Assignment 2: Sentiment Classification Mini Project

Release date: 21th of Jan, 2018
Due date: 28th of Jan, 2018

Introduction

Nowadays people express their opinions all over the internet. They write product reviews, movie reviews, restaurant reviews, political opinions and so on. We use these reviews to make decisions, whether to watch this a movie, or eat in a restaurant, or buy a new phone. While these reviews are really helpful to make decision, they can be a curse sometimes. Imagine you want to buy a new phone and found 100,000 reviews about that phone, can you read them all? OF COURSE NOT! it's a huge effort and some would say a waste of time. It would be very helpful if a program can read these reviews and decide the sentiment for each review automatically.

The sentiment of a review is whether this review express a positive opinion (a like) or a negative opinion (dislike). For example, the following review is a positive one.

- This restaurant is awesome!

The following review is a negative one

- The battery life is too short, don't buy this phone.

In this project your will write a python program that will read a bunch of reviews and for each one of these reviews it will decide whether that review is positive or a negative one.

How can a program know the sentiment automatically?

You probably thought of this by now, deciding the sentiment is not a straightforward task, how can a machine (a computer) decide the sentiment. The answer is Machine Learning.

The idea is to show the computer couple reviews that we know beforehand their true sentiment, so that the computer can learn the model of a positive review and that of a negative review. We call this phase a ***training phase***, which is exactly as it sounds, the program is training on how to differentiate between positive and negative reviews. The output of the training phase is a model for positive reviews and another for negative reviews.

Now that program is trained, we give the program a new review it has not seen before in the training phase. The program uses its models learnt in the training phase to decide the sentiment of the new review. We call this phase ***classification phase***, because we are using the learnt models to decide (classify) the sentiment of new reviews.

If we have couple of new reviews that we know their true sentiment but we did not include them in the training phase, then these reviews are suitable for classification phase, and since we know their true classification (positive of negative) we can compare their true sentiment with predications our program made in order to see if the program did a good job or not. This phase called ***measurement phase***, because we measure how good are the predictions of our program.

In this project you will be guided to write functions to implement these phases: training, classification and measurement.

Data Description

Data is collected from Amazon, imdb and Yelp. You are given two files with the same format, 'TRAINING.txt' contains reviews with their true sentiment. These reviews will be used in the training phase.

The second file is 'TESTING.txt'. contains reviews with their true sentiment. These reviews will be used in the classification phase and we'll use the true sentiment in the measurement phase to see how accurate is the program.

Files format:

Each line contains a review and its true sentiment separated by a tab. Sentiments can take two values, either 1 for positive sentiment, or 0 for negative sentiment.

Examples:

```
The bose noise cancelling is amazing, which is very important for a NYC commuter.	1
The keyboard is a nice compromise between a full QWERTY and the basic cell phone number keypad.	1
Very disappointing.	0
This item worked great, but it broke after 6 months of use.	0
#1 It Works - #2 It is Comfortable.	1
SWEETEST PHONE!!!	1
```

Task 1: Reading Data

The first task is to read the two files, "TRAINING.txt" and "TESTING.txt" and divide the content of each file into two lists; one for positive reviews and the second for negative reviews and return these two lists.

You are required to implement the function:

def **readAndDivideBySentiment**(fileName):

Inputs:
1) fileName: the name of the data file (e.g. "TRAINING.txt"). Type: string

Returns:
1) pos: A list of positive reviews. Type: A list of strings
2) neg: A list of negative reviews. Type: A list of strings

Description:
This function takes a name of the data file (training or testing) for reading. Then it divides the reviews in this file into two lists, a list containing all positive reviews and a list containing all negative reviews. Since this function take the name of ONE file only, this means you have to call this function twice in the main, once for the training file and another for the testing file.

You should note that since we are forming two lists one for positive reviews and another for negative reviews, the 0 and 1 in the files that identify a review is positive or negative should not be including in the returned lists.

Example:

If the input is as follows:

TRAINING.txt
Great price also! 1
This phone is slim and light and the display is beautiful.   1
The bose noise cancelling is amazing, which is very important for a NYC commuter.   1
The keyboard is a nice compromise between a full QWERTY and the basic cell phone number keypad.   1
Very disappointing.   0
This item worked great, but it broke after 6 months of use.   0

Your function should return two lists:

The positive list:
['Great price also!', 'This phone is slim and light and the display is beautiful.', 'The bose noise cancelling is amazing, which is very important for a NYC commuter.', 'The keyboard is a nice compromise between a full QWERTY and the basic cell phone number keypad.']

The negative list:
['Very disappointing.', 'This item worked great, but it broke after 6 months of use.']

Task 1A: Implement this function
Task 1B: Call the function from the main twice, once for the training file and another for the testing file.

Task 2: Data Cleaning

If you take a look at the reviews you will notice that people write in lower case, upper case, use exclamation marks, smiley faces … etc. These different types of writing can confuse your program, so in this task you are required to clean these inconsistencies from the reviews.

Cleaning instructions:

-Numbers as prefix or suffix must be separated from the word:
xyz12 → xyz 12
12xyz → 12 xyz

-Convert all numbers to token 'num':
xyz 12 → xyz num

-Numbers inside words should be left alone:
seq2seq → seq2seq

-Only single underscores separating words should be removed:
power-wise → powerwise
well--it's looking good → well it's looking good

-Continuous numbers should be just one 'num' token:
1 2 35 1.3 → NUM

-all special non-alpha numeric characters should be removed (except for '):
it's good! → it's good

You are required to implement the function:

def **cleanData**(myData):

Inputs:
1) myData: A list of reviews. Type: A list of strings

Returns:
1) cleanedData: A list of cleaned reviews. Type: A list of strings

Description:
This function takes a list of reviews and clean according to the previous instructions

Since this function cleans a list of reviews, we can to call this function four times in the main for the positive training reviews, negative training reviews, positive testing reviews and negative testing reviews.

Examples:
If the input list is:
['Great price also!', 'This phone is slim and light and the display is beautiful.', 'The bose noise cancelling is amazing, which is very important for a NYC commuter.', 'The keyboard is a nice compromise between a full QWERTY and the basic cell phone number keypad.']

The output cleaned list is:
['great price also', 'this phone is slim and light and the display is beautiful', 'the bose noise cancelling is amazing which is very important for a nyc commuter', 'the keyboard is a nice compromise between a full qwerty and the basic cell phone number keypad']

Task 2A: Implement this function
Task 2B: Call the function from the main four times, two times for the training reviews (positive & negative) and another for the test reviews (positive and negative).

Training Phase

In the training phase, you will be calculating some statistics for both the positive and negative reviews of the training data. These statistics are the model learnt by the program which will use later on to classify reviews from the testing reviews. I.e. The statistics calculated for the positive training reviews is the model we learnt for the positive reviews, and the statistics for the negative training reviews is the model we learnt for the negative reviews

Task 3: Calculate Unique Words Frequency

For the cleaned positive training reviews and the cleaned negative training reviews, it is required to identify all the unique words and they count their frequency.

You are required to implement the function:

def **calculateUniqueWordsFreq**(trainData, cutOff):

Inputs:
1) trainData: A list of cleaned training reviews (positive or negative). Type: A list of strings

2) cutOff: an integer specifying the minimum frequency for a word. All words with frequency less that cutOff should not be added in the returned dictionary

Returns:
1) vocab: A dictionary, words are keys and their frequency are values.
Type: A dictionary<key:word(type:string), value:frequency(type:int)>

Description:
This function takes a list of cleaned reviews and construct a dictionary whose keys are the unique words in all reviews of the input list, and values are the frequency(count) of each word.
You need to call this function twice from the main. Once for the cleaned training positive reviews, and another for the cleaned training negative reviews.

Note: A space (“ “) and an empty string (“”) are not considered words, so you should not include them in the output dictionary.

For example:

If the input list was:
['great price also', 'this phone is slim and light and the display is beautiful', 'the bose noise cancelling is amazing which is very important for a nyc commuter', 'the keyboard is a nice compromise between a full qwerty and the basic cell phone number keypad']
The output dictionary should be:

{'display': 1, 'this': 1, 'qwerty': 1, 'also': 1, 'keyboard': 1, 'between': 1, 'a': 3, 'full': 1, 'keypad': 1, 'very': 1, 'compromise': 1, 'number': 1, 'phone': 2, 'price': 1, 'light': 1, 'amazing': 1, 'for': 1, 'slim': 1, 'important': 1, 'and': 3, 'basic': 1, 'great': 1, 'commuter': 1, 'cell': 1, 'which': 1, 'is': 5, 'cancelling': 1, 'beautiful': 1, 'bose': 1, 'the': 4, 'nice': 1, 'nyc': 1, 'noise': 1}

Task 3A: Implement this function
Task 3B: Call the function from the main two times, once for the positive cleaned training reviews and another for the negative cleaned training reviews.

Task 4: Calculate Class[1] Probability

The second (and final) statistic we calculate is the probability of a positive review and the probability of a negative review. It is simply calculated as follows:

$$Probability\ of\ a\ positive\ review = \frac{The\ number\ of\ positive\ reviews\ in\ the\ training\ set}{The\ number\ of\ all\ reviews\ in\ the\ training\ set}$$

$$Probability\ of\ a\ negative\ review = \frac{The\ number\ of\ negative\ reviews\ in\ the\ training\ set}{The\ number\ of\ all\ reviews\ in\ the\ training\ set}$$

You are required to implement this function:

def **calculateClassProbability**(posTrain, negTrain):

[1] The word *class* refer to the classification of a review between positive and negative. I.e. when we say a review is positive, it means the class of this review is positive.

Inputs:
1) posTrain: A list of cleaned positive training reviews. Type: A list of strings
2) negTrain: A list of cleaned negative training reviews. Type: A list of strings

Returns:
1) posProb: how probable a positive review is. Type: float
2) negProb: how probable a negative review is. Type: float

Description:
This function takes both the cleaned positive and negative reviews of the training set and return the probability of a positive review and that of a negative review.

Task 4A: Implement this function
Task 4B: Call the function from the main two times, once for the positive cleaned training reviews and another for the negative cleaned training reviews.

Classification Phase

Now that our models for positive reviews and negative reviews are all trained. We can go ahead and start to use these models to classify the testing reviews between positive and negative.

First we will be using each of our trained models (positive and negative training data) to calculate scores both the testing data (both positive and negative).
Remember that technically we don't know the true class (positive or negative) of the reviews in the testing data, we only added their true classification so that we can tell if our models did a good job or not.

For example, if we have the following review from the testing data and we know it is a positive one.

A Test review: "I wear it everyday and it holds up very well."

We will use our trained model from the training phase both the positive and negative models and calculate the scores they give to this test review.
Let's assume the positive training model gave this test review a score of 0.85 and the negative training model gave it 0.55. Since the positive score is higher than the negative score we will classify this review as a positive one, and since we already know it should be a positive one then our model is correct.

However, if the negative training model gave it a score higher than that of the positive training model we will classify this review as a negative one, and since we already know it should be a positive one then our model is wrong.

This phase is concerned with the calculation of scores, while the next phase (measurement) will be concerned with counting how many correct and wrong classification.

Task 5: Calculate Scores

In this task we will calculate scores for positive and negative testing reviews using the training models positive and negative. This means you need to call this function four times in the main:

The first time to calculate the *positive* test data scores using *positive* training data model
The second time to calculate the *positive* test data scores using *negative* training data model
The third time to calculate the *negative* test data scores using *positive* training data model

The four time to calculate the *negative* test data scores using *negative* training data model

The positive training model is:

1. The dictionary of unique words and their frequency calculated in task 3 using the positive cleaned training set
2. The probability of a positive review calculated in task 4.

Similarly, the negative training model is:

1. The dictionary of unique words and their frequency calculated in task 3 using the negative cleaned training set
2. The probability of a negative review calculated in task 4.

Now, how to calculate the score for a single test review given a training model. Assume we have a test review r and a model m (this model can be a positive training model or a negative training model).

Assume the test review r has n words: $w_0, w_1, w_2, w_3, w_4 \ldots\ w_{n-1}$

The score of review r using model m =

$$Probability(m)\ \times score(w_0)\times\ score(w_1)\times score(w_2)\ \ldots\ score(w_{n-1})$$

- When m is the positive training, then $Probability(m)$ is the probability of a positive review calculated in task 4.
  When m is the negative training, then $Probability(m)$ is the probability of a negative review calculated in task 4.

-

$$score(w_i) = \frac{(Frequency\ of\ w_i\ in\ dictionary\ of\ model\ m) + 1}{(The\ summation\ of\ the\ frequency\ of\ all\ words\ in\ dictionary\ of\ model\ m) + (The\ number\ of\ words\ in\ the\ dictionary)}$$

Example:

Let's assume we have the test review: "oooh i love it".

Let the model be the positive training model:

1. The dictionary (calculated in task 3)

Word	Frequency
i	3
he	1
awesome	5
love	2
like	4

2. The probability of a positive review calculated in task 4 is 0.6

Now let's calculate the score of the test review:

$$score(oooh) = \frac{(0) + 1}{(3 + 1 + 5 + 2 + 4) + (5)} = \ 0.05$$

Where 0 is the frequency of the word “oooh” in the dictionary, and (3+1+5+2+4) is the summation of the frequencies of all the words in the dictionary of the model and (5) is the number of words in the dictionary.

$$score(i) = \frac{(3) + 1}{(3 + 1 + 5 + 2 + 4) + (5)} = 0.2$$
$$score(love) = \frac{(2) + 1}{(3 + 1 + 5 + 2 + 4) + (5)} = 0.15$$
$$score(it) = \frac{(0) + 1}{(3 + 1 + 5 + 2 + 4) + (5)} = 0.05$$

The score of the test review = (0.6) × (0.05 × 0.2 × 0.15 × 0.05) = 0.000045
You are required to implement this function:

def **calculateScores**(classProb, uniqueVocab, testData):

Inputs:
1) classProb: the probability of the class (positive or negative). Type: float
2) uniqueVocab: A dictionary, words are keys and their frequency are values.
Type: A dictionary<key:word(type:string), value:frequency(type:int)>
3) testData: A list of cleaned test reviews (positive or negative).

Returns:
1) testScores: A list of scores given to the input testData according to the input training data. Type: A list of floats

Description:
This function calculates a score for each review in the input testData using the input training model: trainData, classProb and uniqueVocab. Then returns a list of the calculated scores.

Note: the review at index i in testData will have the score at index i in testScores

Task 5A: Implement this function.
Task 5B: Call the function from the main.

Measurement Phase

In this phase, we are concerned with measuring and quantifying how good our models are.

Task 6: Calculate Accuracy

This function takes the scores calculated in task 5 and for each test review it is required to compare its score using the positive training model with its score using the negative training model and classify the test review to the greater score. I.e. if the positive score was greater than this test review is positive otherwise it is negative.

Now that we know the classification of the test review according to our models, we also know the true class of this review (we know whether it should be positive or negative). If the classification using our models matches the true class then we successfully classified this test review, otherwise we misclassified this test review.

In this function we want to count how many correctly classified positive and negative test reviews, and how many misclassified positive and negative test reviews.
you are required to count the following:

- **tp**: It’s short for **true positive**. It’s the number of test reviews that we know they should be *positive* and we classified them as *positive* with our model.

- **fp**: It’s short for **false positive**. It’s the number of test reviews that we know they should be *negative* and we classified them as *positive* with our model.

