University of Passau

Prof. Dr. Siegfried Handschuh
Chair of Computer Science with focus on Digital Libraries and Web Information Systems

Text Mining Project

M. Abubakar Dar
78628

M. Aatif Ishtiaq
78632

Nawab Hussain
82251

Summer Term 2017

Abstract

The central idea behind this subject is to systematically identify, extract, quantify, and study affective states and subjective information for any given tweet. We will perform sentiment analysis on the given set of data. We are going to attempt to complete the 5 tasks mentioned in SemEval-2016 Task 4: Sentiment Analysis in Twitter. We will use supervised learning and try different machine learning techniques, as discussed below, using SAR-14 IMDB reviews dataset.

Index Terms: sentiment analysis, supervised learning, machine learning

Introduction

Sentiment Analysis is the process of determining the sentiment/opinion in each series of words. It aims to identify systematically the attitude of the subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event. It is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics. The Obama administration, for instance ahead of 2012 presidential election used sentiment analysis to quantify public opinion to policy announcements and campaign messages. Twitter is one of the most popular microblogging service. The status messages on twitter (reviews) most of the times express opinions about different topics or events. The purpose of this project is to build an algorithm that can accurately classify Movie reviews based on the overall sentiment of the comment. Our aim is to use what IMDB must offer and extend it as an application of text mining. We aim to use machine learning to improve our mining of text, analyze its sentiment, create different distributions to interpret the overall nature of reviews and evaluate. In our project, we will focus on multi-class classification, or Ordinal classification of our data, on a five-point scale, highly positive, positive, neutral, negative, and highly negative. To generate a model which assess polarity of a given review with a high degree of precision, following steps will be taken:

- Data Collection: This step involves collection of proper label data for the training and testing process. The given dataset has the text missing which is to be downloaded using a proper script, after which a usable data structure while be made.
- Pre-processing: It is a crucial step in the data mining process. The phrase “garbage in, garbage out” is particularly applicable to data mining and machine learning projects. It is an umbrella term that covers an array of operations to get data into a form more appropriate. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis. Before performing sentiment analysis of twitter data, we strip out any html tags, white spaces, expand abbreviations and split the reviews into lists of the words they contain.
- Bag of Words: It is a collection of words used in the documents and pays no attention to the context. The bag of words is the dictionary, which is used by the model and carries sentiment of the document.
- Model training and validation: Training a model involves providing the algorithm training data to learn from. The training data must contain the target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target and it outputs a model that captures these patterns. The trained model is evaluated with a testing data set. The testing data set is a separate portion of the same data set from which the training set is derived. The main purpose of using the testing data set is to test the generalization ability of a trained model. We will also estimate the distribution of the reviews in various classes. Our tasks include the classification of the given reviews and then, quantification of those reviews.

Method

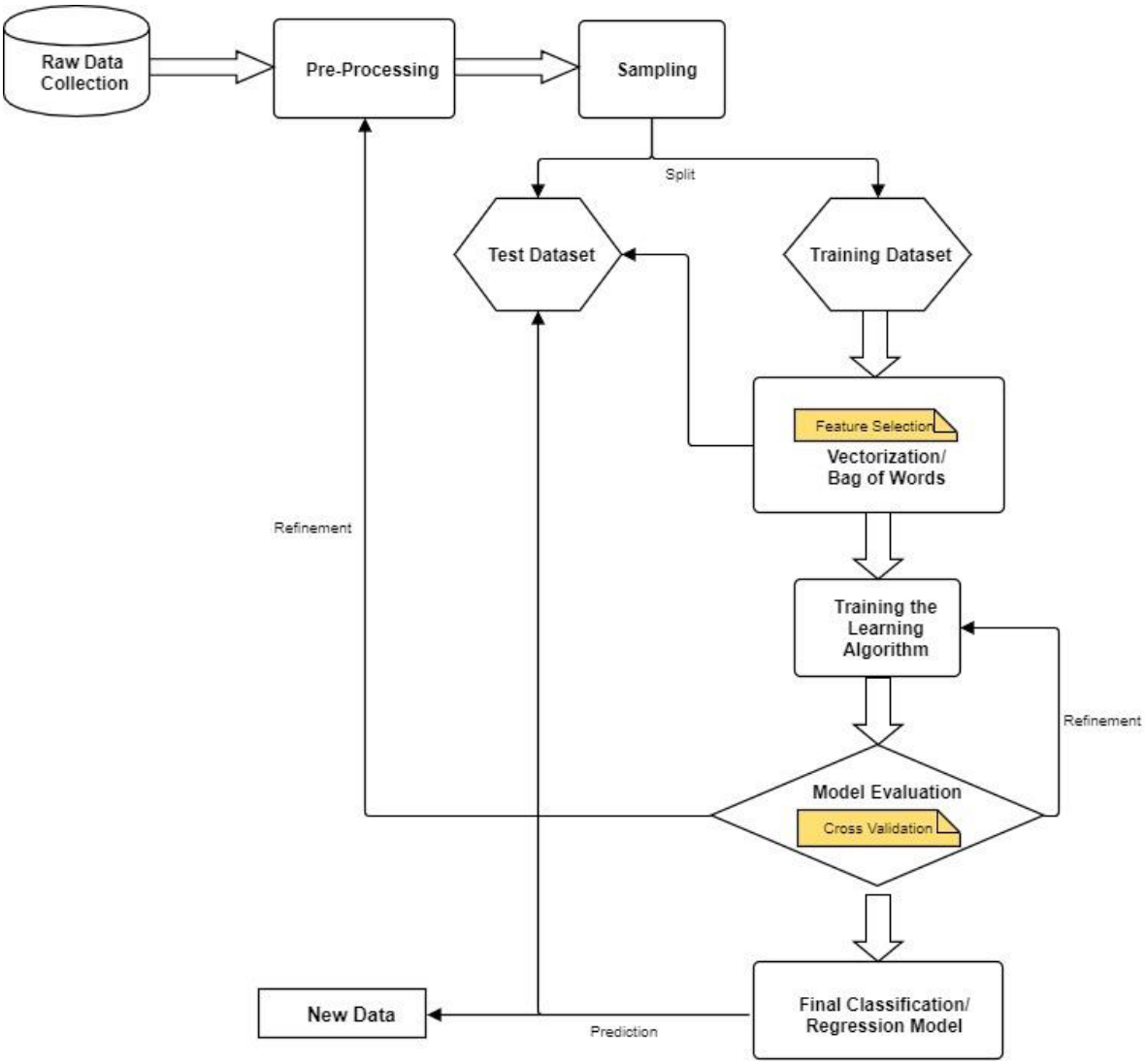


Figure 1 Architecture

Dataset

We are using a dataset of IMDB movie reviews consists of 233600 reviews. It was created by [1] for sentiment classification. The language is English. Although their primary use for this dataset was document classification, we have used it for reviews classification. Our tasks included the classification of reviews up to 5 classes. All of the reviews were complete paragraphs, where the average number of words per review were staggering 121.6, and upon an average of 5 characters per word, that totals up to 608 characters per review as compared to the 140-character limit of reviews.

Transformation

Our data consisted of a detailed review and its respective rating. All the reviews had a rating, ranging from 1 to 10. We decided to generate 3 models based on these ratings.

A Binary class model, where all the reviews with the rating of 1 to 5 are label as **Negative** and all the reviews with the rating of 6 to 10 are **Positive**.

A Tertiary class model, where all the reviews with the rating of 1 to 3 are **Negative**, all the reviews with the rating of 4 to 6 are **Neutral** and all the reviews with the rating of 7 to 10 are **Positive**.

A Five class model, where all the reviews with the rating of 1 to 2 are **Highly Negative**, all the reviews with the rating of 3 to 4 are **Negative**, all the reviews with the rating of 5 to 6 are **Neutral**, all the reviews with the rating of 7 to 8 are **Positive**, and all the reviews with the rating of 9 to 10 are **Highly Positive**.

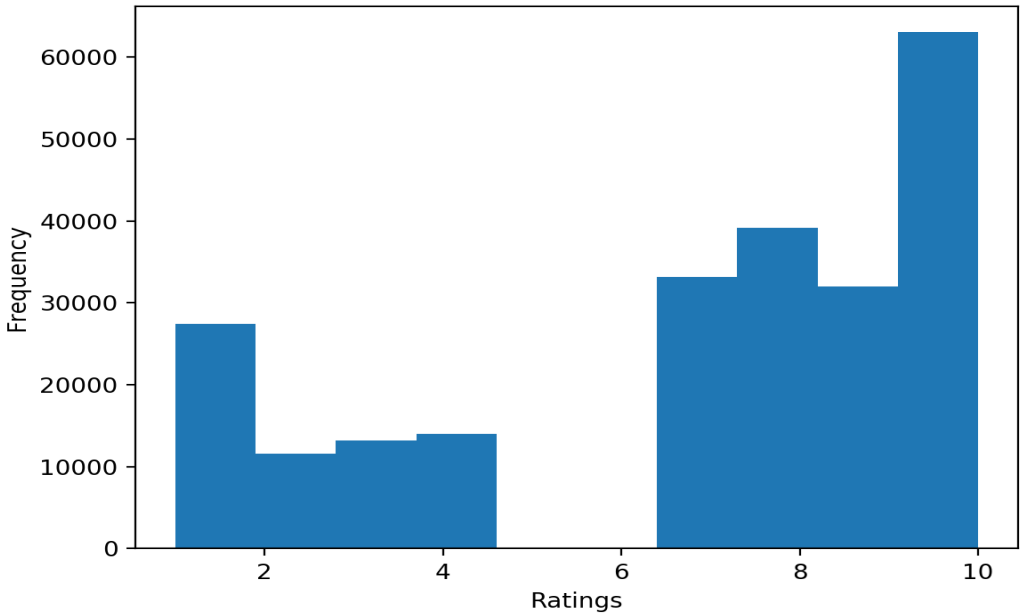


Figure 2 The score distribution of SAR 14

The dataset consists of highly natural reviews. Most of the reviews are in very informal language. Most of the reviews contain slangs, urls, numbers, punctuations, and are in need of much pre-processing to train effectively and get proper results.

" The first art-film ? . this is possibly the most beautiful of the early Lumiere shorts . A boat rows from the front of the screen away from the audience to the back , while to the right , women wait on the port . The camera , as usual , does not move , but the play of sunlight on the waves are gorgeous , and the rippling movement is so vibrant , especially within such a static frame , that it looks like one is watching an actual scene behind glass . As with ` Demolition d'un mur ' , the real frisson of the film comes from the unexpected . The scene proceeds as expected , the boat moving steadily along . But , just as it turns , a stronger wave lunges , and almost capsizes the boat . Before we discover what will happen , the reel , and the film , ends . These early films made no precautions for the necessity of extra reels . But the effect is quite shocking . The unexpected violence is unsettling enough , but with the film over , and loose ends nowhere near being tied up - indeed , just initiated a narrative , in the dying seconds - the audience is left agonising in the dark . What happened next ? Inadvertently , the audience is required to imagine for itself , imagine what 's not capable of being represented by the cinema , something the Hollywood generic system to come will stamp out . With its daring use of ellipsis , is this the first art-film , the first ` Cat People ' ? "

Table 1 Sample Review

Preprocessing

Preprocessing is an essential part any Text mining project. We wanted to clean up our dataset, remove all the meaningless text in the context of sentiment analysis. We considered and applied many different strategies which are as follows;

OpenRefine

OpenRefine is a tool used for cleaning and reformatting huge datasets. We made a copy of our main dataset and tried using OpenRefine to establish the effectiveness of this tool. We concluded that though its an elegant tool, its not for us. Following were the few problems we faced while using OpenRefine:

- Our dataset was too huge for performing any operation. When we tried stripping the spaces from the whole data set, even after 5, 6 hours, the processing continued.
- Our concern was mainly 'Review' column of dataset, which were all unique, whereas OpenRefine effectiveness lays in finding patterns to reduce the number of reviews.

Uniform Resource Locators (URLs)

Urls though important in some respects, were completely useless in the analysis of our text mining project, so we decided to get rid of them. We used regular expressions, to remove any block of text starting from http or https.

Special Characters

Special characters while maybe important for Author clustering or Style Breach detection, are not very useful during sentiment analysis. Although one might argue the important of a Dot(.), representing the start and end of a sentence but when it comes to sentiment analysis, the importance is nullified as it provides no context in semantics. After careful consideration we decided to remove special characters from our dataset. We used regular expression to remove all the special characters. Although there might be some cases where some characters may indicate even stronger emotion i.e. (I am angry vs I am angry!!!). Here adding exclamation marks may indicate even stronger emotion of anger. Similarly adding a question mark (?) could represent a state of strong emotion e.g. are you kidding me?????

Stop-words

Stop words are normally the most used words in any language. In the context of sentiment analysis, stop words are not very useful, so it is a standard practice to get rid of them during the preprocessing stage. Initially we used nltk.corpus to get the list of stop-words used in English language [2]. Even after filtering our dataset through these stop-words, we still had many words which provided no real meaning in sentiment analysis. We found a list of 5000-10000 most used words in English language. We used frequency distribution on our dataset to get the list of most common 200 used words. We compared both of these lists, added a few more words in our custom stop-word list. Later on, during the optimization phase, we changed this list into a Hashmap for faster preprocessing. To make sure that Hashmap worked properly, we converted all the words into their lowercase representation as it does not affect the sentiment analysis. We also observed lots of numbers, spam text that were of no use to our analysis. Here again, we used Regular expression to get rid of all the special characters and number to focus on the actual words for sentiment analysis.

Stop words
'over', 'doing', 'won', 'these', 'under', 'whom', 'ma', 'him', 'to', 'from', 'same', 'then', 'this', 'about, 'that', 'where', 'what, 'why', 'whom', 'is', 'am', 'are', 'for', 'has', 'both''

Table 2 List of few Stop words

Most Common words

In order to determine how classifiers are analyzing text, it is important to analyze the frequency distribution of your corpus. By the time we decided to add this step in our preprocessing phase we had already tried many other classifiers and noted down their results. We observed that most of the most used words were either incomplete, a slang variant the original word or a synonym. We

replaced all the slang used in the top 200 words to their original word i.e. 'movi' with 'movie', or 'movie' with 'film'. We did so, because semantically 'movi', 'movie' and 'film' should mean the same thing, but during classification the classifier will consider them as different features. So we tried to find the synonyms features and merged them for effective feature set.

Time	Distribution
Before	('film', 126108), ('movi', 102662), ('veri', 31922), ('charact', 27721), ('stori', 25591), ('onli', 23145)
After	('movie', 235378), ('very', 31922), ('character', 28280), ('story', 25591), ('only', 23145)

Table 3 Words distribution for 50000 reviews

POS Tagging

POS tagging is a great tool in lexical analysis. In sentiment analysis it can also play an important role if used properly. We observed that in terms of sentiment analysis, some of the tags were irrelevant and certain tags that were very important to the cause. E.g. Proper Nouns (NNP) are normally name of a person, place etc., although might be important for many text analysis applications were not important in our cases [3]. So we tried removing all the NNP from the dataset, while theoretically it seemed like an effective measure, in practical it was a lot more time consuming and yielded even lower results. We deduced that it might have been because, certain repetition in names of actors or names of places could have yielded good results in training set and hence had yielded better results. We also tried using only the adjective-variants and adverb-variants as these words describe the quality of something but then again it yielded ever lower results.

