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 [3]:

```
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 [4]:

```
### 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 [113]:

```
### 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'].replace(['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:
(10000, 9)
Shape of the positive input:
(10000, 9)
Shape of the dataframe:
(20000, 2)
Data Frame of reviews:
```

Out[113]:

	class	text
21778	1	I struggled to pick between the 2 brazillian s...
9317	1	Upon arriving we were greeted by an enthusiast...
58457	-1	I have rented a car from these guys 3 times in...
8956	1	This guys awesome. Shows up quickly, 24hours a...
23848	-1	If this place was in the south, it would be ou...
68173	-1	Bennett Property Management is awful. Working ...
43259	-1	The good: I read some reviews that the cutlets...
37680	-1	Just not the same place it use to be since 200...
5177	1	Lazeez was absolutely fantastic! We had the gy...
9151	1	I seriously LOVED this place!\n\nWe started wi...
21948	1	We had the bacon wrapped spicy meatloaf and be...
3583	1	Cake batter, vanilla snow & peanut butter! \n...
6740	1	If you're an old-school video gamer, this is d...
6261	-1	Hate this place bad customer service the baske...
18708	1	Tried Mom's for the lunch special: $6 (tax inc...
12185	1	Great drinking spot!! Very Divey and down home...
14182	1	Delicious fast-casual restaurant. Everything t...
16867	1	Eric was so sweet. He did such an excellent jo...
16175	1	I met an angel today at Chipotle and I am grat...
35257	-1	I purchased a Groupon for this salon so I coul...
17322	1	Erica is great! She has been cutting hair for...
63080	-1	First of all I am not happy with their restaur...
28856	-1	Disaster:\n\n1. Food is nowhere near the quali...
37618	-1	Decided to give this place a try after hearing...
2313	1	I have become a huge fan of this little downto...
6582	-1	Not recommended. I booked and was told I would...
23126	-1	This is the place where all the rejects go whe...
17316	-1	I was so excited to try this place, as I had h...
12366	-1	Has to be the WORST fish and chips I have ever...
1952	1	Zoës is my favorite place to go. The food is a...
...	...	...
10986	1	I don't usually write reviews, but in this cas...
39110	-1	DO NOT USE THIS COMPANY!!!!the customer servic...
3990	1	I spend just about every day here, and I love ...
17389	1	I've been going to Color Me Glam Boutique for ...
14312	1	I ordered the chopped salad for my wife but I ...
9424	1	Awesome food and great service! Get the chicke...
2093	-1	The people here are extremely rude. No one wil...
22225	1	Absolutely loved it. The food was very tasty a...
12057	1	Stumbled across this place on my last day in V...
71227	-1	I will have to ditto Tania S's review. We did...
11443	1	Amazing!!! Had the stuffed french toast with c...
20321	-1	This store doesn't know how to make a burrito....
54595	-1	Meh. Just meh. In a town like this full of au...
27840	-1	Went into Chili's for dinner. Most of the tab...
19961	-1	From Le journal de Montreal http://www.journal...
10105	-1	Totally rudeee the lady who answered the phone...
16265	1	I've been to the Beverly Hills, Chicago and La...
1683	-1	Worst Sprint store in town. Their staff are th...
68647	-1	I ordered a bed frame over 3 months ago and it...
11782	-1	Despite how nice a business is, if it's unethi...
17492	-1	Don't know how this outfit got such a high rat...
18651	-1	My first experience was probably the worst Sta...
42777	-1	I have faithfully been a customer twice a week...
58672	-1	During my 45.00 oil change apparantly they neg...
18653	-1	Thought they might put some effort forth since...
19562	1	Beyond delicious cakes! The cream cheese fros...
21889	-1	I work at the mall and since they've opened I ...
12933	1	I went to autonation to purchase a used car I ...
16941	-1	Worst service ever that I experienced at wings...
65731	-1	If you want to pay for water and get bad servi...

20000 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 [137]:

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

```
Text: Orlando was very helpful and I was there a couple of times and he was very patient, I had trouble making up

Classification: 1
```

RESPONSE:¶

Yelp Reviews: Reviews classified as negative have words such as 'rude', 'poor', 'avoid', and other words with negative implications. Additionally, even words like 'waiter' tend to show up in negative reviews because they generally mention things the waitter did wrong. On the other hand, reviews classified as positive contained words like 'nice', 'bomb', and 'helpful'. However, length doesn' seem to be as good an indicator as in the Airplane Tweets dataset.

Airplane Tweets: Negative tweets seem to have a higher likelihood of having words with negative connotation like 'not' and 'worst,' as well as more punctuation such as quotation marks. They also seem to have a longer length on average. Positive tweets on the other hand have words like 'great' and even ':)' as well as tending to be shorter and having more exclamation marks.

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 [115]:

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

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

In [116]:

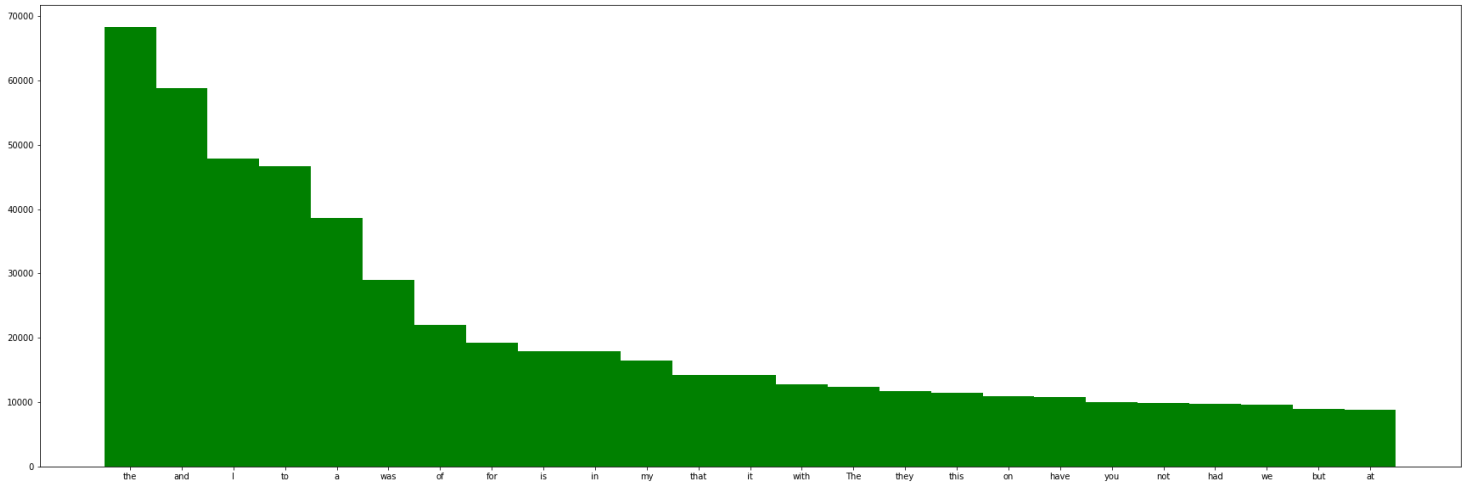
```
print("Length of all text:")
print(len(allText))
print("Number of unique words:")
print(len(wordCounts))
### Begin Part B
mostCommon = dict(wordCounts.most_common(25))

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

```
Length of all text:
9326061
Number of unique words:
87831
```

Out[116]:

```
<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:¶

The most common words for both datasets seem to mainly contain stop words like 'the', 'is', 'and', and 'a'. The Twitter dataset also has '@united' and '@AmericanAirlines' as well as other corporations in its most common words. This makes sense as they are responding to those companies. These words do not seem to be a large predictor of class as they are probably widely used in both negative and positive reviews, as they are common in normal English speak and Tweets.

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

In [117]:

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

In [118]:

```
print(leastCommon)
```

```
{'Strip.....wrong!).': 1, '(grissly': 1, "flavor's": 1, 'chicken".': 1, 'Infortunetly': 1, 'faces!': 1, 'converter
```

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

RESPONSE:¶

The least common words for both datasets seem to mainly be gibberish and badly spelled or formatted words such as 'KTA', 'grissly', and '1-2888155964'. There are also many other words that are just as common with the same number of appearances: 1. These words do not seem useful in classifying a review because they are unlikely to show up again. Moreover, it may take away from other words in training by getting assigned a high magnitude weight in its lone appearance even if it will likely not show up again.

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 [119]:

```
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 set
positiveUnique = { k : mostCommonPositive[k] for k in set(mostCommonPositive) - set(mostCommonNegative) }
negativeUnique = { k : mostCommonNegative[k] for k in set(mostCommonNegative) - 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:
{'then': 2142, 'said': 3075, 'order': 2016, 'did': 2248, 'could': 2290, 'ordered': 1768, 'came': 2090, 'no': 3909,

Most common words in positive reviews:
{'also': 1553, 'love': 1544, 'best': 1774, 'staff': 1198, 'some': 1460, 'great': 3288, 'recommend': 1257, '-': 137
```

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: The most common words in negative reviews not in the positive reviews include 'asked', 'ordered', 'didn't', and 'customer'. This is because negative reviews tend to describe very specific situation where something goes wrong, as well as other negative words like 'never' and 'didn't'. The most common words in positive reviews include words such as 'love', 'best', and 'recommend', which all clearly demonstrate positive senttiment. So these words overall seem to be good indicators for each class, as opposed to the words we saw before. They also show up very often in reviews so will be good at predicting class.

Airplane Tweets: Some words that appear often in negative tweets but are not in the most often list for positive ttweets are 'how' and 'why'. This makes sense as a lot of complaints involve asking questions about something that went wrong. Other words in tthis set are 'Late', 'delayed', and 'Cancelled', which makes sense as these are negative words that would imply dissatisfaction and a negative response. On the other hand, some words that appear in the most common list for positive tweets but not negative are 'thanks', 'love', 'best', and so on. Again, these words make sense as they have a positive connotation and thus would appear more often in positive sentiment tweets. As a result, these words seem to be good indicators of negative/positive sentiment as they appear more often in their respective classes, and thus should be weighted as such in our model. On the other hand, stop words and words that do not appear often as above would no be great predictors of class.

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 [16]:

```
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 the accuracy
    # as the percentage of values that were correctly classified. Also print 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 [121]:

```
### 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:
[[ 913 1108]
 [ 520 1459]]
0.593
```

Out[121]:

```
0.593
```

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

RESPONSE:¶

Yelp Reviews: The accuracy for this baseline model is 0.593. This accuracy makes a lot of sense as length did not seem to be a major differentiator between positive and negative reviews, but it does have a non-insignificant role as a predictor, since it was better than a random accuracy of 0.5.

Airplane Tweets: The overall accuracy for the baseline model with the only feature of length was 0.796. This accuracy was actually surprisingly higher than expected as while length of tweet seemed somewhat correlated with sentiment, I did not realize it would be this significant.

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 [122]:

```
# We create a wordsOrdered list that contains all words in the train data that show up more
# than one time. Each word count should be in its respective place in the feature vector.

modifiedCounter = Counter(el for el in wordCounts.elements() if wordCounts[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 corresponding 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 [123]:

```
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"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1932   89]
 [  89 1890]]
0.9555
```

Out[123]:

```
0.9555
```

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

RESPONSE:¶

Yelp Reviews: This bag of words model achieved a test accuracy of 0.9555. This was pretty surprising as it seemed to do much better than expected, without the model leaning to favoring a class as evidenced by the confusion matrix.

Airplane Tweets: The basic bag of words model achieved an accuracy of 0.912. This was not particularly surprising as it seemed that there were words as described before that were good indicators of sentiment, so those probably played a large role in getting a decent accuracy.

In [124]:

```
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 [125]:

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

In [146]:

```
# Try different words here
weightOfWords('slow')
```

Out[146]:

```
-1.2448829917949824
```

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:
'great': 1.831
'recommend': 0.479
'love': 1.282
'hate': -0.536
'slow': -1.245
'ordered': -0.762

Airplane Tweets:
'good': 1.467
'amazing': 1.608
'fast': 0.515
'delayed': -1.470
'bad': -0.55
'terrible': -1.142
These weights make sense as since we described above, words with positive connotation that we would relate to a positive sentiment seem to have positive weights, some of them larger in magnitude than others, and words with negaive connotations one would associate with a bad experience have negative weights to guide the response to a negative classification.

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 [127]:

```
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 [128]:

```
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"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1926   95]
 [ 101 1878]]
0.951
```

Out[128]:

```
0.951
```

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: The binary bag of words model here resulted in a training accuracy of 0.951. This isn't too far off so while somewhat unexpected it is not a big shock. This is likely because multiple repetitions of the same word actually should have lead to a larger weight and altered the prediction.

Airplane Tweets: The accuracy here was 0.915 for binary bag of words, higher than that of original bag of words. This makes sense as it is possible that a negative word that appears multiple times will be weighted lower if it appears in a negative response during training, so binary bag of words maintains a more accurate sense of the sentiment of a word.

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 [13]:

```
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 [130]:

```
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, testX)))
print("Accuracy:")
accuracyData(negModel, testFeaturizedNeg_data, np.asarray(dfTestset["class"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1935   86]
 [  93 1886]]
0.95525
```

Out[130]:

```
0.95525
```

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

RESPONSE:¶

Yelp Reviews: The negative bag of word featurization had a test accuracy of 0.955. This is basically the same as normal bag of words and the model itself did not pick on any differences, which was somewhat unexpected again but is probably a function of the dataset itself.

Airplane Tweets: This model actually did very similar to normal bag of words with an acccuracy of 0.912. This was somewhat unexpected as not negating words like 'not good' and treating them as an instance of not and an instance of good was thought to have a bad impact on test accuracy. However, in this case it did not, potentially due to a more advanced data cleaning.

Part I: Negative Binary Features¶

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

In [14]:

```
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 [133]:

```
print("Beginning Train Featurization")
negbin_data = np.array(list(map(bag_of_words_neg_binary_featurize, trainX)))
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_featurize, testX)))
print("Accuracy:")
accuracyData(negBinModel, testFeaturizedNegBin_data, np.asarray(dfTestset["class"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1932   89]
 [  98 1881]]
0.95325
```

Out[133]:

```
0.95325
```

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

RESPONSE:¶

Yelp Reviews: 0.953 This result was as expected at a happy medium in between the binary and negation modifications for the feaures, albeit still worse than the normal bag of words. As can be seen, for some datasets, especially those with clear divides between classes and strong word indicators, bag of words without any modifications performed better. You will see in the next part an example with more nuance (including 2 and 4 stars) where it helps to make these modifications. As usual, both should be tried and tested on a validation set to determine which is better.

Airplane Tweets: As expected, this bag of words model combining both the negation and binary modifications did right in the middle of the two independently with an accuracy of 0.914. This makes sense, although it appears that including negated features had a negative effect on the binary bag of words model, potentially due in part to the specific dataset we generated.

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_featurize, trainX)))
print("Beginning Training")
ExtraBagModel = trainModel(ExtraBagFeaturized_data, np.asarray(dfTrainset["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(dfTestset["class"]))
```

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

RESPONSE:¶

Any response here that both obtains an accuracy above 0.97 on a dataset and discusses either cleaning the data better or combining additional features such as length of response, number of exclamation marks, number of question marks, or additional sentiment dictionaries such as The Sentiment Lexicon from the University of Pittsburgh receives full extra credit here.

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 [5]:

```
# 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'})
dfTwos=dfTwos.rename(columns = {'stars':'class'})
dfFours=dfFours.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['text'])
testY = np.asarray(dfTestset['class'])

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

```
Shape of the ones input:
(10000, 9)
Shape of the fives input:
(10000, 9)
Data Frame of reviews:
```

Out[5]:

	business_id	cool	date	funny	review_id	class	text	useful	user_id
117895	1zsM7weLS8fNHomDAqCh4Q	0	2017-07-30 21:01:16	0	8HUD9mFT5ulcXWsJ0tvlHA	-1	Wait for a table wasn't bad for a Sunday brunc...	0	F3wv7xhvjQFHAsD6mubsIQ
44320	yBGqf8l9NOM3vZFnlxcgzQ	1	2010-07-09 18:55:34	3	cVtFum2XFprf5wVQhWB6WA	-1	What a disappointment! I was all excited to s...	10	6b4yYexmPQglmNdNum531Q
73075	K5sUVFSGFEZosixSXgx5sw	0	2018-06-10 21:08:51	0	-5Qc2230M8_Olotphs7BuA	-1	Burger was subpar and the fries were cold, did...	1	zeHuIVREdQtAaiJRpb4low
22447	luW6aZ8XYtxkdvHJoVHlfw	0	2011-01-09 21:18:32	0	olmjKR6uzlpG-8SQlcjdHw	1	I love aloha salon! Not only their service is ...	1	QAdtpOZjG84043dqtXfESg