- **tn**: It’s short for **true negative**. It’s the number of test reviews that we know they should be *negative* and we classified them as *negative* with our model.

- **fn**: It’s short for **false negative**. It’s the number of test reviews that we know they should be *positive* and we classified them as *negative* with our model.

You are required to implement the function:

def **calculateAccuracy**(positiveTestDataPostiveScores, positiveTestDataNegativeScores, negativeTestDataPostiveScores, negativeTestDataNegativeScores)

Inputs:
1) positiveTestDataPostiveScores:
A list of scores for *positive* test data calculated from POSITIVE training data model.
Type: A list of floats
2) positiveTestDataNegativeScores:
A list of scores for *positive* test data calculated from NEGATIVE training data model
Type: A list of floats
3) negativeTestDataPostiveScores:
A list of scores for *negative* test data calculated from POSITIVE training data model
Type: A list of floats
4) negativeTestDataNegativeScores:
A list of scores for *negative* test data calculated from NEGATIVE training data model
Type: A list of floats

Returns:
1) tp: The number of true positives
2) fp: The number of false positives.
3) tn: The number of true negatives.
4) fn: The number of false negatives.

Description:
This function calculates how accurate the predictions we made for the test set. Since we know the correct classification for each review whether it's positive or negative, we compare the prediction we made to the truth and count the number of reviews whose predictions matched the truth and those didn't match. Using these counts we can calculate how accurate our predictions really are.

Example:

Let’s assume we only have 5 test reviews. The first 3 reviews are positive and last 2 are negative.

The input to the function **calculateAccuracy:**

- positiveTestDataPostiveScores: [0.03, 0.0021, 0.15]
- positiveTestDataNegativeScores: [0.001, 0.13, 0.15]
- negativeTestDataPostiveScores: [0.01, 0.00004]
- negativeTestDataNegativeScores: [0.002, 0.08]

This means the first review (which is a positive review) was given 0.03 as a score from the positive model, and 0.001 from the negative model. And the fourth review (which is a negative review) was given 0.01 as a score from the positive model, and 0.002 from the negative model.

First we will classify the positive test reviews (those we know they should be positive). Consider the first and the second list "positiveTestDataPostiveScores", "positiveTestDataNegativeScores" we will loop through them and for each review we will compare its two scores and classify it according to the greater score.

For review#0: the positive score 0.03 is > the negative score 0.001. So, classify it as positive and since this review is in fact a positive one, then we increment the true positive count. tp = 1

For review#1: the positive score 0.0021 is < the negative score 0.13. So, classify it as negative and since this review is in fact a positive one, then we increment the false negative count. fn = 1

For review#2: the positive score 0.15 is = the negative score 0.15. So, classify it as positive and since this review is in fact a positive one, then we increment the true positive count. tp = 2

Now, we will classify the negative test reviews (those we know they should be negative). Consider the third and the fourth list "negativeTestDataPostiveScores", "negativeTestDataNegativeScores" we will loop through them and for each review we will compare its two scores and classify it according to the greater score.

For review#3: the positive score 0.01 is > the negative score 0.002. So, classify it as positive and since this review is in fact a negative one, then we increment the false positive count. fp = 1

For review#4: the positive score 0.00004 is < the negative score 0.08. So, classify it as negative and since this review is in fact a negative one, then we increment the true negative count. tn = 1

Note:
For tp, tn: the higher these counts, the more accurate our models
For fp, fn: the lower these counts, the more accurate our models

Task 6A: Implement this function.
Task 6B: Call the function from the main, and print the returned counts.

Turning in Your Work

- Submit your completed file (HW2.py) on Blackboard.

Rubric

HW 2	
Task 1: Reading Data	**5%**
Task 2: Data Cleaning	**30%**
Task 3: Calculate Unique Words Frequency	**20%**
Task 4: Calculate Class Probability	**5%**
Task 5: Calculate Scores	**25%**
Task 6: Calculate Accuracy	**10%**
Main	**5%**
TOTAL	**100%**