N-Gram

Sometimes its easier to deduce a context if more than 1 word is grouped and interpreted together. During our preprocessing phase we were mostly focusing on single word-based classification of data. After preliminary results with uni-gram, we decided to try bi-gram and tri-gram. We observed that in sentiment analysis 'bi-gram' makes a lot of sense e.g. '**bad'** could be interpreted as a negative sentiment but **'not bad'** changes the sentiment to **neutral**. Similarly, '**good'** implies a positive sentiment but depending on the word preceding it i.e. **'not good'** or **'very good'**, it conveys the sentiment of negative and highly positive respectively. Our accuracy scores varied greatly when we used different group settings. We observed that we got the best results when using the best features from uni-gram, bi-gram and tri-gram together. The second-best scores were when using uni-gram and bi-gram together. Using a uni-gram also yielded reliable results, but tri-gram was the most useless in our case, not only was it much more time consuming but also didn't affect the results in a positive way.

Lemmatization & Stemmer

Lemmatization & Stemmer are both used for the text processing operations that reduces different form of words to their root words. Lemmatization use more comprehensive lexical database but are generally slower than stemming. In our project we used nltk.stem.WordNetLemmatizer, nltk.stem.SnowballStemmer, nltk.stem.EnglishStemmer and nltk.stem.LancasterStemmer. There were slight changes in accuracy scores by using different Lemmatizers and Stemmers. We observed that we got the best accuracy scores while using the nltk.workNetLemmatizer [4].

Bag of words

In text processing usage of sequences of words is not feasible. For this reason, a dictionary of documents is represented by a matrix with one row per document and one column per token (e.g. word) occurring in the corpus. This technique is called the Bag of Words. We used the CountVectorizer and TFIDFVectorizer from sklearn for generating the corpus for the documents of our dataset. The vectorizers include utilities for the most common ways to extract numerical features from documents like tokenizing strings, counting the occurrences of tokens in each document, normalizing and weighting with diminishing importance tokens that occur in most documents [5].

CountVectorizer just counts the word frequencies of the document but with the TFIDFVectorizer the value increases proportionally to the count, but is balanced by the frequency of the word in the corpus; this is the IDF (inverse document frequency part). This counters the fact that some words appear more frequently than others. Without employing the inverse document frequency part; the common and less meaningful words such as “the” and “a” (considering stop words are not filtered) would yield a much higher weight than these less frequent words but more meaningful words.

Model training and validation

Model training involves providing an algorithm with data to learn and evolve from. The main objective of training the model is to gain experience from the training, build a generalized model and make better predictions when unseen data samples are given to it. For the task we have at hand, we have used Supervised learning, where the model is provided with sample data and the target labels/classes that are to be predicted when the unseen data is given to the model. We varied our validation and cross-validation parameters to check various scores i.e. Accuracy, Precision, Recall etc [6]. Below we have discussed some of the machine learning algorithms we used.

Multinomial NB

MultinomialNB implements the naive Bayes algorithm for multinomial distributed data, and is one of the two classic naive Bayes variants (MultinomoialNB and BernoulliNB) used in text classification.

Advantages

The Multinomial NB classifier is better to be used at larger vocabulary sizes. It is a more accurate classifier for dataset having large variance in document lengths. It easily handles documents of varying length by including the appearance of each word.
When working with binary features, as NB assumes feature independence, it drastically reduces the number of probabilities that need to be calculated, which then reduces the size of the training set required. One more advantage of Multinomial NB classifier is that even if one or more features are not available or missing, it is still possible to perform classification but the data or terms for the features that are not available will simply be removed/ ignored from the calculations.

Disadvantages

More consideration is needed when using Multinomial NB because if the non-text features are added to the vocabulary then the event spaces for the distinctive features would clash for the one and same probability mass, even though they are not mutually exclusive to each other.
Also, to consider is the problems that arise with the multinomial assumption in the context of document classification and different possible ways to remove or deal with those problems, including the use of ‘tf-idf’ weights; term frequency–inverse document frequency, instead of simple raw term frequencies and just document length normalization, to produce a naive Bayes classifier that is on par with SVMs (support vector machines).

Decision Tree Classifier

The decision tree classifier evaluates each attribute of the test record with a series of requirements that are carefully selected and each time it gets an answer a follow up check of another. The requirement is conducted until a conclusion for the class label of the record is reached.

Advantages

Decision tree classifiers are easy to understand. Particularly a naive binary decision tree is easy for anyone to code, visualize, manipulate, and explain which is not the case for more advanced classifiers which even though may be more accurate for larger datasets. It can handle both numerical and categorical data. It requires little data preparation (Data normalization). Also, Decision tree Classifiers make minimal assumptions. Decision Tree Classifiers mirror human behavior/ predictions more accurately than other approaches.

Disadvantages

As discussed before, the decision tree classifier does not tend to be as accurate as other approaches. It is not tamper proof as even a slight change in the training data may result in a substantial change in the final predictions.
It can create more complex than required trees that are not able to generalize well, from the training data.

Random Forest Classifier

The Random Forest Classifier, consists of many decision trees and gives out the class that is the mode of the classes outputted by the individual trees. The problem in decision trees of overfitting to their training set is also managed by Random Forest Classifier.

Advantages

Random Forest Classifiers lead up to less variance but more importantly it reduces overfitting of the training dataset, which is often enough to overlook and warrant the additional complexity.

Disadvantages

It is much harder to code, visualize, manipulate, and explain as it is more complex than other approaches hence it is also harder to implement. Thus, it is altogether computationally very expensive.

Logistic Regression

The dependent variable is categorical in a Logistic Regression i.e. the independent variable is used to predict the dependent variable.
It can be multinomial or binomial or even ordinal classifier. Multinomial logistic regression is used to deal with the situations where the outcome can have three or more possible types (e.g., "disease A" vs. "disease B" vs. "disease C") that are not ordered. Binomial logistic regression deals with the situations in which the outcome for a variable can only have two possible types, “0” and “1” e.g. “win” vs. “lose” or “alive” vs. “dead”. Ordinal logistic regression, as the name suggests, deals with dependent variables that are ordered.

Advantages

The independent variable IV and dependent variable DV are not needed to be normally distributed, when using a Logistic Regression. It can handle explicit interactions and may also handle non-linear effects. Logistic Regression does not require that the independents be interval or unbounded. It is very easy to update your model to take in new data in Logistic Regression.

Disadvantages:

Logistic Regression requires large sample size, to achieve stable results. It may also cause multicollinearity. In some cases, it may also overfit the data (Stepwise logistic regression).

One VS One Classifier

One VS One classifier creates one classifier for each pair of classes. At the time of prediction, the class with the most matched variable (The most voted class) is selected.
One VS One classifier provides more precise probability value, than the other classifiers that generate the probability value by means of regression. Although it may have errors or ambiguities, that is; some areas of its input space may get the same number of the matched variables (votes).

Other Techniques

Cross-validation, is a model validation technique used on an independent data set for measuring how the results of a statistical analysis will generalize on it. It is usually used for prediction, and for calculating how efficiently and accurately the predictive model will perform in actual practice. When dealing with a prediction problem, usually the model is given a dataset (of known data), and a dataset of unknown data (or first seen data). The training is run on the known data and the model is tested against the data set of unknown data. Cross validation defines a dataset (validation dataset) to test the model in the training phase, which deals with problems like overfitting and clues in as to how the model will generalize to an unknown dataset.

A single round of cross-validation involves dividing a sample of data into subsets, one of the subset is used as the training set, and the other is used as the validation set or testing set. To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are combined (averaged in our case) over the rounds to estimate a final model.

A big advantage of using cross-validation instead of normal validation is that the data available is not enough for partition it into separate training and test sets when using conventional validation without losing considerable modelling and testing capabilities. Hence, model prediction performance to be estimated fairly we use cross validation.

In summary, cross-validation combines measures of fit to extract a significantly more accurate estimate of the model prediction performance.

K-fold cross-validation, firstly randomly partitions the original sample into m same sized subsamples. Only one subsample is chosen as the validation data for testing the model from m subsamples. The remaining are used as training data. This entire process is repeated m times (a.k.a. the folds), where every time a different subsample is used as the validation data such that each subsample is used exactly once. The m results are then averaged to produce a single estimate. The advantage here is that the subsamples are used as both training and validation data and that each subsample is used as validation data exactly once, which is not the case for repeated random sub-sampling. m generally remains a n unfixed parameter but 10-fold cross-validation is commonly used.

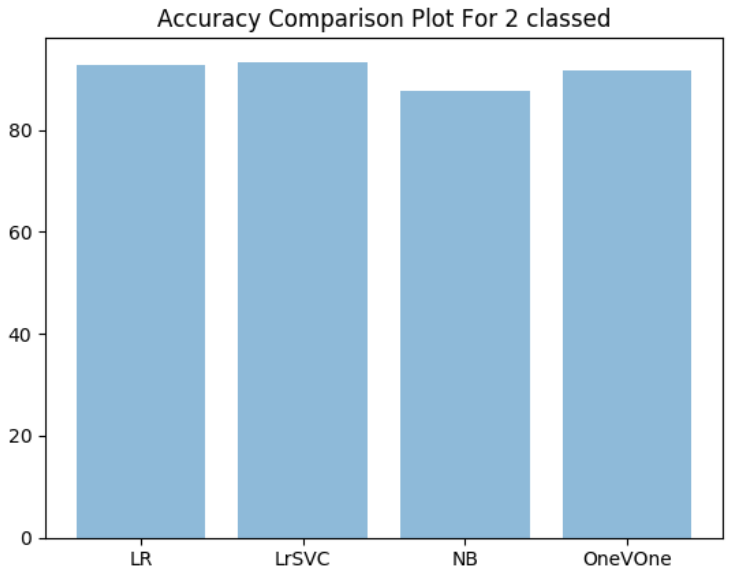
Discussion

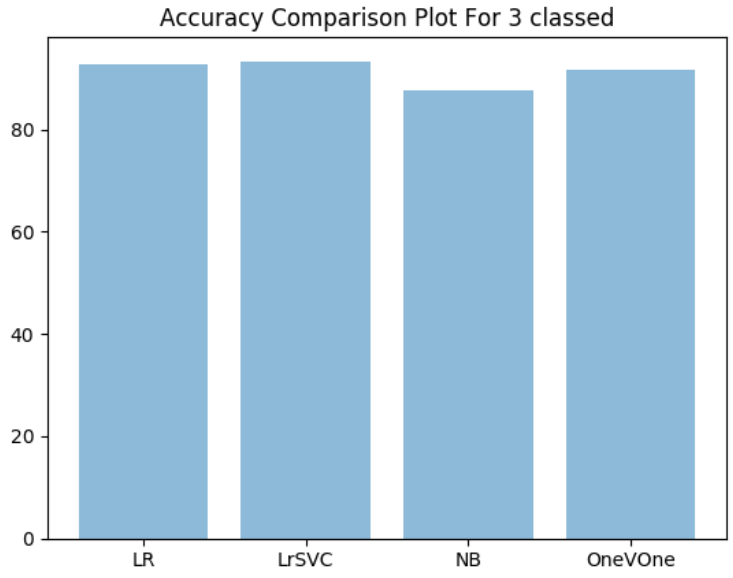
The challenge in the task started with finding the appropriate dataset for the challenge which could provide a real-world example of texts to be analyzed. We looked at quite a few datasets, most of them from the twitter corpus. The real challenge with most of them was that we couldn't decide as to how the labels (target variable) should be assigned to each of the documents in the dataset. The SAR14 dataset that we found had ratings in it which in a way depicted the sentiment of the commenter. We used the ratings provided to generate the labels for training the sentiment models. The only downside was that the class didn't have ratings in the 5-6 range. This affected the neutral target variable with it ending up with 0 documents. We decided to keep the data in its original form to provide our models with the best possible real-world data. The preprocessing part was the most computationally expensive part of the entire process where we used many different techniques

which at first, we believed would have a positive effect on the effectiveness of the corpus being provided to the model for training but proved otherwise. Some of these techniques were stemming, POS tagging (removal of Proper Nouns i.e. NNPs) to limit the corpus to word forms like Adjectives i.e. JJs so that the sentiment of the sentence could be effectively predicted. There were two downsides to these techniques. The first one being that it increased the execution time a lot and the second most important being that it caused reduction in the corpus size. Sometimes we tried by excluding all the non-essential POS tags and sometimes we tried using only the essential POS tags, in all of the cases there were two major downsides to these techniques. The first one being that it had a very time complexity and the second and most important being that it caused reduction in the corpus size. We also wanted to make sure that the words in the feature set are meaningful to sentiment. We computed the frequency distribution of our corpus and got the top 200 words in our corpus. We filtered all the words that we thought were useless in sentiment analysis**.** We also merged the words that were synonyms to each other's as to maximize the feature set i.e. Films and movies are referring to same word so it does not make any sense to have two feature set for it.

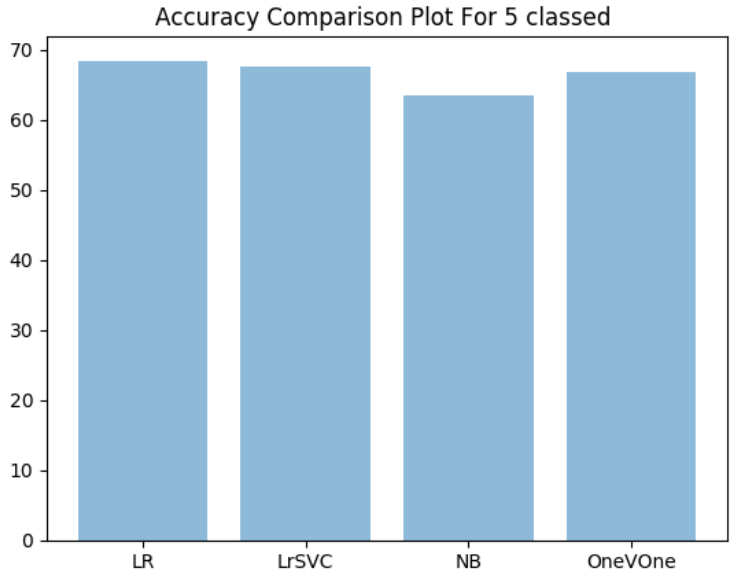
The preprocessing techniques that we found useful were removing URLs, removing special characters, removing numbers and lemmatization. To deal with the time complexity of the preprocessing we worked with dataframes and transformed columns using the apply method. It proved to have a better execution time than some of the other methods like looping through each row and then applying the preprocessing methods. However, the difference between the methods wasn't too big. We used the CountVectorizer and TfidfVectorizer to generate the term document matrix or better known as Bag of Words to train our model. The biggest challenge we faced was tuning our models. Tuning the models takes up a lot of time and processing power which we lacked. The general idea to tune a model, that we had planned to follow, was that we would use the GridSearchCV provided by sklearn to tune the parameters of the models we used and plot the learning curve to find out the best set of parameters which would give better predictions. Training an algorithm at times leads to overfitting the model. In order prevent overfitting we used k-fold cross validation.

Out of all the models we used, Logistic Regression (aka logit, MaxEnt) classifier performed the best in the five-class target problem whereas LinearSVC performed the best in binary and tertiary classification. We ran the model with unigrams, bigrams and trigrams, out of which we thought bigrams were more relevant. For unigrams with LogisticRegression and a 10-fold cross validation we got a mean accuracy of **67.19%**. For bigrams, with the same configuration we got a mean accuracy of **68.51** and with trigrams we got **68.50%.** LogisticRegression has a probabilistic approach and assumes that a decision boundary exists between the different classes. It doesn't assume a linear relationship between the variables, independent and dependent both. The variables don't need to have normal distribution. The classifier also allows us to understand the impact of IV on DV while controlling for other IVs. For Subtask A, where we were supposed to predict the sentiment from the tertiary target classes ranging from positive, neutral and negative we observed a max accuracy of **93.74%** and a mean accuracy of **93.43%**. For Subtask B, where we were supposed to predict the sentiment from the binary classes i.e. negative and positive, we observed an almost similar max accuracy of **93.74%** and mean accuracy of **93.43%**. For Subtask C where the target labels ranged from highly positive to highly negative, we observed a max accuracy of **69.1%** and a mean accuracy of **68.51%**. A more detailed view of the results we observed across the different algorithms along with different combination of feature extraction and preprocessing techniques has been shown in the figure below.

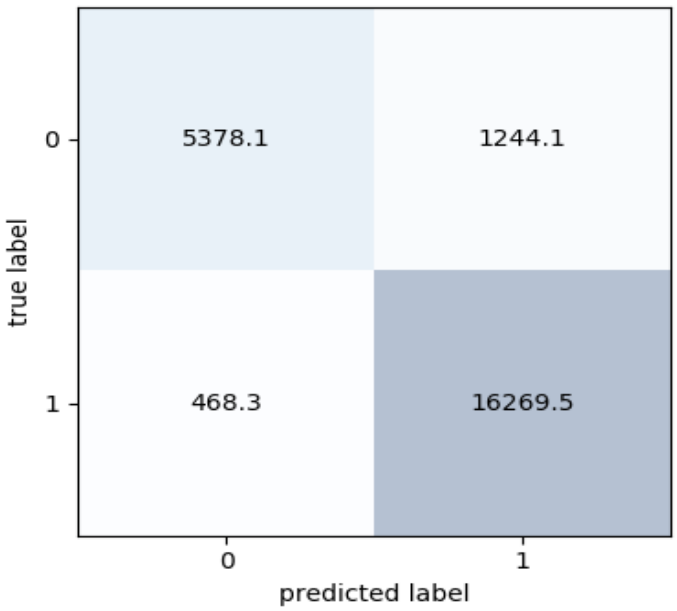




LR	Logistic Regression
LrSVC	Linear SVC
NB	Naïve Bayes
OneVOne	One Vs One (With Linear SVC)

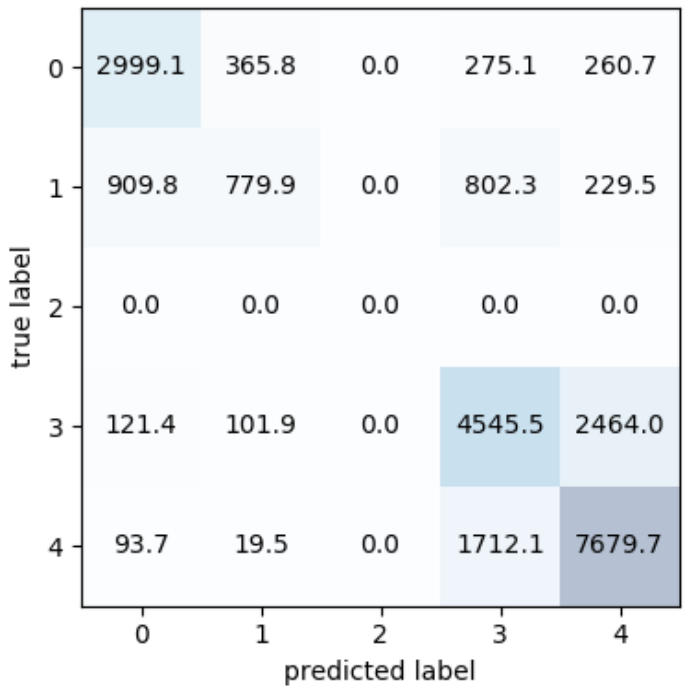


For the Subtask D, where we had to estimate the distribution of the reviews in the POSITIVE and NEGATIVE classes, the classifier made a total of 23360 predictions on an average. Out of these 17513.6 were predicted positive, while only 16737.8 were positive and 5846.4 were predicted negative while 6622.2 were negative.



0	Negative
1	Positive

For Subtask E, where we were supposed to estimate the distribution of the reviews across the five classes of a five-point scale, ranging from HIGHLYNEGATIVE to HIGHLYPOSITIVE, the classifier made a total of 23360 predictions on an average. Out of these 4124 were predicted HIGHLY NEGATIVE but only 3900.7 were in fact HIGHLY NEGATIVE. For the NEGATIVE class it predicted the average distribution as 1267.1 but in reality, 2721.5 were NEGATIVE. For the POSITIVE class the average distribution predicted was 7335 but only 7232.8 were POSITIVE. For the HIGHLY POSITIVE, distribution predicted was 10633.9, while in fact it was 9505. The diagonal in both the confusion matrices shows the correct number of predictions made.



0	Highly Negative
1	Negative
2	Neutral
3	Positive
4	Highly Positive

In the end, we have summarized a list of experiments with varying parameters and their respective accuracy results in the following Table4 and Table5.

Pre-Processing Used	Vectorizer	Classifier	N-Gram Used	Accuracy (%age)
• Stemming • URL Removal • Special Character Removal	Count Vectorizer	OneVsOneClassifier, estimator=LinearSCV	(1, 2)	64.7
• Stemming • URL Removal • Special Character Removal	Count Vectorizer	LogisticRegression, featureset=50000	(1, 1)	65
• Stemming • URL Removal • Special Character Removal	Count Vectorizer	NaiveBaise, featureset=20000	(1, 2)	63
• Stemming • URL Removal • Special Character Removal	Count Vectorizer	LogisticRegression, featureset=50000	(2, 2)	61
• Stemming • URL Removal • Special Character Removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, default min_df, max_df, solver=sag	(1, 2)	66.3
• Stemming • URL Removal • Special Character Removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 5, max_df = 0.8, solver=sag	(1, 2)	67.69
• Stemming • URL Removal • Special Character Removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 5, max_df = 0.8, solver=sag	(2, 2)	64.20
• Stemming • URL Removal • Special Character Removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 25, max_df = default, solver=sag	(1, 2)	67.01
• Stemming • URL Removal • Special Character Removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 100, max_df = default, solver=sag	(1, 2)	66.66
• Stemming • URL Removal • Special Character Removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 25, max_df = 0.8, solver=newton-cg	(2, 2)	67.74
• Stemming • URL Removal • Special Character Removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 25, max_df = 0.8, solver=newton-cg	(1, 2)	64.2
• Stemming • URL Removal • Special Character Removal • Extra words in the stop-words filter- Ignoring all the proper nouns	TFIDF Vectorizer	LogisticRegression, min_df = 25, max_df = 0.8, solver=newton-cg	(1, 2)	66.95

• Lemmatizing • URL removal • Special character removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 25, max_df = 0.8, solver=newton-cg	(1, 2)	67.92
• Wordnet Lemmatizing • URL removal • Special character removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 25, max_df = 0.8, solver=newton-cg	(1, 3)	68.01
• Wordnet Lemmatizing • URL removal • Special character removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 25, max_df = 0.8, solver=newton-cg	(1, 3)	61.77
• Wordnet Lemmatizing • URL removal • Special character removal • Extra words in the stop-words filter	TFIDF Vectorizer	LogisticRegression, min_df = 25, max_df = 0.8, solver=newton-cg	(1, 3)	61.77

Table 4 Results with varying Parameters

Classifier	Feature Set	Accuracy (%age)	Precision (%age)	Recall (%age)
OneVsOneClassifier(LinearSVC())	4000	67.14	63.98	62.71
OneVsOneClassifier(LinearSVC())	4500	67.003	63.83	62.61
OneVsOneClassifier(LinearSVC())	5000	67.09	63.92	62.71
OneVsOneClassifier(LinearSVC())	5500	67.10	63.84	62.70
LinearSVC	4500	66.64	63.52	60.75
LinearSVC	5000	66.50	63.72	60.46
LogisticRegression	4000	66.72	63.95	60.58
LogisticRegression	4000 with Saga Solver	67.06	64.25	61.18
LogisticRegression	4000 with Saga Solver & TFIDF Vectorizer	67.27	65.28	61.13
LogisticRegression	4500	66.62	63.79	60.66
LogisticRegression	3500	66.51	63.74	60.30
OneVsOneClassifier	4000 with 10-fold CV	66.91		
LogisticRegression	4000 with 10-fold CV & TFIDF Vectorizer	67.24		

Table 5 Classifiers with varying Feature set

The source code has been uploaded on Gitlab: *https://gitlab.com/nawab.hussaen/SentimentAnalysis* for reference.

Conclusion

Just as the name suggests, we started off Naively with the Naïve Bayes based Classifiers. We used simple pre-processing techniques with Count Vectorizers to form a Bag of words most suitable for sentiment analysis. We soon realized that it involved a lot more research and pre-processing techniques than just removing URLs and special characters. We tried many advance classifiers i.e. Logistic Regression, Random forest etc. to see the bias-variance trade-off. We applied parameters and cross validation techniques to avoid under-fitting and over-fitting and to see the variation in our scores. We measured our results with Accuracy, Precision and Recall. We computed Confusion matrices to show our results. We believe we did justice to our 5-Tasks challenge mentioned in SemEval-2016 Task 4: Sentiment Analysis in Twitter. We experimented with different classifiers by varying different parameters to get the best possible results for our dataset. Our conclusive results as mentioned in the discussion are a lot better than our initial results and given more time we could apply different techniques i.e. GridSearchCV, deep learning models to enhance the results even more.

Bibliography

[1] D. Q. Nguyen, D. Q. Nguyen, T. Vu and S. B. Pham, " Sentiment classification on polarity reviews: an empirical study using rating-based features," in *Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Hanoi, 2014.

[2] S. Bird, E. Klein and E. Loper, "Accessing Text Corpora and Lexical Resources," 15 1 2015. [Online]. Available: http://www.nltk.org/book/ch02.html.

[3] U. o. Pennsylvania, "Alphabetical list of part-of-speech tags used in the Penn Treebank Project," [Online]. Available: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

[4] TextMiner, "Dive Into NLTK, Part IV: Stemming and Lemmatization," 18 07 2014. [Online]. Available: http://textminingonline.com/dive-into-nltk-part-iv-stemming-and-lemmatization. [Accessed 9 09 2017].

[5] "Bag of Words & TF-IDF," [Online]. Available: https://deeplearning4j.org/bagofwords-tf-idf. [Accessed 2 10 2017].

[6] S.-l. developers, "Documentation for sklearn.metrics.precision_recall_fscore_support," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html. [Accessed 12 10 2017].