binary Features

Follow the code below and answer the questions below for combining the two features we worked on.

In [87]:
```
def bag_of_words_neg_binary_featurize(review):
    ### Begin Part I
    reviewWords = review.split()
    vec = np.zeros(len(modifiedCounter)*2)
    isNegative = False
    for word in reviewWords:
        if word in wordsOrdered:
            if isNegative:
                vec[wordsOrdered.index(word)+len(modifiedCounter)] = 1
            else:
                vec[wordsOrdered.index(word)] = 1
            isNegative = False
        if "n't" in word or word == "not":
            isNegative = True
    return vec
    ### End Part I
```

Run the below script and see how well the bag of words model performs. Warning: this block may around 10 minutes to run.

In [88]:
```
print("Beginning Train Featurization")
negbin_data = np.array(list(map(bag_of_words_neg_binary_featurize, train
X)))
print("Beginning Training")
negBinModel = trainModel(negbin_data, np.asarray(dfTrainset["class"]))
print("Beginning Test Featurization")
testFeaturizedNegBin_data = np.array(list(map(bag_of_words_neg_binary_fe
aturize, testX)))
print("Accuracy:")
accuracyData(negBinModel, testFeaturizedNegBin_data, np.asarray(dfTestse
t["class"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1750   67]
 [ 132  359]]
0.9137781629116117
```

Out[88]:
```
0.9137781629116117
```

Was the result as expected? Why or why not? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

3. Extra Credit

Part J (OPTIONAL): Enhanced Model

In order to get extra credit, Try to create some sort of featurization below that will reach an accuracy of .97 or higher for either model. Ideas to keep in mind are the Bigram model that was discussed in the notes that takes consecutive words into account as well as methods to increase the number of features we use. Good luck!! HINT: You can combine additional features like length with existing bag of words features.

In [ ]:
```
def bag_of_words_extra_credit_featurize(review):
    ### Begin Part J
    # User solution!
    ### End Part J
```

In [ ]:
```
print("Beginning Train Featurization")
ExtraBagFeaturized_data = np.array(list(map(bag_of_words_extra_credit_fe
aturize, trainX)))
print("Beginning Training")
ExtraBagModel = trainModel(ExtraBagFeaturized_data, np.asarray(dfTrainse
t["class"]))
print("Beginning Test Featurization")
testFeaturizedBinBag_extra = np.array(list(map(bag_of_words_extra_credit
_featurize, testX)))
print("Accuracy:")
accuracyData(ExtraBagModel, testFeaturizedBinBag_extra, np.asarray(dfTes
tset["class"]))
```

What features did you add? Why did you do so? What was your accuracy percentage?

RESPONSE:

ONLY RUN BELOW CODE IF YOU ARE ON THE YELP DATASET

4. Evaluating Yelp Model with Less Polar Data

Now we will be performing a similar analysis on the Yelp Dataset but including both 1 star and 2 star reviews as the negative class and 4 star and 5 star reviews as the positive class. This way there will be less of a clear divide between the two classes and students should see how adapting the bag of words model can prove beneficial.

In [ ]:
```
# Get one star reviews and label them with -1
dfOnes = df[df['stars'] == 1]
dfOnes = dfOnes.head(10000)
dfOnes['stars'] = dfOnes['stars'].apply(lambda x: -1)


dfTwos = df[df['stars'] == 2]
dfTwos = dfTwos.head(10000)
dfTwos['stars'] = dfTwos['stars'].apply(lambda x: -1)

# Get five star reviews and label them with 1
print("Shape of the ones input: ")
print(dfOnes.shape)

dfFives = df[df['stars'] == 5]
dfFives = dfFives.head(10000)
dfFives['stars'] = dfFives['stars'].apply(lambda x: 1)

dfFours = df[df['stars'] == 4]
dfFours = dfFours.head(10000)
dfFours['stars'] = dfFours['stars'].apply(lambda x: 1)

print("Shape of the fives input: ")
print(dfFives.shape)
dfCombined = pd.concat([dfOnes, dfTwos, dfFours, dfFives], axis=0)
dfCombined=dfCombined.rename(columns = {'stars':'class'})
dfCombined = dfCombined.sample(frac=1)

dfTrainset = dfCombined.head(int(len(dfCombined.index) * .8))
dfTestset = dfCombined.tail(int(len(dfCombined.index) * .2))

trainX = np.asarray(dfTrainset['text'])
trainY = np.asarray(dfTrainset['class'])

testX = np.asarray(dfTestset['class'])
testY = np.asarray(dfTestset['class'])

print('Data Frame of reviews:')
dfCombined
```

Part K: Data Sampling of the 2 and 4 star reviews

In [ ]:
```
sample = dfTwos.sample()
print("Text: " + sample['text'].values[0]  + "\n")
print("Class: " + str(sample['class'].values[0]))
```

In [ ]:
```
sample = dfFours.sample()
print("Text: " + sample['text'].values[0]  + "\n")
print("Class: " + str(sample['class'].values[0]))
```

In [ ]:
```
allText = ' '.join(dfCombined["text"])
words = allText.split()

wordCounts = Counter()
for word in words:
    wordCounts[word] += 1

modifiedCounter = Counter(el for el in wordCounts.elements() if wordCoun
ts[el] > 1)
wordsOrdered = [key for key, _ in modifiedCounter.most_common()]
```

Part L: Baseline Model

In [ ]:
```
print("Beginning Train Featurization")
currBaselineFeaturized_data = np.array(list(map(baseline_featurize, trai
nX)))
print("Beginning Training")
currBaselineModel = trainModel(currBaselineFeaturized_data, np.asarray(d
fTrainset["class"]))
print("Beginning Test Featurization")
testFeaturizedBaseline_data = np.array(list(map(baseline_featurize, test
X)))
print("Accuracy:")
accuracyData(currBaselineModel, testFeaturizedBaseline_data, np.asarray(
dfTestset["class"]))
```

What was your accuracy percentage? Was it what you expected? How did it compare to the regular Bag of Words model? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

Part M: Bag of Words Model

In [ ]:
```
print("Beginning Train Featurization")
currBagFeaturized_data = np.array(list(map(bag_of_words_featurize, trainX)))
print("Beginning Training")
currBagModel = trainModel(currBagFeaturized_data, np.asarray(dfTrainset["class"]))
print("Beginning Test Featurization")
testFeaturizedBag_data = np.array(list(map(bag_of_words_featurize, testX)))
print("Accuracy:")
accuracyData(currBagModel, testFeaturizedBag_data, np.asarray(dfTestset["class"]))
```

What was your accuracy percentage? Was it what you expected? How did it compare to the regular Bag of Words model? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

Part N: Binary Bag of Words Model

In [ ]:
```
print("Beginning Train Featurization")
currBinBagFeaturized_data = np.array(list(map(bag_of_words_binary_featurize, trainX)))
print("Beginning Training")
currBinBagModel = trainModel(currBinBagFeaturized_data, np.asarray(dfTrainset["class"]))
print("Beginning Test Featurization")
testFeaturizedBinBag_data = np.array(list(map(bag_of_words_binary_featurize, testX)))
print("Accuracy:")
accuracyData(currBinBagModel, testFeaturizedBinBag_data, np.asarray(dfTestset["class"]))
```

What was your accuracy percentage? Was it what you expected? How did it compare to the regular Bag of Words model? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

Part O: Negative Bag of Words Model

In [ ]:
```
print("Beginning Train Featurization")
neg_data = np.array(list(map(bag_of_words_neg_featurize, trainX)))
print("Beginning Training")
negModel = trainModel(neg_data, np.asarray(dfTrainset["class"]))
print("Beginning Test Featurization")
testFeaturizedNeg_data = np.array(list(map(bag_of_words_neg_featurize, t
estX)))
print("Accuracy:")
accuracyData(negModel, testFeaturizedNeg_data, np.asarray(dfTestset["cla
ss"]))
```

What was your accuracy percentage? Was it what you expected? How did it compare to the regular Bag of Words model? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets:

Part P: Negative Binary Bag of Words Model

In [ ]:
```
print("Beginning Train Featurization")
negbin_data = np.array(list(map(bag_of_words_neg_binary_featurize, train
X)))
print("Beginning Training")
negBinModel = trainModel(negbin_data, np.asarray(dfTrainset["class"]))
print("Beginning Test Featurization")
testFeaturizedNegBin_data = np.array(list(map(bag_of_words_neg_binary_fe
aturize, testX)))
print("Accuracy:")
accuracyData(negBinModel, testFeaturizedNegBin_data, np.asarray(dfTestse
t["class"]))
```

What was your accuracy percentage? Was it what you expected? How did it compare to the regular Bag of Words model? Answer for both datasets.

RESPONSE:

Yelp Reviews:

Airplane Tweets: