Sentiment Analysis of Reviews using Deep and Machine Learning algorithms on Textual data

SP18-IN-CSCI-59000-30659
Big Data Analytics
Project Report

Under the guidance of: Dr. Yuni Xia

By

Prem Chand Avanigadda
Aakarsh Nadella
Bhavani Prasad Rao Ejanthkar

CONTENTS

1. Abstract

The growing popularity towards online purchases and e-commerce websites has been very well observed over the last five years. Customers rate and review the products based on their level of satisfaction. These reviews can be analyzed to extract useful hidden insights and provide a better alternative to enhance the productivity of business, by improving the quality of the product. Building a predictive model using text mining, can classify the polarity of the review automatically and further analysis of these gives the opportunity for sellers to better understand the customer needs.

2. Motivation

We had built models earlier on continuous(numerical) data which are used for both prediction and classification. But according to a survey, almost 90% of the data that is generated is of unstructured format in the form of emails, customer reviews, research papers, journals and digital libraries etc. In addition, as there is huge research undergoing on analyzing textual data, we wanted to extend our work towards that area. The main reason for us to choose Amazon product reviews dataset is the growing trend towards online purchasing. Careful examination of this dataset gives a way to understand the customer needs and to improve business value.

3. Introduction

Sentiment analysis has been practiced on various topics. For example, sentiment analysis studies for movie reviews, product reviews, news, articles, tweets, etc. In this review, specific sentiment approaches are reported. The sentiment of users on web will make impact on the product buyers, employees, owners, and product vendors. The sentiment analysis is generally performed on a text data that classifies the opinion or feelings into positive, negative and neutral. Extracting sentiment from a text is done using text mining.

Text Mining is method of deriving a high-quality information from text. Text mining involves information retrieval, lexical analysis, word frequency distribution, pattern recognition, information extraction, link and association analysis, visualization, and predictive analytics. The goal is to transform the text into data for analysis using Natural Language Processing (NLP) and analytics method. Two different approaches used for mining text are by building machine learning and deep learning models. Machine Learning in context of text mining is a set of statistical methods applied to identify opinion, sentiment, polarity of the text. Deep Learning is a broader family of machine learning methods based on learning data representation as opposed to task specific algorithms [1].

In our work, we shall build a RNN model using Long Short-Term Memory(LSTM) and Naive-Bayes classifier which are deep learning and machine learning algorithms respectively.

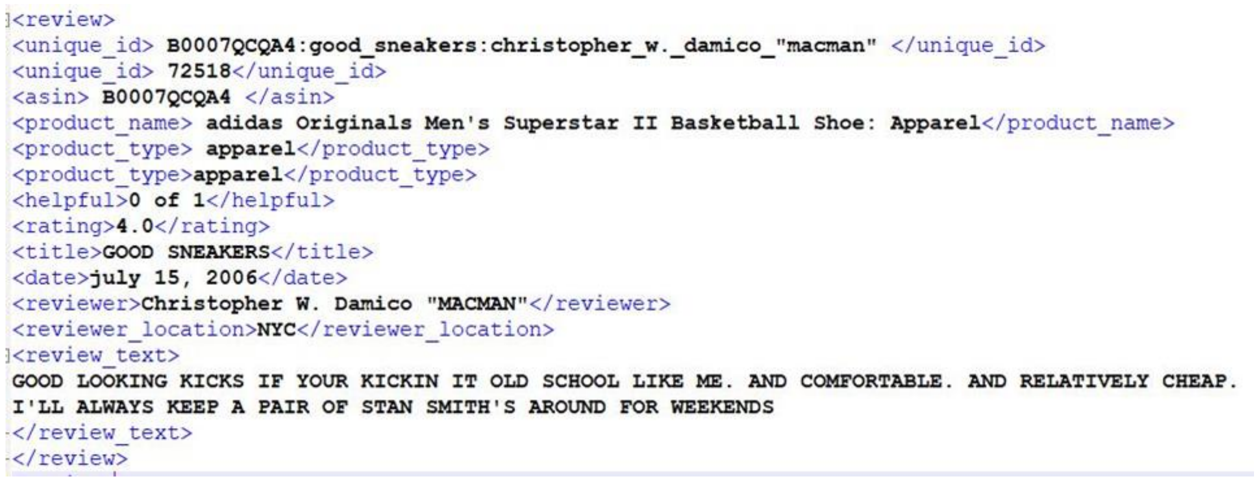
4. Related Work

In the paper "*Sentiment Analysis Using Deep Learning Techniques: A Review*" [2], author build various predictive models using Convolution Neural Networks, Deep Neural Networks, Recurrent Neural Networks(RNN), Support Vector Machines(SVM), Maximum entropy etc. on twitter data. The reason he mentioned for building deep learning models their self-learning capability. From his work he concluded that these deep learning networks give high accuracy when compared to model built using SVM. The limitation is that deep learning models need to be trained with high amount of data and is of high cost as they require GPU for computation.

5. Dataset Description

Our dataset contains product reviews from Amazon collected by Julian McAuley [3]. Initial dataset was large with 142.8 million reviews spanning from 1996 - 2014. So, we took about 5GB data containing nearly 38,546 reviews to suit our project and processing requirements. Each product review includes rating, review in text, helpfulness notes, product description, category information, price and brand name in XML format.

Dataset is divided in 24 sub categories with Books, Electronics, Movies and TV, CDs and Vinyl, Clothing, Shoes and Jewelry , Home and Kitchen, Kindle Store, Sports and Outdoors, Cell Phones and Accessories, Health and Personal Care, Toys and Games, Video Games, Tools and Home Improvement, Beauty, Apps for Android, Office Products, Pet Supplies, Automotive, Grocery and Gourmet Food, Patio, Lawn and Garden, Baby, Digital Music, Musical Instruments and, Amazon Instant Video. In this project, our focus is to analyze the review text with no regard to which category they belong. Each category is divided into positive and negative reviews. So, all positive data is merged into one file with 21,971 reviews and negative data into other file with 16575 reviews. Sample of the dataset is shown in figure 1.



```
<review>
<unique_id> B0007QCQA4:good_sneakers:christopher_w._damico_"macman" </unique_id>
<unique_id> 72518</unique_id>
<asin> B0007QCQA4 </asin>
<product_name> adidas Originals Men's Superstar II Basketball Shoe: Apparel</product_name>
<product_type> apparel</product_type>
<product_type>apparel</product_type>
<helpful>0 of 1</helpful>
<rating>4.0</rating>
<title>GOOD SNEAKERS</title>
<date>july 15, 2006</date>
<reviewer>Christopher W. Damico "MACMAN"</reviewer>
<reviewer_location>NYC</reviewer_location>
<review_text>
GOOD LOOKING KICKS IF YOUR KICKIN IT OLD SCHOOL LIKE ME. AND COMFORTABLE. AND RELATIVELY CHEAP.
I'LL ALWAYS KEEP A PAIR OF STAN SMITH'S AROUND FOR WEEKENDS
</review_text>
</review>
```

Figure 1: Overview of dataset

6. Data Preprocessing

6.1. Extracting the reviews from XML

We had data in xml document, where xml is a hierarchical data format, most natural way to represent this hierarchical format is with trees. To extract text in each tag, we had xml.etree.ElementTree package in python library, this ElementTree represents whole XML document as a tree. Element represents a single node in the tree, which simplifies the iteration through XML tags. The project goal is to identify the sentiment of the text. So, we need only text reviews from the XML, which present in the <review_text> tag. By iterating through < review_text> in whole XML document we created two separate CSV (comma separated value) files containing positive and negative text respectively.

6.2. Text Preprocessing

Pre-processing is the process of cleaning and preparing the text for training the classifier. review texts contain usually lots of noise such as emojis, special characters, and symbols. This noise may impact the efficiency of the classifier. Each word in the review is considered as one dimension. Keeping those words makes the dimensionality of the problem high and hence the classification is more difficult. So, preprocessing greatly affects the efficiency and fastness of the classifier.

We had done word tokenization to ease the process of preprocessing, we used 're’ package in python which deals with regular expressions, it is used to convert each word to lowercase and restricted the characters in a word between a-z,0-9. This will help to eliminate Latin words and special characters. Still, we shall have noise in the form of stop words (‘and’, they’, ‘where’ etc.) and verbs, stemming is needed to remove verb forms, which will decrease the dimensionality of a word. Stemming make sure all the different verb forms will be in one type. We used **nltk** (natural language toolkit) python package, which had pre-defined nltk.corpus which contain set of stop words, and functions to perform stemming. By applying these functions on the list of words we can remove the noise in the text.

7. Implementation

7.1. Environment

For implementation of our work, we used Python programming as it contains many predefined libraries for handling the data efficiently, which simplifies our task. As we wanted to build a deep learning algorithm, we used TensorFlow package which is based on flow graphs, scaling machine learning code. This makes the numerical computations faster for a high number of iterations, reducing the need for loops. Also, this computes the gradient automatically within less time, to reduce the loss.

7.2. Word Vector

Before building the model, one of the important steps is vectorization of words. Google has built **Word2Vec** tool for facilitating this purpose. According this tool, each word is assigned a unique vector representation of the specified dimension, which is also referred to as weights. These weights are assigned based on many factors such as the words that are repeated most number of times are assigned less weights, words that appear in similar context share same semantic meaning and similar vector representation, based on the window size surrounding the context word. Dimension is specified based on total number of unique words present in the text corpus and the average number of words present in each document or review.

The two different architectures designed for Word2Vec implementation are Continuous Bag of Words(CBOW) and Skip-Gram. The prior one uses the context words (words in window) to predict the center word while the later one predicts the context words using center word. For this project, we used GloVe, an application of Word2Vec, for vectorization of words. This model is built on Google News Dataset assigning vector representation for nearly 400,000 words with dimensionality of 50. Id's are also assigned for each word which we need to map with those words that are in our reviews.

7.3. Recurrent Neural Networks

As it is textual data and the output depends on both present and previous inputs, we need a technique that can handle previous data stored in a memory. Recurrent Neural Network(RNN) model facilitates this purpose. In a traditional neural network, all the inputs are sent to the hidden layer randomly at once, while in RNN the inputs are sent to hidden layer sequentially and additionally a timestamp is associated with each input. The limitation of this model is that it can store previous information or input only up to certain steps, say two. So, to handle this issue we use RNN using Long Short-Term Memory (LSTM) architecture. Using this we can define a window size, which specifies the number of previous inputs it needs to remember.

LSTM has a gated architecture. It has 3 different gates such as input gate, forget gate, output gate and a memory container. At each step, the current input from the input gate is fed to the previous inputs stored in the memory container. An activation function is applied to the resultant to determine the context and predict the next word. Considering result from the above, model determines to know whether the context should be updated or not. Forget gate stores the changes or updates of the context. The result will be sent through the output gate for the next iteration.

The implementation of the above model starts with defining the maximum words a review can have as 300, from the analysis. Later each word in the review is mapped with ids specified with that of the word vector model, thus creating id matrix for all the reviews. In the next step, the entire dataset is divided into training and testing in the ratio 75:25 and labels are appended according for positive and negative reviews as 1 and 0 respectively. The predictive model is built now with window size of 64(i.e. Keeping in memory the previous 64 inputs) and the model is trained over 100,000 iterations with a learning rate of 0.001. The testing dataset is sent as batches of size 24(24 reviews) and are randomly chosen. Ten such random batches are chosen, to determine the accuracy of the model.

7.4. Naive-Bayes Classification

The naive Bayes classifier is based on Bayes theorem which relay on probability and naive independent assumptions. It requires small amount of training data when compared to other alternative methods, it is very efficient and computationally less intensive due to calculation of probabilities.

$$p\left(\frac{c}{review}\right) = \frac{p\left(\frac{review}{c}\right)p(c)}{p(review)}$$

As, each review is list of words in our context. Bayes theorem modified as follows

$$p\left(\frac{c}{review}\right) = \frac{p\left(w1, w2, w3, w4 \ldots \frac{wn}{c}\right)p(c)}{p(w1, w2, w3, w4, w5..wn)}$$

p(c) represents probability of classes. In our context we had positive and negative classes. The main assumption in text classification is every word is independent to each other. However, making this exemption greatly simplifies the training and calculation time. Let us see how it works.

$$p\left(\frac{c}{review}\right) = \frac{p\left(\frac{w1}{c}\right)p\left(\frac{w2}{c}\right)p\left(\frac{w3}{c}\right)p\left(w\frac{4}{c}\right)p\left(\frac{wn}{c}\right)p(c)}{p(w1, w2, w3, w4, w5..wn)}$$

Here, each word probability p(w/c) is checked with respect to the bag of words we had from the training set, which gives probability to the class it belongs.Finally, by computing the probabilities we will get total class probability(positive or negative probability) of the review.

