Report

Part 1: Classification and Evaluation

ALPHA Values	Test Accuracy
0.1	0.81132
0.5	0.81148
1.0	0.82284
5.0	0.82479
10	0.82722

Part 2: Probability Prediction:

Record the probability estimated for the first 10 reviews in the test data for ALPHA 1.0.

Review 1:

-1.0 -1.0 0.003682506321622078 0.9963174936784206 I went and saw this movie last night after being coaxed to by a few friends of mine. I'll admit that I was reluctant to see it because from what I knew of Ashton Kutcher he was only able to do comedy. I was wrong. Kutcher played the character of Jake Fischer very well, and Kevin Costner played Ben Randall with such professionalism. The sign of a good movie is that it can toy with our emotions. This one did exactly that. The entire theater (which was sold out) was overcome by laughter during the first half of the movie, and were moved to tears during the second half. While exiting the theater I not only saw many women in tears, but many full grown men as well, trying desperately not to let anyone see them crying. This movie was great, and I suggest that you go see it before you judge.

Review 2:

-1.0 -1.0 3.9776289702989794e-08 0.9999999602237047 Actor turned director Bill Paxton follows up his promising debut, the Gothic-horror "Frailty", with this family friendly sports drama about the 1913 U.S. Open where a young American caddy rises from his humble background to play against his Bristish idol in what was dubbed as "The Greatest Game Ever Played." I'm no fan of golf, and these scrappy underdog sports flicks are a dime a dozen (most recently done to grand effect with "Miracle" and "Cinderella Man"), but some how this film was enthralling all the same.

The film starts with some creative opening credits (imagine a Disneyfied version of the animated opening credits of HBO's "Carnivale" and "Rome"), but lumbers along slowly for its first by-the-numbers hour. Once the action moves to the U.S. Open things pick up very well. Paxton does a nice job and shows a knack for effective directorial flourishes (I loved the rain-soaked montage of the action on day two of the open) that propel the plot further or add some unexpected psychological depth to the proceedings. There's some compelling character development when the British Harry Vardon is haunted by images of the aristocrats in black suits and top hats who destroyed his family cottage as a child to make way for a golf course. He also does a good job of visually depicting what goes on in the players' heads under pressure. Golf, a painfully boring sport, is brought vividly alive here. Credit should also be given the set designers and costume department for creating an engaging period-piece atmosphere of London and Boston at the beginning of the twentieth century.

You know how this is going to end not only because it's based on a true story but also because films in this genre follow the same template over and over, but Paxton puts on a better than average show and perhaps indicates

more talent behind the camera than he ever had in front of it. Despite the formulaic nature, this is a nice and easy film to root for that deserves to find an audience.

Review 3:

-1.0 1.0 0.9791576631454483 0.02084233685465691 As a recreational golfer with some knowledge of the sport's history, I was pleased with Disney's sensitivity to the issues of class in golf in the early twentieth century. The movie depicted well the psychological battles that Harry Vardon fought within himself, from his childhood trauma of being evicted to his own inability to break that glass ceiling that prevents him from being accepted as an equal in English golf society. Likewise, the young Ouimet goes through his own class struggles, being a mere caddie in the eyes of the upper crust Americans who scoff at his attempts to rise above his standing.

What I loved best, however, is how this theme of class is manifested in the characters of Ouimet's parents. His father is a working-class drone who sees the value of hard work but is intimidated by the upper class; his mother, however, recognizes her son's talent and desire and encourages him to pursue his dream of competing against those who think he is inferior.

Finally, the golf scenes are well photographed. Although the course used in the movie was not the actual site of the historical tournament, the little liberties taken by Disney do not detract from the beauty of the film. There's one little Disney moment at the pool table; otherwise, the viewer does not really think Disney. The ending, as in "Miracle," is not some Disney creation, but one that only human history could have written.

Review 4:

-1.0 -1.0 0.033216000968771 0.966783999031316 I saw this film in a sneak preview, and it is delightful. The cinematography is unusually creative, the acting is good, and the story is fabulous. If this movie does not do well, it won't be because it doesn't deserve to. Before this film, I didn't realize how charming Shia Lebouf could be. He does a marvelous, self-contained, job as the lead. There's something incredibly sweet about him, and it makes the movie even better. The other actors do a good job as well, and the film contains moments of really high suspense, more than one might expect from a movie about golf. Sports movies are a dime a dozen, but this one stands out.

This is one I'd recommend to anyone.

Review 5:

-1.0 -1.0 0.0024519124659441002 0.9975480875340544 Bill Paxton has taken the true story of the 1913 US golf open and made a film that is about much more than an extra-ordinary game of golf. The film also deals directly with the class tensions of the early twentieth century and touches upon the profound anti-Catholic prejudices of both the British and American establishments. But at heart the film is about that perennial favourite of triumph against the odds.

The acting is exemplary throughout. Stephen Dillane is excellent as usual, but the revelation of the movie is Shia LaBoeuf who delivers a disciplined, dignified and highly sympathetic performance as a working class Franco-Irish kid fighting his way through the prejudices of the New England WASP establishment. For those who are only familiar with his slap-stick performances in "Even Stevens" this demonstration of his maturity is a delightful surprise. And Josh Flitter as the ten year old caddy threatens to steal every scene in which he appears.

A old fashioned movie in the best sense of the word: fine acting, clear directing and a great story that grips to the end - the final scene an affectionate nod to Casablanca is just one of the many pleasures that fill a great movie.

Review 6:

-1.0 -1.0 6.10737157681176e-19 1.0 I saw this film on September 1st, 2005 in Indianapolis. I am one of the judges for the Heartland Film Festival that screens films for their Truly Moving Picture Award. A Truly

Moving Picture "...explores the human journey by artistically expressing hope and respect for the positive values of life." Heartland gave that award to this film.

This is a story of golf in the early part of the 20th century. At that time, it was the game of upper class and rich "gentlemen", and working people could only participate by being caddies at country clubs. With this backdrop, this based-on-a-true-story unfolds with a young, working class boy who takes on the golf establishment and the greatest golfer in the world, Harry Vardon.

And the story is inspirational. Against all odds, Francis Ouimet (played by Shia LaBeouf of "Holes") gets to compete against the greatest golfers of the U.S. and Great Britain at the 1913 U.S. Open. Francis is ill-prepared, and has a child for a caddy. (The caddy is hilarious and motivational and steals every scene he appears in.) But despite these handicaps, Francis displays courage, spirit, heroism, and humility at this world class event.

And, we learn a lot about the early years of golf; for example, the use of small wooden clubs, the layout of the short holes, the manual scoreboard, the golfers swinging with pipes in their mouths, the terrible conditions of the greens and fairways, and the play not being canceled even in torrential rain.

This film has stunning cinematography and art direction and editing. And with no big movie stars, the story is somehow more believable.

This adds to the inventory of great sports movies in the vein of "Miracle" and "Remember the Titans."

FYI - There is a Truly Moving Pictures web site where there is a listing of past winners going back 70 years.

Review 7:

-1.0 -1.0 1.4263748309625483e-08 0.9999999857363946 Maybe I'm reading into this too much, but I wonder how much of a hand Hongsheng had in developing the film. I mean, when a story is told casting the main character as himself, I would think he would be a heavy hand in writing, documenting, etc. and that would make it a little biased.

But...his family and friends also may have had a hand in getting the actual details about Hongsheng's life. I think the best view would have been told from Hongsheng's family and friends' perspectives. They saw his transformation and weren't so messed up on drugs that they remember everything.

As for Hongsheng being full of himself, the consistencies of the Jesus Christ pose make him appear as a martyr who sacrificed his life (metaphorically, of course, he's obviously still alive as he was cast as himself) for his family's happiness. Huh?

The viewer sees him at his lowest points while still maintaining a superiority complex. He lies on the grass coming down from (during?) a high by himself and with his father, he contemplates life and has visions of dragons at his window, he celebrates his freedom on a bicycle all while outstretching his arms, his head cocked to the side.

It's fabulous that he's off of drugs now, but he's no hero. He went from a high point in his career in acting to his most vulnerable point while on drugs to come back somewhere in the middle.

This same device is used in Ted Demme's "Blow" where the audience empathizes with the main character who is shown as a flawed hero.

However, "Quitting" ("Zuotian") is a film that is recommended, mostly for its haunting soundtrack, superb acting, and landscapes. But, the best part is the feeling that one gets when what we presume to be the house of Jia Hongsheng is actually a stage setting for a play. It makes the viewer feel as if Hongsheng's life was merely a play told in many difficult parts.

Review 8:

1.0 1.0 1.0 3.0207862270305336e-16 I felt this film did have many good qualities. The cinematography was certainly different exposing the stage aspect of the set and story. The original characters as actors was certainly an achievement and I felt most played quite convincingly, of course they are playing themselves, but definitely unique. The cultural aspects may leave many disappointed as a familiarity with the Chinese and Oriental culture will answer a lot of questions regarding parent/child relationships and the stigma that goes with any drug use. I found the Jia Hongsheng story interesting. On a down note, the story is in Beijing and some of the fashion and music reek of early 90s even though this was made in 2001, so it's really cheesy

sometimes (the Beatles crap, etc). Whatever, not a top ten or twenty but if it's on the television, check it out.

Review 9:

-1.0 -1.0 5.434640323067272e-08 0.9999999456536003 This movie is amazing because the fact that the real people portray themselves and their real life experience and do such a good job it's like they're almost living the past over again. Jia Hongsheng plays himself an actor who quit everything except music and drugs struggling with depression and searching for the meaning of life while being angry at everyone especially the people who care for him most. There's moments in the movie that will make you wanna cry because the family especially the father did such a good job. However, this movie is not for everyone. Many people who suffer from depression will understand Hongsheng's problem and why he does the things he does for example keep himself shut in a dark room or go for walks or bike rides by himself. Others might see the movie as boring because it's just so real that its almost like a documentary. Overall this movie is great and Hongsheng deserved an Oscar for this movie so did his Dad.

Review 10:

-1.0 -1.0 0.3812825413141651 0.6187174586858595 "Quitting" may be as much about exiting a pre-ordained identity as about drug withdrawal. As a rural guy coming to Beijing, class and success must have struck this young artist face on as an appeal to separate from his roots and far surpass his peasant parents' acting success. Troubles arise, however, when the new man is too new, when it demands too big a departure from family, history, nature, and personal identity. The ensuing splits, and confusion between the imaginary and the real and the dissonance between the ordinary and the heroic are the stuff of a gut check on the one hand or a complete escape from self on the other. Hongshen slips into the latter and his long and lonely road back to self can be grim.

But what an exceptionally convincing particularity, honesty, and sensuousness director Zhang Yang, and his actors, bring to this journey. No clichés, no stereotypes, no rigid gender roles, no requisite sex, romance or violence scenes, no requisite street language and, to boot, no assumed money to float character acts and whims.

Hongshen Jia is in his mid-twenties. He's a talented actor, impressionable, vain, idealistic, and perhaps emotionally starved. The perfect recipe for his enablers. Soon he's the "cool" actor, idolized by youth. "He was hot in the early nineties." "He always had to be the most fashionable." He needs extremes, and goes in for heavy metal, adopts earrings and a scarf. His acting means the arts, friends--and roles, But not the kind that offer any personal challenge or input. And his self-criticism, dulled by the immediacy of success, opens the doors to an irrational self-doubt, self-hatred-- "I didn't know how to act" "I felt like a phony"--and to readily available drugs to counter them. He says "I had to get high to do what director wanted." So, his shallow identity as an actor becomes, via drugs, an escape from identity.

Hongshen's disengagement from drugs and his false life is very gradual, intermittent--and doggedly his own. Solitude, space, meditative thinking, speech refusal, replace therapy. The abstract is out. And a great deal of his change occurs outdoors---not in idealized locations but mainly on green patches under the freeways, bridges, and high-rises of Beijing. The physicality is almost romantic, but is not. The bike rides to Ritan Park, the long spontaneous walks, the drenching sun and rain, grassy picnics, the sky patterns and kites that absorb his musing are very specific. He drifts in order to arrive, all the while picking up cues to a more real and realistic identity. "I started to open up" he says of this period in retrospect. And the contact seems to start with his lanky body which projects a kind of dancer's positioning (clumsy, graceful, humorous, telling) in a current circumstance. If mind or spirit is lacking, his legs can compel him to walk all night.

Central to his comeback is the rejection of set roles. To punctuate his end to acting and his determination to a new identity, he smashes his videos and TV, and bangs his head till bloody against his "John Lennon Forever" poster. He has let down his iconic anti-establishment artist---but he's the only viable guide he knows. He even imagines himself as John's son

(Yoko Ono), and adopts his "Mother Mary" as an intercessor in his "hour of darkness" and "time of trouble." (the wrenching, shaking pain in the park--hallucinatory and skitzoid ordeals) "Music is so much more real than acting" he says. And speaks of Lennon's influence as "showing me a new way." In the mental institute, the life-saving apples (resistance, nourishment) reflect Lennon's presence, as does Hongshen's need to re-hang his hero's poster in his redecorated room.

If Lennon's influence is spiriting, Hongshen's father's influence is grounding. Although father and son are both actors and users (drugs and drink), it is Fegsen's differences from his son that underwrites his change. For the father is more secure in himself: he accepts that he's Chinese, a peasant in a line of peasants, a rural theater director. And he exercises control over both his habit and his emotions. It's this recognizable identity that drives Hongshen to treat him like a sounding board, sometimes with anger and rage, sometimes with humor (the blue jeans, Beatles) and passivity. In his most crazed, and violent exchange with his father in which he accuses him of being a liar, and a fake, he exposes more of himself than his father: "all the acts I acted before were bullshit... life is bullshit." And to Hongshen's emphatic "you are NOT my father," he softly replies, "why can't a peasant be your father?"

Under these two teachers and with much additional help from his mother, sister, friends, inmates at the rehab inst., he makes some tangible connection to a real (not whole) self. As the long term drug effects recede, so does his old identity. Indebtedness replaces pride, trust distrust. Integrity banishes his black cloud. All his edges soften. "You are just a human being" he repeats endlessly after being released from the strap-down incurred for refusing medicine. Back home, lard peasant soap is fine with him now. And his once "rare and true friendships" begin again as is so evident in the back to poignant back-to-back fence scene with his musician buddy. Hongshen says of this movie: "it's a good chance to think about my life." And I might add, become a New Actor, one bound to art and life. Like Lennon, he has gained success without a loss of identity.

Code Snippet of Naïve Bayes Train Data:

```
def Train(self, X, Y):
    #TODO: Estimate Naive Bayes model parameters
    positive_indices = np.argwhere(Y == 1.0).flatten()
    negative_indices = np.argwhere(Y == -1.0).flatten()
    print(positive_indices)

    self.num_positive_reviews = len(positive_indices)
    self.num_negative_reviews = len(negative_indices)

    self.count_positive = csr_matrix.sum(X[np.ix_(positive_indices)], axis=0) #np.ones([1,X.shape[1]])
    self.count_negative = csr_matrix.sum(X[np.ix_(negative_indices)], axis=0)#np.ones([1,X.shape[1]])

    self.total_positive_words = csr_matrix.sum(X[np.ix_(positive_indices)])
    self.total_negative_words = csr_matrix.sum(X[np.ix_(negative_indices)])

    self.deno_pos = float(self.total_positive_words + self.ALPHA * X.shape[1])
    self.deno_neg = float(self.total_negative_words + self.ALPHA * X.shape[1])

    self.count_positive = (self.count_positive + self.ALPHA)
    self.count_negative = (self.count_negative + self.ALPHA)

    return
```

Code Snippet of Naïve Bayes Predicting Label:

```
def PredictLabel(self, X, probThresh, flag):
    #TODO: Implement Naive Bayes Classification
    self.P_positive = log(float(self.num_positive_reviews)) - log(float(self.num_positive_reviews + self.num_negative_reviews))
    self.P_negative = log(float(self.num_negative_reviews)) - log(float(self.num_positive_reviews + self.num_negative_reviews))
    pred_labels = []
    #print(self.P_positive)
    sh = X.shape[0]

    for i in range(sh):
        z = X[i].nonzero()
        possum = self.P_positive
        negsum = self.P_negative
        for j in range(len(z[0])):
            # Look at each feature
            r = i  #row index
            c =z[1][j]   #column index
            count = X[r, c]

            prob_pos = log(self.count_positive[0, c]) - log(self.deno_pos)
            possum = possum + count * prob_pos

            prob_neg = log(self.count_negative[0, c]) - log(self.deno_neg)
            negsum = negsum + count * prob_neg
            pass
        predicted_prob_positive = exp(possum - self.LogSum(possum, negsum))
        predicted_prob_negative = exp(negsum - self.LogSum(possum, negsum))

        if flag==False:
            if predicted_prob_positive > probThresh:
                pred_labels.append(1.0)
            else :
                pred_labels.append(-1.0)

        else :
            if possum > negsum:
                # Predict positive
                pred_labels.append(1.0)
            else:
                # Predict negative
                pred_labels.append(-1.0)

    return pred_labels
```

Code Snippet of Naïve Bayes Predicting Probability of reviews:

```
def PredictProb(self, test, indexes):

    for i in indexes:
        # TO DO: Predict the probability of the i_th review in test being positive review
        # TO DO: Use the LogSum function to avoid underflow/overflow
        z = test.X[i].nonzero()
        sum_positive = self.P_positive
        sum_negative = self.P_negative
        predicted_label = 0

        for j in range(len(z[0])):
            row_index = i
            col_index = z[1][j]
            count = test.X[row_index, col_index]
            #print('count::'%count)

            prob_pos = log(self.count_positive[0, col_index])
            sum_positive = sum_positive + count * prob_pos

            prob_neg = log(self.count_negative[0, col_index])
            sum_negative = sum_negative + count * prob_neg

        predicted_prob_positive = exp(sum_positive - self.LogSum(sum_positive, sum_negative))
        predicted_prob_negative = exp(sum_negative - self.LogSum(sum_positive, sum_negative))

        if sum_positive > sum_negative:
            predicted_label = 1.0
        else:
            predicted_label = -1.0

        #print test.Y[i], test.X_reviews[i]
        # TO DO: Comment the line above, and uncomment the line below
        print(test.Y[i], predicted_label, predicted_prob_positive, predicted_prob_negative, test.X_reviews[i])
```

Part 3: Precision and Recall

I have implemented the EvalRecall and EvalPrecision methods in Eval class. I have calculated Recall and precision values for probThresh: 0.2,0.4,0.6,0.8. and plotted the curve as Precision against Recall for positive and negative labels. Plotted curves for positive and negative class have been saved in Pos_class Neg_class respectively.

Code Snippet for EvalRecall Method

```
def EvalRecall(self):
    prediction = list(self.pred)
    FN=0
    TP=0
    FP=0
    TN=0
    actual = list(self.gold)
    for i in range(len(actual)):
        if prediction[i]== -1 and actual[i]==1:
            FN += 1
        elif prediction[i]== 1 and actual[i]==1:
            TP += 1
        elif prediction[i]== 1 and actual[i]==-1:
            FP += 1
        elif prediction[i]== -1 and actual[i]==-1:
            TN +=1

    return float(TP/(TP+FN)), float(TN/(TN+FP))
```

Code Snippet for EvalPrecision Method:

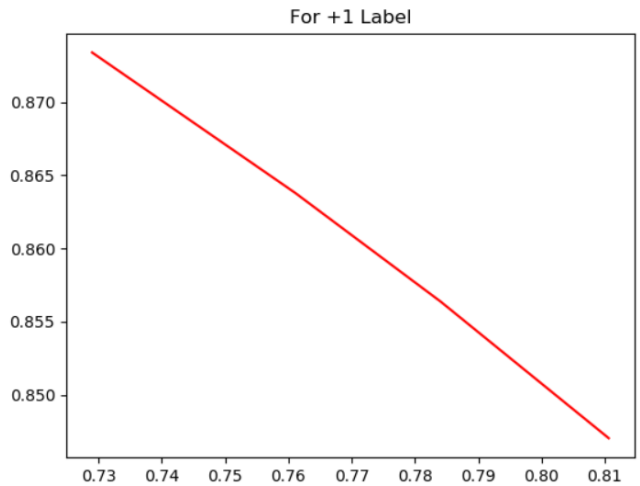
```
def EvalPrecision(self):
    prediction = list(self.pred)
    FP=0
    TP=0
    FN=0
    TN=0
    actual = list(self.gold)
    for i in range(len(actual)):
        if prediction[i]== -1 and actual[i]==1:
            FN += 1
        elif prediction[i]== 1 and actual[i]==1:
            TP += 1
        elif prediction[i]== 1 and actual[i]==-1:
            FP += 1
        elif prediction[i]== -1 and actual[i]==-1:
            TN +=1
    return float(TP/(TP+FP)), float(TN/(TN+FN))
```

Code snippet for Precision-Recall Curve

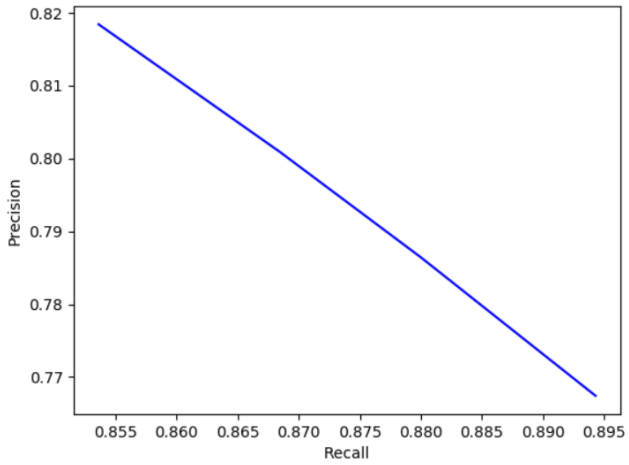
```
def Eval(self, test):
    Y_pred = self.PredictLabel(test.X,0,True)
    recall_pos = []
    recall_neg = []
    precision_pos = []
    precision_neg =[]
    ev = Eval(Y_pred, test.Y)
    #print(ev.EvalRecall())
    #print(ev.EvalPrecision())
    probThresh = [0.2,0.4,0.6,0.8]
    for i in range(len(probThresh)):
        pred = self.PredictLabel(test.X,probThresh[i],False)
        ev = Eval(pred, test.Y)
        #ev.EvalRecall()
        #print('Threshold Value %f'%probThresh[i])
        recallP,recallN=ev.EvalRecall()
        recall_pos.append(recallP)
        recall_neg.append(recallN)
        precisionP,precisionN=ev.EvalPrecision()
        precision_pos.append(precisionP)
        precision_neg.append(precisionN)
    plt.pause(0.1)
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    fig=plt.figure()
    plt.title('For +1 Label')
    plt.plot(recall_pos,precision_pos,'r')
    fig.savefig('Pos_class.png')
    plt.title('For -1 Label')
    fig1=plt.figure(1)
    plt.plot(recall_neg,precision_neg,'b')
    fig1.savefig('Neg_class.png')

    return ev.Accuracy()
```

Precision-Recall Curve for +1 class for all threshold values:



Precision-Recall Curve for -1 class for all threshold values:



Relationship Observation:

From given data set, I have plotted above graphs, which shows us that as Recall increases Precision decreases i.e inverse relationship is present between Precision and Recall.

Part 4: Features

In this, Firstly I have calculated the word frequency of word in specific category.Then I calculated the weight of each word.

Code Snippet for calculating weight for word and displaying top 20 words according to their weight.

```
def WordWeight(self,directory,testdata):
    pFiles = os.listdir("%s/pos" % directory)
    wordFreq ={}
    wordWt ={}
    #unwanted_chars = ".,-_()"
    stop_words = set(stopwords.words('english'))
    for i in range(len(pFiles)):
        f = pFiles[i]
        lines = ""
        for line in open("%s/pos/%s" % (directory, f), encoding="utf8"):
            lines += line
            wordCounts = Counter([testdata.vocab.GetID(w.lower()) for w in line.split(" ")])
            for (wordId, count) in wordCounts.items():
                if wordId >= 0:
                    word = testdata.vocab.GetWord(wordId)
                    if word not in stop_words:
                        #word = word.strip(unwanted_chars)
                        if word not in wordFreq:
                            wordFreq[word] = 1
                        else :
                            wordFreq[word] += 1

    self.P_positive = exp(self.P_positive)
    self.P_negative = exp(self.P_negative)

    for (w,freq) in wordFreq.items():
        wordWt[w]=float((freq/(self.total_positive_words))*self.P_positive)
    Top_Wt=(sorted(wordWt.items(), key=lambda x:-x[1]))[0:21]
    print('Top 20 Positive words')
    print(Top_Wt)
```

```
#for negative words:
nFiles = os.listdir("%s/neg" % directory)
wordFreqN ={}
wordWtN ={}
unwanted_chars = " (and so on)"
for i in range(len(nFiles)):
    f = nFiles[i]
    lines = ""
    for line in open("%s/neg/%s" % (directory, f), encoding="utf8"):
        lines += line
        wordCounts = Counter([testdata.vocab.GetID(w.lower()) for w in line.split("
        #print(wordCounts)
        a=testdata.vocab.GetWord(192)
        #print(a)
        #userinput = input('Enter')
        for (wordId, count) in wordCounts.items():
            if wordId >= 0:
                word = testdata.vocab.GetWord(wordId)
                if word not in stop_words:
                    #word = word.strip(unwanted_chars)
                    if word not in wordFreqN:
                        wordFreqN[word] = 1
                    else :
                        wordFreqN[word] += 1
self.P_positive = exp(self.P_positive)
self.P_negative = exp(self.P_negative)
for (w,freq) in wordFreqN.items():
    wordWtN[w]=float((freq/(self.total_negative_words))*self.P_positive)
Top_Wt=(sorted(wordWtN.items(), key=lambda x:-x[1]))[0:21]
print('Top 20 Negative words')
print(Top_Wt)
```

Top 20 Positive words

[('/><br', 0.0012022188153159362), ('one', 0.0011035266887845184), ('film', 0.0010176780581715728), ('movie', 0.0010092283898041567), ('like', 0.000834658241333344), ('see', 0.0006822262239851609), ('good', 0.0006585671525563963), ('great', 0.0006455546632705758), ('would', 0.0005818441637802598), ('really', 0.0005766053693924619), ('also', 0.000572380535208754), ('even', 0.0005512563642902143), ('story', 0.000546355556637113), ('much', 0.0005112049362286628), ('time', 0.0004976854668407973), ('first', 0.0004936296260244377), ('get', 0.0004880528449019432), ('well', 0.00048636291122846004), ('best', 0.0004701395479630215), ('many', 0.00046726666071810006), ('/>the', 0.0004415796688811557)]

Top 20 Negative words

[('/><br', 0.0042398957868656955), ('movie', 0.004139340382137443), ('one', 0.0036422767337648313), ('like', 0.0034457366245232465), ('film', 0.0032537672154965822), ('even', 0.0027309933784150416), ('would', 0.0025155991307869103), ('good', 0.0023367704848781426), ('really', 0.002147657763485804), ('bad', 0.00203681828327398), ('see', 0.002029390895424734), ('get', 0.0020059660568232662), ('much', 0.0019242647904815608), ('make', 0.001900839951880093), ('could', 0.0018916985514502518), ('time', 0.0016974437923161275), ('made', 0.0016185992136087475), ('people', 0.0015791769242550577), ('first', 0.001509473745977519), ('/>the', 0.0014946189702790271), ('story', 0.0014883342574835115)]