Fall | 16

SENTIMENT ANALYSIS FROM PRODUCT REVIEWS

FINAL PROJECT REPORT

Submitted By :- Debasis Dwivedy (DDWIVEDY)

Sentiment analysis is a field to analyze the opinion and feelings of people for polarity and emotion recognition from their text or speech towards a product or services. It is an important part of the product and consumer management lifecycle where the feedback (direct or indirect) of the consumers plays an important role towards its shape and future enhancement of the product. This is a difficult task to carry out because people express their sentiments in different ways and there is no one specific way or method, which can be applied to all the different scenarios to detect emotions. For example, some people like to use emoticons, some use sarcasm, some use negative sentences, some use slangs (which could be local or global) etc. These are only to site a few ways in which people express themselves; in reality the list is long. All the mentioned sentiments are difficult to infer by using a single method.This project applies and makes a comparative analysis of different algorithms, to predict sentiments of a product review as positive, negative or neutral based on a context and the relationship between words within that context. The dataset for this project is taken from online websites like CNET and Amazon where people share their reviews of a particular product. The conclusion of this project would be to compare and analyze different available algorithms on this unique problem and suggest improvements to the same.

For L645/B659 Advanced Natural Language Processing
INDIANA UNIVERSITY BLOOMINGTON
FALL 2016

Table of Contents

INTRODUCTION

Sentiments and emotions are an important tool in a decision making process. Be it election analysis, product review for enhancement or movie review, sentiments play an important role in the future course of action. Product review by using sentiment analysis approach is difficult, as people might not have an absolute positive or negative opinion (they might like some features while hate some); they might use sarcasm or negative phrases which might be difficult to detect; compare it with some other product for which is not there in our training set and we have no idea whether the consumer holds positive or negative opinion about that., etc. For this the classifier must cope with the unstructured text of the review.

To address the above mentioned concerns we use Naïve Bayes classifier machine learning algorithm technique to predict the emotion associated with the review as positive, negative or neutral. The neutral sentiment does not mean that the consumer has no opinion; it just means that the classifier is unable to predict it and might need human intervention. After human classification, this could be added to our dataset and feedback to the system and thus the system is now able to detect the polarity based on our analysis. This is a continuous process and leads to incremental improvement of our machine learning model.

The dependencies associated with this project are mentioned below: -
1. Python 3.5
2. NLTK(for corpus tagging)
3. scikit-learn 0.19 (for machine learning)
4. gensim(deep learning)
5.pandas 0.19(for data frames)
6.matplotlib (for plotting)
7.Numpy and Scipy (for matrix operations)
8.TIMBL - 6.4.2(KARST Super computer)

PROBLEM STATEMENT AND VARIOUS SOLUTION APPROACH

- ## Product Review

  Product review is the analysis and evaluation of categorical products from different genre. A large number of online websites share product reviews (from different genre like technology, health care, beauty etc.). I have taken CNET and Amazon as out dataset for the study. These website are on a consistent look out for methods that can improve their usefulness by classifying the reviews based on rating from 1 to 5. These rating can be an outcome of multiple factors, such as if it is absolutely positive, negative or mixed. I have taken a rating greater than 3 as Positive, less than 3 as Negative and equal to 3 as Neutral. In case of mixed review it become more important to classify the product based on its effectiveness on a particular front and

failures on the others. This can help future or prospective customers to make an informed choice. In this chaotic Internet world, our goal is to make the life of the customers and ultimately the companies serving their customers easier. This will not only help the customers and companies addressing the concerns of the customers, but also the existing product companies which plan to enhance or new product companies to launch new products. As a know fact companies spend a lot of money on market research, our approach can help them make an informed choice.

- **Text Mining and Sentiment Analysis**

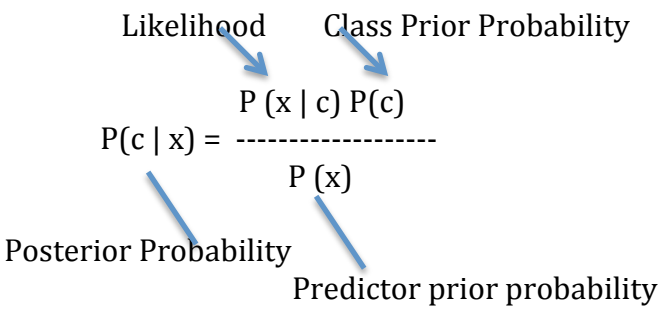
Text mining and sentiment analysis are often used interchangeably which refers to the extraction of data using data mining techniques and analyze it by natural language processing techniques using machine learning algorithms, rule based learning algorithms, supervised and unsupervised techniques or using Neural Nets (The hot topic of the present time). The main objective of this study is to determine the polarity of a given text (positive, negative or neutral) by applying one or several of the approaches mentioned above. This task gets petty complicated petty soon: - one reason, we as human beings (unlike machine) express in different and sometime weird ways.

- **Machine Learning Algorithms**

Machine learning techniques are widely used in NLP and hence sentiment analysis. Such as Support Vector Machines (SVM), Naïve Bayes (NB) and Maximum Entropy (MaxEnt) methods has been used by academia and companies to infer sentiments and opinions. Moving a step further I want to try applying Neural Nets to our problem. These techniques will provide us the information as to when and why should we apply these techniques. It's a known fact that none of the above mentioned method works well for all kind of dataset.

Naïve Bayes Classifier

The Naïve Bayes classifier is a standard probabilistic classifier based on Bayes theorem with strong independence assumption between events. Bayes theorem describes the relationship between probabilities of two events, based on conditions that might be related to the events. Bayes theorem is expressed mathematically as follows: -



Where,
c → the current event, in our case the word we are analyzing right now.
x → the event that has already occurred, in our case the words preceding '**c**'
P(c) → The prior probability of **c**
P (x) → The prior probability of **x**
P(c|x) → The probability that we will find **c** given the previous occurrence was **x**
P(x|c) → The probability that we will find **x** given the previous occurrence was **c**

All the above-mentioned probability calculations were made from the training set obtained from CNET and Amazon.

The assumption that Naïve Bayes classifier makes is of **"mutual independence".** The occurrence of one event is independent of the occurrence of another. This is called conditional independence and is denoted by "**|**". It also ignores the prior probability of predictor by assuming it has no impact on relative probability.

The probability calculation can be mathematically represented as: -
$P(c \mid X) = P(x_1 \mid c)\ P(x_2 \mid c)\ P(x_3 \mid c)\ P(x_4 \mid c)\ P(x_5 \mid c) \ldots\ P(x_n \mid c)\ P(c)$

Support Vector Machines

Support Vector Machines (SVM) is a statistical supervised learning algorithm used for text classification. SVM algorithm classify the data into two labels by learning a hyper plane from the training set that separates the data into two classes. That is, on given labeled training data(supervised learning), the algorithm gives an optimum hyper plane which categorizes new data. In this project, SVM classifier with a linear kernel of scikit-learn is used to predict the sentiment of product review.

Deep Learning (Word2Vector & Doc2Vector)

This method is applied when the training data is exhaustive or complete by using neural network model. This currently is a subject of research and gives desired result if we have large dataset, which could be used to create vector for each label and then use cosine distance with nearest neighbor classification. They represent each word as a vector and an operator (a matrix). But on the hind side these require a lot of training, but nevertheless gives good results. This is still in progress and I am working on it.

I have implemented word2vector and Doc2vector for my final project on sentiment analysis as a deep learning approach. My project deals with product reviews available on amazon and CNN. For my implementation I have taken the labeled data obtained from the Internet and used it to train my system.

Word2vec is an amazing tool to create feature vectors that captures context of words, while at the same time reducing the size of data. Word2Vec has two different methods: Continuous Bag of Words (CBOW) and Skip-gram. In the

CBOW method, the goal is to predict a word given the surrounding words. Skip-gram is the converse: we want to predict a window of words given a single word. Both methods use artificial neural networks as their classification algorithm.
So, Initially each word in the vocabulary is a random N-dimensional vector. During training, the algorithm learns the optimal vector for each word using the CBOW or skip-gram model.

These vectors now capture the context of surrounding words. This can be seen by using basic algebra to find word relations. These word vectors can be fed into a classification algorithm, as opposed to bag-of-words, to predict sentiments. The advantage is that we now have some word context, and our feature space is much lower. We also had to do very little manual feature creation since neural network was able to extract those features for us. Since text have varying length, one might take average of all word vectors as the input to a classification algorithm to classify whole text documents.

Drawbacks of word2vector if removed would be helpful: -

a) Since word order is ignored completely it become difficult to classify content as the context is removed, even with the averaging technique.

b) It is complicated to understand as it is a neural net-based approach and all the result depends on the weights and bias. Each word is represented as a number and with the weights associated in the hidden layer they are classified as similar or dissimilar.

c) Very little is actually understood why two words are considered similar.

d) The high performance output obtained is because of the hidden parameters which is associated with neural net and if removed, perform similar or even lesser than other know algorithms.

To overcome the shortcomings of our word2vector approach I applied Doc2vector deep learning model to our dataset. Doc2vec modifies the word2vec algorithm to unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents. The input to Doc2Vec is an iterator of Labeled Sentence objects. Each such object represents a single sentence, and consists of two simple lists: a list of words and a list of labels. The algorithm then runs through the sentences iterator twice: once to build the vocab, and once to train the model on the input data, learning a vector representation for each word and for each label in the dataset.

Bag of Words with Random Forest Classifier

In this approach, we learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. We'll have used the feature extraction module from scikit-learn to create bag-of-words features.

After we find the count of words occurring in the document, we vectorize the count using CountVectorizer of scikit-learn. This creates a feature vector, which is to be feed to our Random Forest Classifier. We use the Random Forest Classifier of scikit-learn to make our predictions. Random forest uses many tree-based classifier to make the predictions, hence we call it a Forest. We have to give the tree count as a parameter to it. We have used 100 as our input tree count to the forest. The more number of trees we use will give us a better result, but will take a longer time to run.

Memory Based Learner (TIMBL)

Memory-Based Learning (MBL) is based on the idea that intelligent behavior can be obtained by analogical reasoning, rather than by the application of abstract mental rules as in rule induction and rule-based processing. MBL algorithms take a set of examples (fixed-length patterns of feature-values and their associated class) as input, and produce a classifier, which can classify new, previously unseen, input patterns.

I have taken TIMBL as our memory based learner. TiMBL comprises of several Memory-Based Learning algorithms that are used to perform various Natural Language Processing Techniques like K-Nearest Neighbour, Decision-Tree Approximation, etc. TiMBL is being used in this project to classify the stance of all the tweets from all the targets.TiMBL is run on the test data file after being trained from the train data set file for various values of k in the default settings of TiMBL.
We first separate the reviews into training and testing set. We then create a Bag of Words model and vectorize the model separately for the training and testing set. Once we have a separate training and testing vectorized bag of words file, we provide this as the input file to TIMBL. I have used TIBL installed in KARST supercomputer at IU. You need to create an account (if not already present) and then load the TIMBL module. After successfully loading the module provide the input files (Training and Testing) to TIMBL.
For creating the feature vector for our TIMBL, I have taken two approaches:-
a) Simple Bag of Word approach
b) Bag of Words combined with Subjectivity Lexicon

For the second approach the subjectivity lexicon is used to give a positive or negative weight to a word based on its strength, unlike the first approach where we just count the occurrence of words in the document. I get slightly better result with the second approach.

SOLUTION APPROACH

The approach followed to reach to a solution is described in three phases: -

- Data Extraction
- Data Preprocessing
- Sentiment Analysis Algorithm

In the data extraction phase the reviews are obtained from various sources to form parsed data frames. In the Data Pre processing phase, we split the data into two sets: - Training Set and Testing Set, with the precaution that we do not include any review in both. Finally we use the training dataset to train our model using the Sentiment analysis algorithms as mentioned above and test the model on our Testing dataset to obtain result and hence accuracy.

For Naïve Bayes classifier we take the training set formed as the input to initialize the model. For each review in the training set the classifier the classifier determines how each text form shows the sentiment of the person and thus determines the output. After rigorous training we apply the model to our testing dataset and try to predict the result. The result is then matched with the actual result, which is obtained by the start rating (1 to 5) on the website and thus acts as our gold standard.

- **DATA EXTRACTION**

  Data is collected from the website like amazon using their product advertising API's. The data is obtained as text file. We need to convert it into CSV file for processing. The code to convert the text to CSV is present in "**Fetch_Data.py**" file. We have downloaded a large amazon dataset, which was used to run on Big Red 2, and the result as mentioned above was obtained from it. The larger dataset is divided into smaller dataset so that if required one could run it on his/her personal computer. If running on your personal computer run the micro version of the dataset to get quick results. The others might take a lot of time.

- **DATA PRE-PROCESSING**

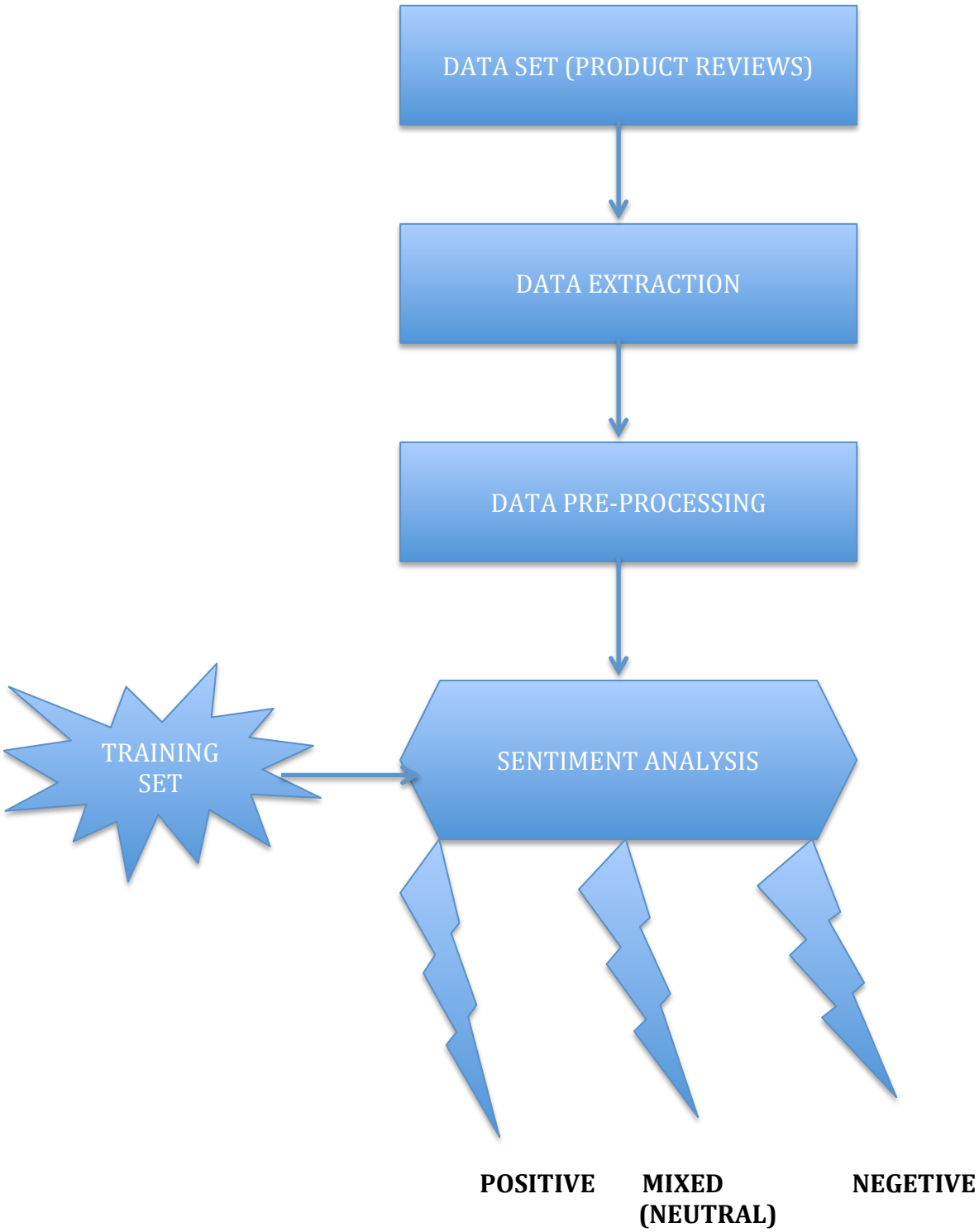
  Once we are done with the collection of review data into frames, it is passed to pre-processing phase. This phase breaks down the data into vector space where vector represents a feature. The conversion of data to vector space is done by using scikit-learn API. This API takes the textual content and converts them into "bag of words" vector, which takes each word as a feature vector. I have used the split function of scikit-learn API's "train_test_split" function to split out dataset into two models:- training and testing. The splitting is unequal with $3/4^{th}$ of the data being used for training and $1/4^{th}$ for testing.

- **CLASSIFICATION (SENTIMENT ANALYSIS)**

  The Naïve Bayes classifier of scikit-learn API's is used to classify the testing set. The classifier is first trained to recognize the pattern of the data. After the training it is tested on the Test dataset to view the results. Once the result is obtained it is compared with the result of amazon based on the ratings obtained (1 to 5). Furthermore, 5-fold validation function of scikit-learn is performed to optimize the accuracy of the Naïve Bayes Classifier model. The SVM classifier and random forest classifier with a linear kernel of scikit-learn is used to predict the reviews.

For deep learning I have used Word2Vec and Doc2Vec classifiers present in gensim package.
The memory based machine learner uses TIMBL. All the results are obtained and compared to see which approach gives us the best result.

WORKFLOW



RESULTS

All the results were carried out using python using scikit-learn, gensim and NLTK library for various algorithms used to address out problem statement.

The Naïve Bayesian approach is carried out using five fold cross validation along with the default one to achieve better results. The default Naïve Bayesian classifier results in over-fitting, which can be assumed, from the fact that it did not perform well enough compared to other algorithms. The cross validation approach is taken to reduce the over-fitting concern.

The SVM model of scikit-learn gives us the best result but taken a long time to process the data as we have used 5 fold validation with the SVM result. The SVM approach can create a problem if the dataset is huge, in which case we have to run it on a super-computer. The supercomputer takes about 40 to 50 minutes to run the program for SVM for our large dataset.

I used random forest classifier available in scikit-learn package to run our test- using bag of word model. For this experiment we have taken *n=100* as our tree count. A higher value of tree count gives us a better result but at the cost of time. The improvement of the result is not that great with increased tree count, so we concluded that *n=100* as our best-case scenario for our case.

The deep learning method provides us with good results within a short period of time. I used two deep learning approaches: - word2vec and doc2vec available in gensim package. Doc2Vec gives us a slightly better accuracy compared to Word2Vec as in the latter the context is not taken into account whereas the former takes context of word while performing classification. The deep learning approach, given we have a large enough dataset is ideal when compared to any other method taking accuracy and time into consideration.

I have also implemented TIMBL, which is a memory-based learner to do the classification. We used TIMBL with two scenarios: - one where we create a bag of words and create a feature vector and classify using k-means and the second where we use subjectivity lexicon to improve our classification. Overall the classifier does not provide a good accuracy compared to other approaches.

DEFAULT SETTING

While running this task, TiMBL was made sure to run under default settings. Under default settings, the value of n was set to 3500 and value of k was set to 1.

MPQA SUBJECTIVITY LEXICON EXTENSION

To perform this task, the features of our data set were extended to incorporate for MPQA Subjectivity Lexicon. After running TiMBL on the data set now, I noticed an increase in the accuracy.

MODEL	ACCURACY	PRECISION	RECALL
NAÏVE BAYES	87%	91%	93%
NAÏVE BAYES WITH 5-FOLD CROSS VALIDATION	81%	86%	92%
SUPPORT VECTOR MACHINE	94%	95%	97%
BOW +Random Forest Classifier	85%	85%	99%
BOW + Word2Vector	87%	92%	96%
BOW + Doc2Vector	88%	92%	96%
TIMBL without Subjectivity Lexicon	56%	68%	76%
TIMBL With subjectivity lexicon	63%	66%	79%

All the above results are obtained from running "amazon_total.txt" on BIG RED 2.

FUTURE IMPROVEMENT

The approaches mentioned above have provided a decent result, but are subject to over-fitting in some case. In the future we can try out using *Recurrent Neural Nets* to our problem and see if we can achieve any better result. This approach as far as I know has not been used for this kind of study and would be curious to find out if we can use it or modify it to address our problem statement.
It would be amazing if we could combine memory based learning and deep learning in some way to address the boundary conditions or exceptional cases, which might fail in our deep learning approach if we have rouge data.

CONCLUSION

Sentiment analysis is an active field of research, which can have many desired and undesired consequences associated with it. Given the sensitivity of the topic we can conclude that getting sentiment of a person from text or speech as input is not an easy task because of the fact that sometimes even the person or subject under study is unclear about his own sentiments. So all we do is to weight his alignment towards the matter.

There are many challenges that we face while addressing this topic such as rouge data, syntactic and semantic language rules. All these problems can be addressed using supervised machine learning approach where human intervention can come into picture when the machine is unsure.

ACKNOWLEDGMENT

The completion of this project was only possible due to the constant guidance and support of Professor Damir Cavar and our classmates. There are many books and publications that inspired me towards the completion of this project, which is mentioned in the reference section.

REFERENCES

- Jurafsky, D. and Martin, J. (2009). Speech and language processing. 1st ed. Upper Saddle River, N.J.: Pearson Prentice Hall.
- (proycon), M. (2016). TiMBL. [online] Languagemachines.github.io. Available at: https://languagemachines.github.io/timbl/ [Accessed 8 Dec. 2016].
- Chang, C.-C. and C.-J. Lin (2001). LIBSVM: A Library for Support Vector Machines.
- https://www.kaggle.com
- Zhang, Y., Jin, R. and Zhou, Z. (n.d.). Understanding Bag-of-Words Model: A Statistical Framework.
- Gimpel, K., Schneider, N., OConnor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J. and Smith, N. (n.d.). Partof- Speech Tagging for Twitter: Annotation, Features, and Experiments.
- Manning, C. (n.d.). Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (n.d.). Distributed Representations of Words and Phrases and their Compositionality.
- Chris Nicholson, A. (2016). Word2vec: Neural Word Embeddings in Java - Deeplearning4j: Open-source, Distributed Deep Learning for the JVM. [online] Deeplearning4j.org. Available at: https://deeplearning4j.org/word2vec [Accessed 8 Dec. 2016].
- https://journalofbigdata.springeropen.com/articles/10.1186/s40537-015-0015-2
- https://github.com/hackreduce/Hackathon/wiki/Amazon-review-dataset
- http://stackoverflow.com/questions/7551262/training-data-for-sentiment-analysis
- Data Mining and Analysis in the Engineering Field- Chapter 11: Machine Learning Approaches for Sentiment Analysis- Vishal Bhatnagar
- Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data-John Wiley & Sons- Chapter 7: Advanced Analytical

Theory and Methods: Classification
- http://www.saedsayad.com/naive_bayesian.htm
- http://scikit-learn.org/stable/modules/naive_bayes.html
- http://scikit-learn.org/stable/modules/cross_validation.html
- http://docs.scipy.org/doc/numpy-dev/user/index.html
- http://scikit-learn.org/stable/
- https://en.wikipedia.org/wiki/Overfitting
- Handbook of Research on Text and Web Mining Technologies by Min Song and Yi-fang Brook Wu (eds)- Chapter VII - Improving Techniques for Naïve Bayes Text Classifiers
- Y. & Freund, R.E Schapire, (1996). Experiments with a new boosting algorithm, Proceedings from ICML '96: The 13th International Conference on Machine Learning, Bari, Italy: Morgan Kaufmann, 148–156.
- https://www.enthought.com/products/canopy/
- https://docs.python.org/2/library/json.html
- http://pandas.pydata.org/
- http://matplotlib.org/
- http://docs.python-requests.org/en/latest/
- https://en.wikipedia.org/wiki/Support_vector_machine
- Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data-John Wiley & Sons- Chapter 9- Advanced Analytical Theory and Methods: Text Analysis