30171	9Jo1pu0y2zU6ktiwQm6gNA	0	2017-06-07 02:31:32	0	k_1ITZFaoCA-FtIrBo0LvQ	1	I only had their spicy poke bowl, but I went t...	0	EBkNxxo181SpO1CYinI3xg
94841	nxlgrjTdUhvTg_3Ai0JKqw	0	2017-09-10 22:25:30	0	gX8paNzzNOH_8RCg2gKlhA	-1	Not what I expected. It is ordinary, nothing r...	0	kjzg7jTm6tWhETeJ3xDqZw
14195	4lGaWH9jUYMtP2uHIFEqFQ	0	2014-04-01 17:06:26	0	P30fI8uVkPFmQCLJ0PqqHA	1	This is actually one of the easier airports to...	1	sh3V6hnNMLVV9YwFA9QNaw
18383	8nDEOGVVvReXFJ2zjPh4Pw	0	2013-05-09 15:57:45	0	7ObEq54s3BVYa4k0kW1Ktg	-1	Not the best of attitudes, I thought this plac...	1	Y8_yCJ7N7LyqqJ-rSxd4iQ
13082	x8h3A9dOKux99qAp6Bakbg	0	2010-11-22 20:40:47	1	40lOk-VBxiq7d-KUlUlC9w	-1	How is In the Company of Thieves? It depends ...	4	TIll8A0a_GqegEm4h3yiHw
101896	6ZlHxvFTHC1pvAzAS0uLDA	0	2015-07-30 00:43:29	1	MybJ5GnsoSSDX9P8Ydoc2g	-1	I ordered a #2 pork roll sandwich. I nearly ...	0	oAJzUPW-XtpEQyle_QXLkA
16007	J7fekWie4ZwR03AZVBisFw	0	2015-01-04 01:37:20	0	DFrsl3rLHQyDm1m6jvU18A	-1	My husband and I were staying at a hotel near ...	0	go0FTv1slZ128s_Y24lgdw
32213	KalAJyO0Zpg3K1wVwYXBHA	0	2014-06-13 22:04:03	0	snl4SU-FQB9V6NEzisTbcQ	-1	Bad experience! Blackened salmon sandwich app...	2	0lZTY9OnTEc5gaBPRGvgeA
75878	iuSq6jpt-r-7JxqXE5hK6g	0	2011-07-04 13:17:31	1	U1Zs5yQi0j_ab2LpWncTsA	-1	I've been to Heart and Soul two times and both...	3	NUqAqRPyWOgPDdMdhtSebw
74	4ZbRwCB9oGibxK21MUZKHA	0	2018-08-04 22:27:08	0	E5iXQGD_x_Fyee1kOKgGTw	-1	I went into this store yesterday and it was ho...	1	ffMc-TmUUhEbj5LLNILB4A
27338	_iGvLfEsqDwPUxRUAe6tUw	0	2015-02-03 03:01:22	0	DHXZOvG6urLihqCsdL59Lw	-1	Poor salesmanship, deceptive during trade-in ...	1	EgbeJrycEtSkps8UbJRpaQ
4958	6TvB1PZCY1Ap2S-yNJPs4w	0	2018-09-14 04:11:12	0	u_OpkjRC-wY8x16dmjgGew	-1	Aaliyah at the restaurant was super rude . May...	0	nh6YUBtU-azoYVKDK8O7KQ
7059	tWiFat101lD5w_wgAPMXhA	0	2014-12-18 04:20:13	0	T1yrvyOcdsvm0wOMNSrZww	1	This place is an institution. I have been goin...	0	Owo6mggrsd4cuKlC0wWQ6w
39270	LGEIsxeJQATo9J1IA-TEdA	1	2018-07-09 01:36:28	1	mbn_GyffLzGUTFeWwT06Xw	-1	This place used to be 5 stars. I've been goin...	2	Z8dN-_8IMZ3FnHg2AhGlXA

	business_id	cool	date	funny	review_id	class	text	useful	user_id
14418	nvBhlpH8TWbCl70_30gqxg	0	2017-12-16 00:17:09	1	iUkMdJ3mdbCxhhySwIHT2A	-1	Why is this theater so filthy? Dirty! Trash an...	0	1_CkjgX5iGjbVpW8O-GU5Q
63863	3GfdCuI0YCc5U3rLLLPHUw	0	2016-03-18 16:50:10	0	Q1gJnlSBzfd6sxw39jjuNA	-1	I would have given the place a rave review if ...	0	AdgrOZj2hrh0HlGyu7JWHg
61884	u_vPjx925UPEG9DFOAAvFQ	0	2015-07-07 17:43:48	0	CLis0AoUs5Coi_fAye16tQ	-1	Very disappointed, we stayed on the 14th floor...	0	SENiX02vQAUqFUB-NsITmQ
7580	_orp_eadrrigWixdMfpB9A	0	2016-10-26 22:56:58	1	a1E8gNIp1bs9lkUWr8m4ZA	1	Dr. McCoy is awesome and his team is also very...	0	M0UiTAlVZx317KGSZ5lxLg
49311	4kDLEb1OgE7IrO54lCkn3A	0	2018-07-26 20:15:53	0	fmlVWFzycbqQDqkMlp88Ew	-1	Went with a few friends since everywhere else ...	0	lmiPjQTtcXq400p6djLrtA
50477	YHXczxm4W3BkGT-z7vZBBw	0	2015-10-18 14:58:10	0	i5WYsZnf-r_Te_932rZwdQ	-1	I was pretty excited to hear about a new Chati...	2	zWWcik1fRPZviBCQLC26FQ
97933	uh7DUWtPoZkuEE05fghJ_w	0	2016-12-22 05:53:56	0	LCBBrcyQVwOoDmDEG51t4Q	-1	Chic.... to enjoy ambiance this is the place	0	2qkXXaDBo_MdSBHj8Mdj0g
11561	i6gKYG_YFuF5o90jPuWYvw	0	2017-03-15 02:47:56	0	QaKD8EQaiwHLAuCO42KN8g	-1	Giving them 2 stars because they have made the...	0	zyg4-MFtfPWmwucVazSjfw
25492	Rfu2q_Jc0fAGP_5eclar3g	0	2012-05-29 04:13:45	0	uHnCz4_jvgcDruzb8eItoA	1	Let me say this... my total package cost me $4...	14	N_8Ad4o7pgBbHDG7AC11wA
28989	ppobG2P-YmqTPy0s4MTRtA	0	2016-07-15 00:51:47	0	MlyUSNbTES7bxh6oHL5zmA	-1	Maybe we were here on an off-day, but my party...	1	No1Qn10ByvTfDVUKDUB3aA
11990	eaNenRk_liZBERFFLCXqqQ	0	2016-06-12 06:28:30	0	aBD1oLFaTN7LAcvOVcfHaw	1	I don't usually do reviews but this is 5 stars...	0	Ng5lLOfZjl-pbPbnLw5EmA
11376	Os1n1_idfw9vv9kwULGJnQ	0	2012-12-04 04:52:31	0	HMhDTuz5mD6Gv9JpYm3SlQ	1	Someone please explain what I would have done ...	0	BA46YFb2JxWvz1c1ApZ9ZQ
...	...	...	...	...	...	...	...	...	...
110664	ijg7qQCYnhUWBd5JUxbfvA	0	2018-08-05 20:26:03	0	_R052l9dVVKd80aCK-BqWA	-1	Food was great. $5 Bloody Mary was Good. Beer ...	0	mGHG6Qh6RQh4dK_eR0WRsg
6703	oz-U184llqjVIt398hLntQ	1	2015-03-18 19:25:15	1	OPZKeZyhvoShzu6LRQ3kig	1	As a gay male walking into this restaurant I m...	1	89ExXwHanHBxx04yMSZmNw
18340	dPGs5b0N9MarZjVgQVelGQ	0	2017-11-07 02:16:32	0	7_D5oAsiAgFMWQfkvrgv4Q	1	Very nice place. Service tends to be very good...	0	vOPnC4XPRSknykbxYIG8Hw
48086	u_vPjx925UPEG9DFOAAvFQ	0	2018-04-05 02:52:10	0	Ke_BLSjY6V12i1D5s3Ieeg	-1	After a long drive we got to the hotel on 3/30...	0	Qhmk2ieq8RpckAV8x2cD-Q
38772	1lLJDy73uBp_l2LJ0B0Fkw	0	2011-02-16 16:50:15	0	2UvBQ5XpA9FAvf9hIDSJaw	1	I have to admit, there is absolutley nothing ...	0	jx4_0MDOY-FW4G7yW49WxA
44664	0j7VWa5uGebj_94C1K2i2g	0	2017-07-08 20:24:33	0	oGGqWgyeNcOLYmUWfDPQWA	-1	GOD I HATE WALMART, and this one is no excepti...	2	o9MwJLcMUJ48LRVIUmr6BQ
17089	3GfdCuI0YCc5U3rLLLPHUw	1	2014-04-21 09:38:08	0	ndtKjKcT8Biykk3Jeb3Dag	1	One of the newer izakayas in the valley. They ...	2	gqpnDoFx42U5NdtejYfKOg
8337	KUOa1acSFn6DkO9jp-sn6A	0	2017-06-06 18:52:41	1	2G7GKwpWJJviVErk6-Csbw	-1	This Famous Dave's was awful! The only reason...	1	w8rOqeKZCPtDu-DXXSZIwQ

	business_id	cool	date	funny	review_id	class	text	useful	user_id
18096	e1SSSU7lUrC3R_cmw4vW9g	1	2014-11-13 13:29:04	0	D1lxK6Vcztw3P3CKyLek2A	1	Un très beau et accueillant café, la meilleure...	0	TuSA83Ob6un1m0LXl2JHnQ
1960	k4jX-Xe9dFu2pmdnlrcwgA	0	2013-12-28 21:50:03	0	UUSHvjTqoQ6bPx69XtDF_g	-1	My husband and I went here after stopping at t...	0	PjlYINWKAxdWvL1zz4UlJg
18942	4gwh0q7JsdzHIm_pdYUlZg	13	2013-01-03 17:18:44	0	SWEBe2_Z_Stc2R8Hu0C2LA	1	impeccable service. delectable food. this plac...	12	zSwb7qNpSgU3ekHMpiHsOA
33982	F2pfjAZ_3dMTGCKv6c5wOw	0	2014-07-28 21:25:37	0	2j9o2wkdoC4hU5ELdKKsxw	1	Have been wanting to go here for quite some ti...	0	j2FfSIomcaNYvv8ywEIzQA
101987	seYvTmOZGJ2IAMdfQa8pkg	0	2014-05-10 20:10:11	0	RMQ9PFcEDotH8pf0FvI4mg	-1	This place needs to make up its mind! Sometime...	1	VFWymyNU_Bp8VCl9_RD8og
50685	YidWQ-CipHgGpA8jygsPpQ	0	2015-01-21 02:30:15	0	rHlUTLFvQgguRiK1xoF_LQ	-1	Had an appointment to get my dogs bathed at th...	0	2eNy5pf5ctWxouGGaASTOQ
15252	Te8ubqIn8aH2uplVhvMntw	0	2017-01-14 22:09:51	0	Ltfz2_Tibu0LQVp7PJYVeQ	1	The gelato tastes great. The price did not jus...	0	2pRDsLUqTK0o9ePcliyy_A
6653	hUgaylJ8gR3BNMcFiASQSA	0	2014-11-30 01:37:56	0	QiXtP62NeBMnqri3d8oz-g	1	Wow! Matt Lemm from Funny Magic For Kids has ...	3	SBhmnRD-w_xgQ-RQRRCz5w
69257	ShUh_MMkaVp_KXCtNjPvXA	0	2012-10-12 13:58:54	0	vfkijFVcBvlcNzcMTqp-ZQ	-1	The space is nice. The dry-rub ribs were good,...	0	7tRj6BKT1tTlZ1AbmZWAmw
16237	vDqKDzGuHXBznrC1HlgW1A	0	2018-09-25 17:47:01	0	DJyXRQl7rmqAnQGnq87IoA	-1	I'm very upset with this practice. I have trie...	0	-fwrHX9L3a0IlU3wkhxd2w
36281	ME6D5nLlfG0xZCtICOYDuw	0	2012-01-04 15:28:11	0	26hEOE4jqk3RmrfeAWfneg	1	Great decor and awesome food but some dishes h...	0	8x-lBENayDizUL8wXlTyOg
10845	1T6N959Q85RcNol_TuULew	0	2017-07-08 18:22:08	0	E1zzDf2Y86fRIV7Z7oxrQA	1	This AAA office is a pleasure to visit. The fr...	1	VL-KXKqEZ2682Qcs6B44Jg
27145	_gO1inMQhfXlK_S46SgGJA	1	2014-09-10 01:31:39	1	FYubvyEcoi-BudnWV4MasA	-1	My wife and I came here on our wedding anniver...	2	4SM_JIV-ZUk5Zl87dTKEtQ
6603	SoCVWvr5f5lYJEY_62bgAw	0	2016-04-11 20:22:27	2	iNsFMNikcmbdl7vuzKuMtg	-1	I came here at 9pm and the sign on the door sa...	0	9D0-Jj34bTK9P3v8zSFy9w
66696	5aeFlQjZlwJhR7gA-9VMWA	0	2017-04-09 01:36:07	0	AnZpvCM-423W3LIVnZijuA	-1	Do not go here! We recently went to get a car ...	0	LooAeVfGYajxhtYsfdP18A
62080	8HOZJRybEinI3krJK6NHbg	0	2011-09-27 13:28:16	2	bW0G_mh0OLauX-1jzVOw-Q	-1	I've been wanting LASIK for years and this pla...	40	G_Wv-inIZFSQyz98PGKfCw
14952	SoCVWvr5f5lYJEY_62bgAw	1	2016-04-12 00:18:54	1	wH4mXLLea2IfNttDDsDh5g	1	I had to reset my Yelp password so I could lea...	1	bj1a_mGtKwZGFwR3cmOMVg
6798	-hipP74BQsEFa2WnDvbi6w	0	2017-01-27 20:35:13	0	sXjEffi2NcnDy2bzP2sCxA	1	Dr Desanto is amazing! He's kind and patient. ...	1	q0Cti-VYvu8sDtGMzV2RpQ
1593	rcaPajgKOJC2vo_l3xa42A	0	2013-09-04 02:00:58	0	CkU7dYkQ2_PdMG6ztpjgxg	-1	My friends and I were very excited to have bru...	0	YPVEDPGqxIbcFElZd2iRzQ
44441	H6skMpg_g-sOrfTQf_Q5LQ	0	2015-11-19 18:20:01	0	QVkqNuBNiDCybrtX2aRp1w	-1	the food wasn't as bad as I expected it to be ...	1	FsqkZac7zHkMKwx2WHf28Q

