JIS COLLEGE OF ENGINEERING

Kalyani, Nadia- 741235

PROJECT VI- Sentiment Analysis using Logistic Regression

INFORMATION TECHNOLOGY

Semester: 6

Year: 3

Project Guide: Ms Monali Sanyal

Team CupCakes:

- Aman Shaw (123210904201)
- Anshu Kumar Jha (123210904202)
- Prosen Roy (123210904206)
- Subham Gupta (123210904207)
- Sumit Prasad Gupta (123210904208)

Sentiment Analysis

Sentiment analysis(or opinion mining) is a natural language processing technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs.

Project Details

- Building sentiment analysis model
- Building Features
- Building Logistic Regression model

Firstly, we will understand how to deal with the text data and produce feature set from the data.

Second, we build our own Logistic Regression model to predict sentiment of the data.

Project Snaps


```
#importing necessary libs

import nltk, re, string
from nltk.corpus import stopwords, twitter_samples
import numpy as np
import pickle


#Preprocessing of the tweets that is our data

def process_tweet(tweet):
  stemmer = nltk.PorterStemmer();
  stopwords_english = stopwords.words('english')
  tweet = re.sub(r'\$\w*', '', tweet)
  tweet = re.sub(r'^RT[\s]+', '', tweet)
  tweet = re.sub(r'https?:\/\/.*[\r\n]*', '', tweet)
  tweet = re.sub(r'#', '', tweet)
  tokenizer = nltk.TweetTokenizer(preserve_case=False, strip_handles=True, reduce_len=True)
  tweet_tokens = tokenizer.tokenize(tweet)

  tweets_clean = []
  for word in tweet_tokens:
    if (word not in stopwords_english and
        word not in string.punctuation):
      stem_word = stemmer.stem(word)
      tweets_clean.append(stem_word)
  return tweets_clean



#build_freqs func

def build_freqs(tweets,ys):
  """Build frequencies.
  Input:
    tweets: a list of tweets
      ys: an m x 1 array with sentiment label of each tweet
          (0 or 1)
  Output:
    freqs: a dictionary mapping each (word, sentiment) pair to its frequency
  """
  yslist = np.squeeze(ys).tolist()

  freqs = {}
  for y, tweet in zip(yslist, tweets):
    for word in process_tweet(tweet):
      pair = (word, y)
      if pair in freqs:
        freqs[pair] += 1
      else:
        freqs[pair] = 1
  return freqs


import nltk
nltk.download('twitter_samples')
```



```
nltk.download('stopwords')
tweets = ['i am happy', 'i am tricked', 'i am sad', 'i am tired', 'i am tired']
ys = [1, 0, 0, 0, 0]
res = build_freqs(tweets, ys)
print(res)
```

```
[nltk_data] Downloading package twitter_samples to /root/nltk_data...
[nltk_data]   Unzipping corpora/twitter_samples.zip.
{('happi', 1): 1, ('trick', 0): 1, ('sad', 0): 1, ('tire', 0): 2}
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
#select the set of positive and negative tweets
nltk.download('twitter_samples')
all_positive_tweets = twitter_samples.strings('positive_tweets.json')
all_negative_tweets = twitter_samples.strings('negative_tweets.json')
```

```
[nltk_data] Downloading package twitter_samples to /root/nltk_data...
[nltk_data]   Package twitter_samples is already up-to-date!
```

```
#splittling data into two pieces (for training and testing)
test_pos = all_positive_tweets[4000:]
train_pos = all_positive_tweets[:4000]
test_neg = all_negative_tweets[4000:]
train_neg = all_negative_tweets[:4000]


train_x = train_pos + train_neg
test_x = test_pos + test_neg


#combining positive and negative labels(building y - target var)
train_y = np.append(np.ones((len(train_pos), 1)), np.zeros((len(train_neg), 1)), axis=0)
test_y = np.append(np.ones((len(test_pos), 1)), np.zeros((len(test_neg), 1)), axis=0)


#create frequency dictionary
freqs = build_freqs(train_x, train_y)


#checking outputs

print("type(freqs) = " + str(type(freqs)))
print("len(freqs) = " + str(len(freqs.keys())))
```

```
type(freqs) = <class 'dict'>
len(freqs) = 11337
```

```
#test the function

print('this is an example of positive tweet: \n', train_x[22])
print('\n this is an example of the processed version of the tweet: \n', process_tweet(train_x[22]))
```

```
this is an example of positive tweet:
 @gculloty87 Yeah I suppose she was lol! Chat in a bit just off out x :))
```



```
this is an example of the processed version of the tweet:
['yeah', 'suppos', 'lol', 'chat', 'bit', 'x', ':)']
```

```
#logistic regression
#sigmoid func

def sigmoid(z):
  """
  Input:
    z: is the input (can be scalar or an array)
  Output:
    h: the sigmoid of z
  """
  zz = np.negative(z)
  h = 1 / (1 + np.exp(zz))
  return h


#cost func and gradient

def gradientDiscent(x, y, theta, alpha, num_iters):
  """
  Input:
    x: matrix of features which is (m,n+1)
    y: corresponding labels of the input matrix x, dimentions (m,1)
    theta: weight vector of dimension (n+1,1)
    alpha: learning rate
    num_iters: number of iterations to train the model
  Output:
    J: the final cost
  """

  m = x.shape[0]
  for i in range(0, num_iters):
    z = np.dot(x, theta)
    h = sigmoid(z)
    cost = -1. / m * (np.dot(y.transpose(), np.log(h)) + np.dot((1 - y).transpose(), np.log(1 - h)))
    theta = theta - (alpha / m) * np.dot(x.transpose(), (h - y))
  cost = float(cost)
  return cost, theta



#extracting features
def extract_features(tweet, freqs):
  """
  Input:
    tweet: a label of words for one tweet
    freqs: a dictionary corresponding to the frequencies of each tuple (word, label)
  Output:
    x: a feature vector of dimension (1,3)
  """

  word_1 = process_tweet(tweet)
  x = np.zeros((1, 3))
```

Project Snaps


```
    #bias term is set to 1
    x[0, 0] = 1

    for word in word_1:
      #incrementing the word count for positive label 1
      x[0, 1] += freqs.get((word, 1.0), 0)
      #incrementing the word count for negative label 0
      x[0, 2] += freqs.get((word, 0.0), 0)

    assert (x.shape == (1, 3))
    return x


#test on training data
tmp1 = extract_features(train_x[22], freqs)
print(tmp1)
```

```
    [[1.000e+00 3.006e+03 1.240e+02]]
```

```
#training the model

#collect the features 'x' and stack them into a matrix 'x'
X = np.zeros((len(train_x), 3))
for i in range(len(train_x)):
  X[i, :] = extract_features(train_x[i], freqs)

#traing labels corresponding to X
Y = train_y

#applying gradient descent
#these values are predefined(Andrew NG)
J, theta = gradientDiscent(X, Y, np.zeros((3, 1)), 1e-9, 1500)



def predict_tweet(tweet, freqs, theta):
  """
  Input:
    tweet: a string
    freqs: a dictionary corresponding to the frequencies of each tuple (word, label)
    theta: (3,1) vector of weights
  Output:
    y_pred: the probability of a tweet being positive or negative
  """
  #extract_features of tweet and store into x
  x = extract_features(tweet, freqs)
  y_pred = sigmoid(np.dot(x, theta))

  return y_pred



def test_logistic_regression(test_x, test_y, freqs, theta):
  """
  Input:
    test_x: a list of tweets
```



```
    test_y: (m, 1) vector with the corresponding labels for the list of tweets
    freqs: a dictionary with the frequency of each pair
    theta: weight vector of dimension (3, 1)
  Output:
    accuracy: (# of tweets classified correctly) / (total # of tweets)
  """
  #list for storing prediction
  y_hat = []

  for tweet in test_x:
    #get the label prediction for the tweet
    y_pred = predict_tweet(tweet, freqs, theta)
    if y_pred > 0.5:
      y_hat.append(1)
    else:
      y_hat.append(0)

  accuracy = (y_hat == np.squeeze(test_y)).sum() / len(test_x)
  return accuracy


tmp_accuracy = test_logistic_regression(test_x, test_y, freqs, theta)
print(f"Logistic regression model's accuracy ={tmp_accuracy:.4f}")
```

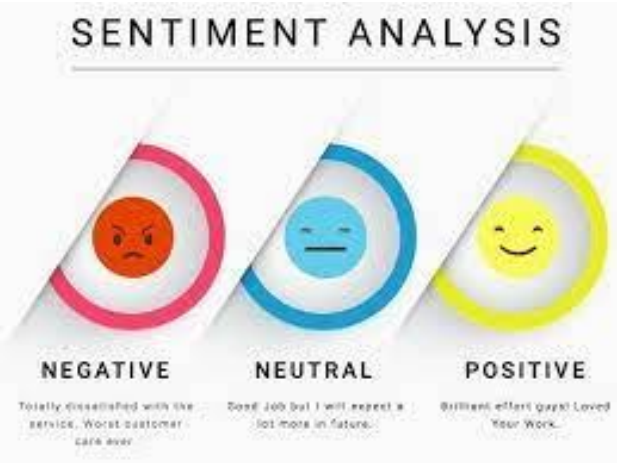
```
    Logistic regression model's accuracy =0.9950
```

```
#predicting with own tweet

def pre(sentence):
  yhat = predict_tweet(sentence, freqs, theta)
  if yhat > 0.5:
    return 'Positive sentiment'
  elif yhat == 0:
    return 'Neutral sentiment'
  else:
    return 'Negative sentiment'


my_tweet = 'it is so hot today but it is the perfect day for a beach party'
res = pre(my_tweet)
print(res)
```

```
    Positive sentiment
```

Future Scopes

- We can build a web application on which we can mount our model.
- And then we can deploy the web application on a platform.
- We can try and build this model for different languages.



References

https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Thank You