Final Project

COSC 6342 Machine Learning – Fall 2017

University Of Houston

Sentiment Analysis Of Data Using Machine Learning

Priscilla Imandi

PS ID : 1619570

Contents

SURVEY

INTRODUCTION AND BACKGROUND

Machine Learning is a field of Computer Science that gives computers the ability to learn without being explicitly programmed. It has been doing various tasks for us from detecting diseases to classifying objects over time. This is achieved by building models that provide solutions/answers to our problems. This model is created using a process called as training. Now, the goal of training is to create an accurate model that provides correct answers most of the time. But in order to train our model we need data to train it. This data is collected from various sources- real world data (Data of Cancer patients), simulated data(computational fluid dynamics). Firstly, there is a need to divide the data into separate parts training data and testing data .This training data cannot be given directly to our model, features have to be extracted from it so that they are understandable by the model and also the model should be able to provide solutions based on these features. Let us say the color and length of an object determines the class/ category into which the object falls into. This step is very important because the quality and quantity of data will directly determine how good our predictive model can be. Therefore, our training data will be a table of features and the class of the example. We can also visualize the data so as to find any patterns or relationship in the data, to see any imbalance/outliers in the data. The next step in the workflow is choosing the appropriate model, according to the data that we have. The main step training – where we incrementally improve our Model's ability to predict. For the model, we have weights and biases matrices. According to the training we adjust the values of these weights and biases. If there is a wrong classification, we adjust these values according to the right output that is required, so that we have more accurate predictions on the next time around. Now we test the model after all the training examples are done. This testing is done on the test/evaluation data which has never been used for training. This metric allows us to see how our model might perform against data that it has not yet seen. This is representative of how the model might perform in the real world.

Due to rapid increase in the amount of user-generated data on social media platforms like Twitter, several opportunities and new open doors have been prompted for organizations that endeavor hard to keep a track on customer reviews and opinions about their products. Twitter is a huge fast emergent micro-blogging social networking platform for users to express their views about politics, products sports etc. These views are useful for businesses, government and individuals. Hence, tweets can be used as a valuable source for mining public's opinion. Sentiment analysis is

a process of automatically identifying whether a user-generated text expresses positive, negative or neutral opinion about an entity (i.e. product, people, topic, event etc.). The current project is about Sentiment Analysis of a given text using classifiers. Sentiment Analysis is the combination of Natural Language processing and Text Analysis to get the sentiment or the feeling of the user based on the words present in the text data.

- A review example: *(1) I bought an iPhone a few days ago. (2) It was such a nice phone. (3) The touch screen was really cool. (4) The voice quality was clear too. (5) Although the battery life was not long, that is ok for me. (6) However, my mother was mad with me as I did not tell her before I bought it. (7) She also thought the phone was too expensive, and wanted me to return it to the shop. . . .*

Sentiment Analysis

It is the automated extraction of subjective content from digital text and predicting the subjectivity such as positive or negative. Subjectivity is the expression of somebody's emotions, opinions and sentiments.

There are multiple ways to do it, but here is the general pipeline that can be used as a start. Don't get me wrong I'm skipping the details and making some simplifications here.

Step 1: Get some sentiment examples

As for every supervised learning problem, the algorithm needs to be trained from labeled examples in order to generalize to new data. Twitter was used as a source of text in this project, we gathered tweets for major companies by using keywords and dates to scrap them. Twitter API (although not being the most efficient way)was used to gather text data.

Step 2: Extract features from examples And Clean The Data

Transform each example into a feature vector. The simplest way to do it is to have a vector where each dimension represents the frequency of a given word in the document. Stopwords removal, punctuation removal, stemming etc were few of the techniques used to clean the text. This is common to every text analysis problem.