We had two files with positive and negative reviews, we combine both files and split the data into training and testing data with ratio 75:25 respectively. This training data is passed to 'NaiveBayesClassifier.train' function imported from nltk package to build a classifier. This classifier predicts the nature of review in test data and assign class label to the review.

Naive Bayes using Bigrams:

The problem of language detection is that human language has structure. Bigrams work by capturing this structure. Thus, certain combinations of letters are more likely in some languages than others. This is the basis of Bigram classification. By applying bigrams on English alphabets, we will get 676 possible combination of letters. So, number of combinations of letters increases dimensionality which helps to match rare word patterns to get accurate results. We used pre-defined function 'BigramCollocationFinder' from nltk.collocations to convert the word to bigrams. So, this list of bigrams is used for training the classifier.

8. Results

1. On analyzing all the reviews, it is observed that most of the reviews have more than 200 words. So, we took the maximum number of words a review can have as 300 to provide uniform analysis of all the reviews by specifying input matrix of dimension 300.

```
The total number of files is 38548
The total number of words in the files is 4113263
The average number of words in the files is 106.70496523814465
```

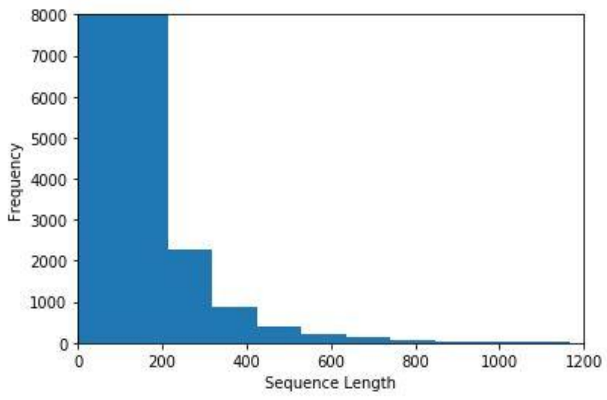


Figure 2: Analysis of reviews

2. The accuracy of the model while training over 100,000 iterations can be visualized using tensorboard as follows.

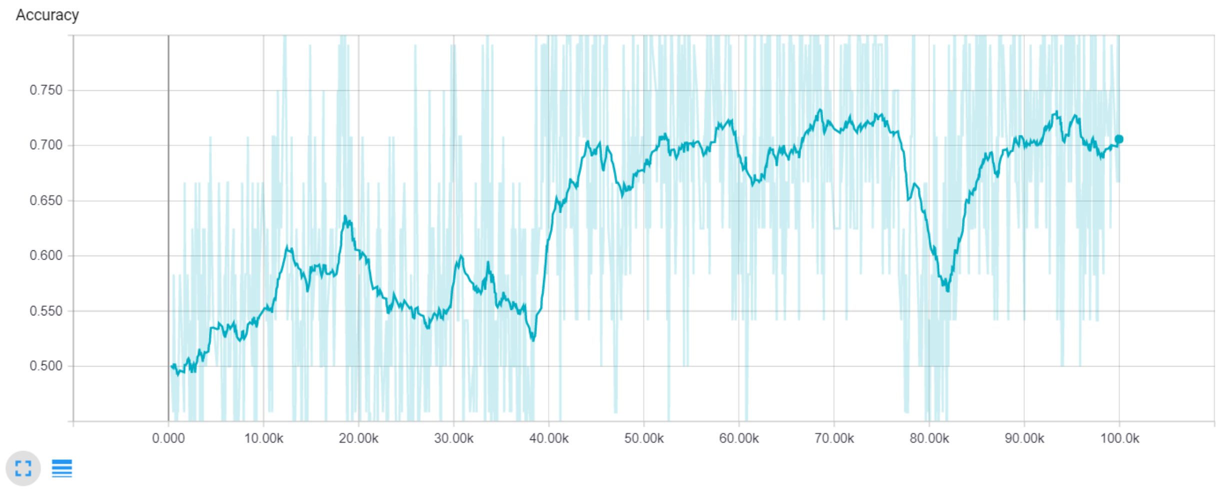
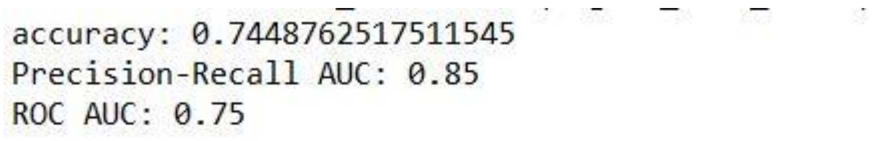


Figure 3: Accuracy while training RNN model

3. The average accuracy of the testing data when sent in 10 different batches of size 24 taken randomly, is found to be 70.83%. The reason for less accuracy of the deep learning model is due to smaller size of data on which the model is built.
4. We got accuracy of 68.5 for naive bayes without Bigrams.
5. Accuracy of 74.5% for naive bayes using Bigrams is achieved. By observing its ROC curve, area under curve is 75 percent, which shows it's efficiency of prediction.



```
accuracy: 0.7448762517511545
Precision-Recall AUC: 0.85
ROC AUC: 0.75
```

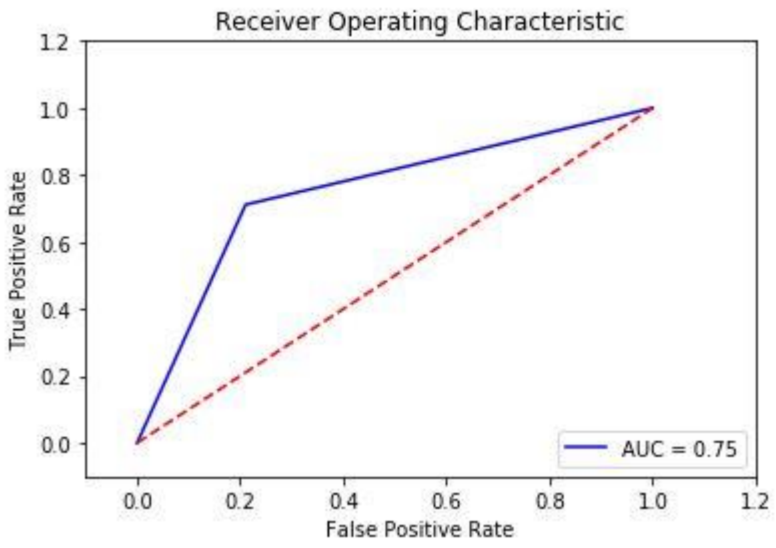


Figure 4: Receiver Operating Characteristic(ROC) curve

6. By comparing all models on each other, naive bayes using Bigrams has more accuracy in predicting sentiment of text.

Classifier	Accuracy
Naive bayes with Bigram	74.5
RNN using LSTM	70.83
Naive bayes	68.5

Table 1: Evaluation of models

9. Future Work

We would like to extend our work by building predictive models using other machine learning algorithms which might improve the accuracy. Also, to improve accuracy of the deep learning model which we built, we need to train the model using more reviews (approximately 1 lakh each of both positive and negative). Also, in the dataset, there are some reviews (say 15-20%) given by customers in Spanish. Efficient techniques to handle such data needs to be designed. Also, for improving accuracy of Naive-Bayes classifier, we will replace Bigram model with N-grams.

Role of Team Members

Task	People
1. Data Preprocessing	Bhavani Prasad Rao Ejanthkar
2. RNN using LSTM	Aakarsh Nadella
3. Naive-Bayes classifier	Prem chand Avanigadda
4. Evaluating and Comparing Algorithms	Bhavani Prasad Rao Ejanthkar
5. Writing Report	Equal contribution of All
6. Slides, Demo and Presentation	Equal contribution of All

References

[1] https://en.wikipedia.org/wiki/Deep_learning
[2] https://pdfs.semanticscholar.org/8892/24a64a5bc5f9e965f418a63b6768f7164993.pdf
[3] http://jmcauley.ucsd.edu/data/amazon/