	business_id	cool	date	funny	review_id	class	text	useful	user_id
19446	vnvQ0ID9MDje2DFde9PKQA	1	2012-06-27 22:41:37	1	NW7OlAQatRwB42y5bM67Xg	-1	We'll start with no smoking, no gambling, and ...	5	8QKrhvVqhEkD8xo4E4s0GQ
54258	eqQnIjAAhOUuHqK6gTWRXw	0	2017-03-18 01:50:00	0	HrROQDXgCE5TMObe8en9Tg	-1	Went for dinner today at 4pm the place was emp...	0	eK5ncNLYkwiFaQ4kxxwAvA

40000 rows × 9 columns

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

Run the below block multiple times to see different reviews and their respective classes. These reviews are two and four star reviews now, so there are less extreme. Please comment below on what interesting aspects of the reviews you find associated with each class and how these differ from the reviews before.

In [9]:

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

```
Text: If you're looking for diner food, that's diner quality, then this is the place. It's everything you would fi

The food is edible, but nothing beyond that. The food is diner food, and frankly, not even good diner food. I had

That being said, my dad LOVES this place and thinks it's the only place worth eating breakfast in town. Different

Class: -1
```

In [10]:

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

```
Text: Brand new eatery at Pape Ave & Aldrych. Amazing fresh food. Probably the best gyros I have ever experienced.

Class: 1
```

RESPONSE:¶

Yelp Reviews: The negative reviews seem to be of a similar manner but a lot milder than before. For example, reviews still contain words such as 'burnt' and 'OK', but also other words such as 'kind' and 'friendly'. There is more complexity in these reviews than in the other ones, and even reviews with some positive words can be negative.

On the other hand, the positive reviews seem to have less complexity than the negative ones. Many reviews contain primarily positive words like 'happy', 'amazing', and 'friendly' but are four stars purely because the reviewer did not deem it good enough for a five. The reviews themselves seem to not have too much ambiguity, as opposed to the negative ones.

Any meaningful answer that discusses more complexity and nuance in reviews as opposed to the polarity of the one and five star reviews should get full credit.

In [11]:

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

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

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

Part L: Baseline Model¶

In [162]:

```
print("Beginning Train Featurization")
currBaselineFeaturized_data = np.array(list(map(baseline_featurize, trainX)))
print("Beginning Training")
currBaselineModel = trainModel(currBaselineFeaturized_data, np.asarray(dfTrainset["class"]))
print("Beginning Test Featurization")
testFeaturizedBaseline_data = np.array(list(map(baseline_featurize, testX)))
print("Accuracy:")
accuracyData(currBaselineModel, testFeaturizedBaseline_data, np.asarray(dfTestset["class"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[1713 2278]
 [1106 2903]]
0.577
```

Out[162]:

```
0.577
```

What was your accuracy percentage? Was it what you expected?¶

RESPONSE:¶

Yelp Reviews: The accuracy for the baseline model was 0.577, similar to before, meaning that length of message is better than randomly guessing but not enough to do too much better.

Part M: Bag of Words Model¶

In [163]:

```
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"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[3676  315]
 [ 299 3710]]
0.92325
```

Out[163]:

```
0.92325
```

What was your accuracy percentage? Was it what you expected?¶

RESPONSE:¶

Yelp Reviews: The accuracy of the initial bag of words model is 0.923 which is significantly lower than the acccuracy when run on the dataset with just 1's and 5's. This makes sense as there is more nuance now so the model will have more difficulty classifying.

Part N: Binary Bag of Words Model¶

In [164]:

```
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"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[3673  318]
 [ 295 3714]]
0.923375
```

Out[164]:

```
0.923375
```

What was your accuracy percentage? Was it what you expected? How did it compare to the regular Bag of Words model?¶

RESPONSE:¶

Yelp Reviews: The accuracy with binary bag of words is slightly higher at 0.9234 vs 0.933 from before, so the model did improve with binary bag of words as expected but not by too much. This is because with longer reviews binary bag of words will weight a certain few stronger words much higher based on their presence.

Part O: Negative Bag of Words Model¶

In [17]:

```
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, testX)))
print("Accuracy:")
accuracyData(negModel, testFeaturizedNeg_data, np.asarray(dfTestset["class"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[3690  319]
 [ 288 3703]]
0.924125
```

Out[17]:

```
0.924125
```

What was your accuracy percentage? Was it what you expected? How did it compare to the regular Bag of Words model?¶

RESPONSE:¶

Yelp Reviews: This accuracy of 0.924 is higher than both the normal Bag of Words and Binary Bag of Words that were just run. This is because it enables us to account for negatives like "not good" but the difference is still not a very large margin.

Part P: Negative Binary Bag of Words Model¶

In [18]:

```
print("Beginning Train Featurization")
negbin_data = np.array(list(map(bag_of_words_neg_binary_featurize, trainX)))
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_featurize, testX)))
print("Accuracy:")
accuracyData(negBinModel, testFeaturizedNegBin_data, np.asarray(dfTestset["class"]))
```

```
Beginning Train Featurization
Beginning Training
Beginning Test Featurization
Accuracy:
[[3798  211]
 [ 176 3815]]
0.951625
```

Out[18]:

```
0.951625
```

What was your accuracy percentage? Was it what you expected? How did it compare to the regular Bag of Words model?¶

RESPONSE:¶

Yelp Reviews: The accuracy of 0.952 is significantly higher than any of the models so far. The combination of both the corrections for binary bag of words and allowing for negation did much better together than either of them individually. Ultimately, it can be seen that when the dataset gets more convoluted and the binary classification is not clearly divided the original Bag of Words featurization may not do as well as expected. Instead, modifications like the ones we made for binary Bag of Words and providing additional features for negation have the potential to result in large improvements in model performance and thus should be determined in an experimen. Additional summary level features can and should be added in order to improve the model further, such as length, punctuation, other dictionaries, and more.