Step 3: Train the parameters

This is where your model will learn from the data. It is then important to fine the sentiment of the statement, there are many smart and reliable algorithms out there to tag statements with complex or simple sentiments. Some sentiment algorithm would also give you sentiments like anxiety, sadness etc while the most commonly used are positive, negative and neutral. Natural language

processing is used to tag each word with a sentiment and the overall score or sentiment that the statement would get depends on the underlying word sentiments. There are multiple ways of using features to generate an output, but one of the simplest algorithms is logistic regression. Other well-known algorithms are Naive Bayes, SVM, Decision Trees and Neural Networks, but I'm going to use logistic regression as an example here.

In the simplest form, each feature will be associated with a weight. Let's say the word "love" has a weight equal to +4, "hate" is -10, "the" is 0 ... For a given example, the weights corresponding to the features will be summed, and it will be considered "positive" if the total is > 0, "negative" otherwise. Our model will then try to find the optimal set of weights to maximize the number of examples in our data that are predicted correctly.

If you have more than 2 output classes, for example if you want to classify between "positive", "neutral" and "negative", each feature will have as many weights as there are classes, and the class with the highest weighted feature sum wins.

Step 4: Test the model

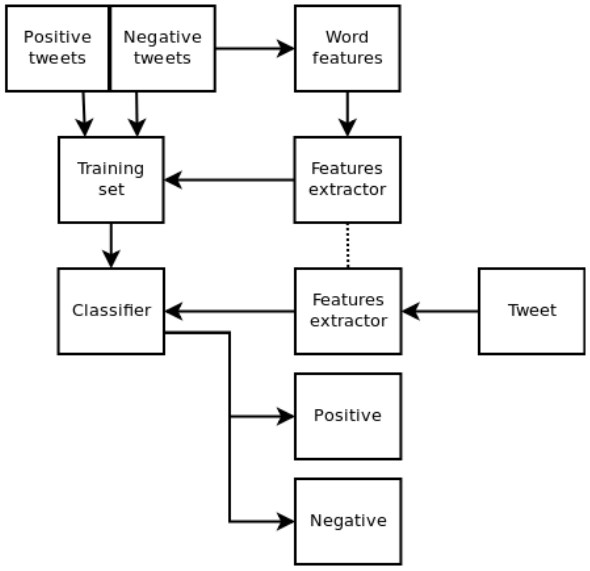
After we have trained the parameters to fit the training data, we have to make sure our model generalizes to new data, because it's really easy to overfit. The general way of regularizing the model is to prevent parameters from having extreme values.

Going further

One of the big cons of our model is that it doesn't take into account the order of words in the document, since the feature vector is showing only the frequency of the words. For example, the sentences "good guy beats bad guy" and "bad guy beats good guy" have the same feature representations but have a different sentiment.

One way to overcome this problem is to generate more features, like the frequency of n-grams or the syntactic dependency between words. But my favorite model by Socher et al. takes a completely different approach. It uses the syntactic structure of the document to build up a vector representation by combining recursively the vector representations of words.

DEVELOPMENT OF THE IDEA



First we get the data through our Twitter account using the twitter api tweepy.

self.api = tweepy.API(self.auth)

Now we can get the streaming data from online by authenticating using tokens and keys.

Now we can tokenize the data using NLTK

Nltk.tokenize()

Now apply any classification algorithm like Naïve Bayes to classify our tweets based on polarity.

training_set = nltk.classify.apply_features(extract_features, tweets)

classifier = nltk.NaiveBayesClassifier.train(training_set)

The polarity of a tweet can be calculated by calculating the individual tweets weights and then summing up the value for the entire tweet and then checking it for a threshold and check in which category it falls into.

RESULTS

The confusion matrix for a run

Here,

True Negative = All the examples that are actually negative and classified Negative.(165)

True Positive = All the examples that are actually positive and classified Positive.(195)

True Neutral = All the examples that are actually neutral and classified neutral.(256)

False Negative = All the examples that are actually negative but are classified as positive or neutral.(12+35)

False Positive = All the examples that are actually positive but are classified as negative or neutral.(36+30)

False Neutral = All the examples that are actually neutral but are classified as positive or negative.(34+79)

Confusion Matrix

	True Negative	True Neutral	True Positive
Predicted Negative	165	34	36
Predicted Neutral	12	256	30
Predicted Positive	35	79	195

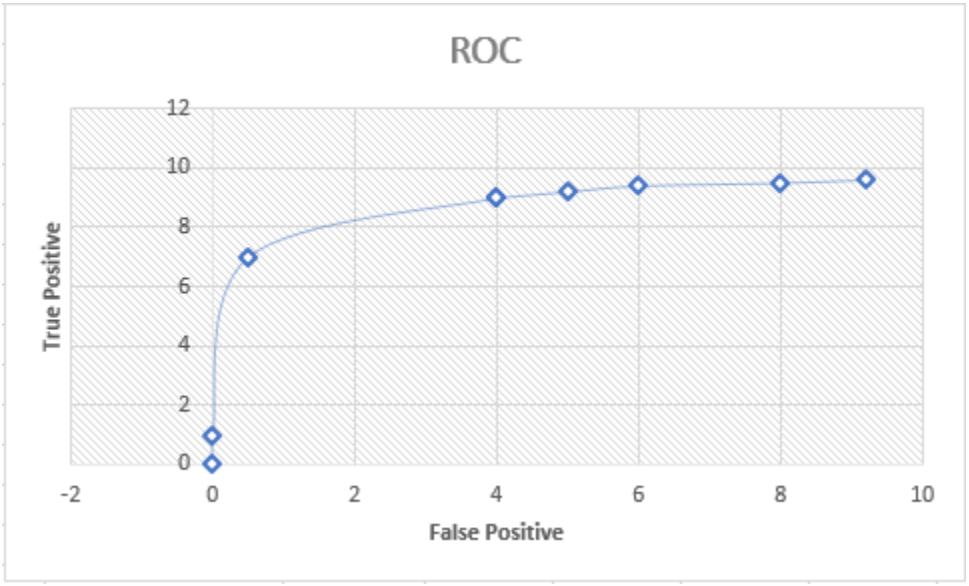
Accuracy for this Run is = $\frac{True\ Negative+True\ Neutral+True\ Positive}{True\ Negative + True\ Neutral + True\ Positive + False\ Negative + False\ Neutral + False\ Positive}$

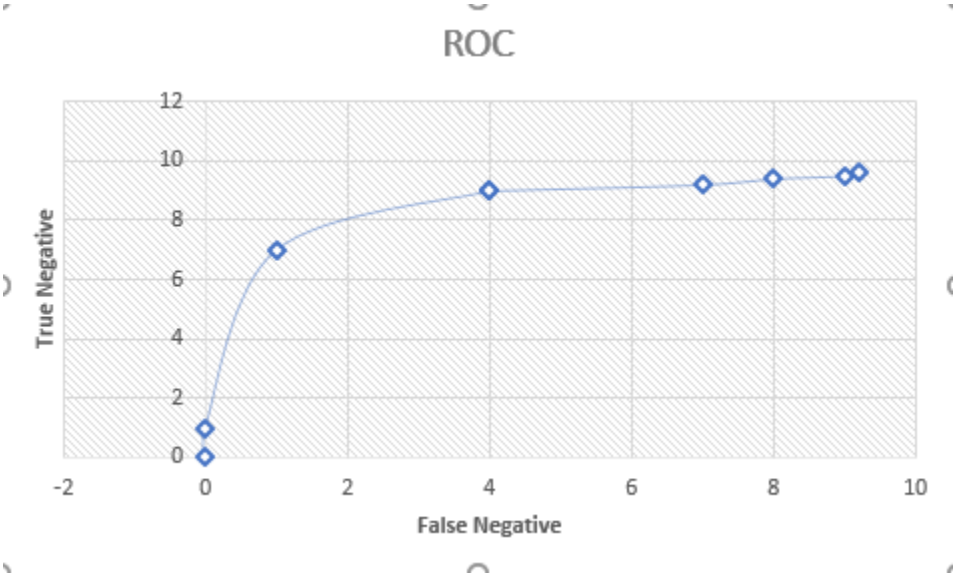
$= \frac{165+195+256}{12+35+36+30+34+79+165+195+256}$

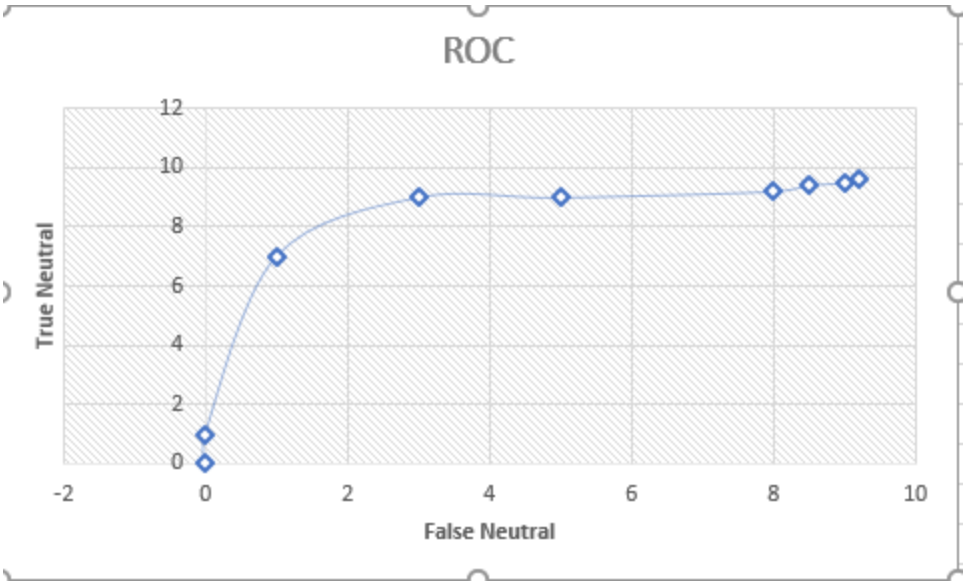
= 73.159%

ROC Curves

receiver operating characteristic curve, i.e. ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.







Experimental Output

For the word "torture", the following results are obtained, with more of negativity.

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/priscilla/PycharmProjects/TwitterMining/run_sentiment.py
Positive tweets percentage: 30.555555555555557 %
Negative tweets percentage: 25.0 %
Neutral tweets percentage: 44.44444444444444 %
```

For the word "Machine Learning", the opinion is general with different percentages.

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/priscilla/PycharmProjects/TwitterMining/run_sentiment.py
Positive tweets percentage: 35.61643835616438 %
Negative tweets percentage: 8.219178082191782 %
Neutral tweets percentage: 56.16438356164384 %
```

For the word "Love", the opinion is highly positive.

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/priscilla/PycharmProjects/TwitterMining/run_sentiment.py
Positive tweets percentage: 73.19587628865979 %
Negative tweets percentage: 7.216494845360825 %
Neutral tweets percentage: 19.587628865979383 %
```

CONCLUSION

It can be concluded that for positive words the percentage of positive tweets are more and the same is true for negative and neutral words respectively. But we cannot expect 100% of positivity for a positive word, similarly for negative and neutral words as well. There might be other words as well in the tweet which influence the polarity of the sentence.

But it has to be kept in mind that this does not always happen. There might be double negatives like "*I don't know nothing about computers."* . Such sentences actually are positive but give a negative sound because of so many negative words. So the polarity of the sentence would become negative. This is a bit difficult to avoid. In such situations, the sentences shouldbe considered in phrases, where the use of Natural Language Processing is necessary.

Public opinion is very important in the current era. Companies are influencing public opinion by generating fake news and organizations to protect their interest against scientific facts like climate change.

REFERENCES

1) https://blog.algorithmia.com/introduction-sentiment-analysis-algorithms/
2) https://en.wikipedia.org/wiki/Sentiment_analysis
3) https://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/
4) https://www.slideshare.net/sumit786raj/sentiment-analysis-of-twitter-data
5) https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets