Sentiment Analysis

Mayank Mahavar
National Institute of Technology,
Surathkal, Karnataka,India
Email :- mayankmahavar111@gmail.com

Dhruv Agarwal
Manipal Institute of Technology,
Manipal, Karnataka, India
Email :- dhruvagr@gmail.com

1. Introduction

In recent years, many people have access to the Internet due to enhancements in technology. Thus people have been expressing their opinion on social platforms. The rise of social media such as blogs and social networks has fueled interest in sentiment analysis. With the proliferation of reviews, ratings, recommendations and other forms of online expression, online opinion has turned into a kind of virtual currency for businesses looking to market their products, identify new opportunities and manage their reputations.
Sentiment Analysis, also known as opinion mining, is the process of determining whether a given sentence is positive, negative or neutral. Sentiment analysis aims to determine the attitude of a speaker, writer, or other subject with respect to some topic. A common use case for this technology is to discover how people feel about a particular topic.

The sentiment found within comments, feedback or critiques provide useful indicators for many different purposes. These sentiments can be categorized either into two categories: positive and negative; or into an n-point scale, e.g., very good, good, satisfactory, bad, very bad. In this respect, a sentiment analysis task can be interpreted as a classification task where each category represents a sentiment. Sentiment analysis provides companies with a means to estimate the extent of product acceptance and to determine strategies to improve product quality. It also facilitates policy makers or politicians to analyze public sentiments with respect to policies, public services or political issues.

There are various applications of Sentiment Analysis. Some of them have been listed below-
1. Customer Survey about certain products and services
2. Tone recognition for brands
3. Evaluation of voters during elections
4. Corporate announcements and real time opinion streaming
5. News analysis for stock selection portfolio
6. Predict past, present and future reputation
7. Analyzing conference calls

The Obama administration used sentiment analysis to gauge public opinion to policy announcements and campaign messages ahead of 2012 presidential election.
The ability to quickly understand consumer attitudes and react accordingly is something that Expedia Canada took advantage of when they noticed that there was a steady increase in negative feedback to the music used in one of their television adverts.

In the following document Sentiment analysis is done using three algorithms-

1. Naive Bayes'
2. Maximum Entropy
3.Support Vector Machine

2. Naive Bayes'

Naive Bayes' classifiers are linear classifiers that are known for being simple yet very efficient. The probabilistic model of Naive Bayes' classifiers is based on Bayes' theorem, and the adjective *naive* comes from the assumption that the features in a dataset are mutually independent.
In practice, the independence assumption is often violated, but naive Bayes classifiers still tend to perform very well under this unrealistic assumption. Especially for small sample sizes, Naive Bayes' classifiers can outperform the more powerful alternatives.

Multi-variate Bernoulli Naive Bayes' :
This is a binary model. It counts whether the word has occurred or not. Multiple occurrence of the words do not make a difference to this model.

Multinomial Naive Bayes':
This model counts multiple occurrences of a word. In this model words which occur frequently have higher weightage than the words which occur less frequently.

In the following document, Multinomial Naive Bayes' is used.

2.1 Preprocessing of Text

In order to run Sentiment Analysis on a sentence, the sentence is preprocessed using the following ways-

Named Entity Recognition (NER)
All words in the test file are tested for NER. The words which do not belong to any NER category (Name, location, organization, time, date, money) are added in the ArrayList. Stanford NLPCore's NER API is used for implementing the above. If the word belongs to any of the NER's then it is not taken into account for further processing as NER terms do not contribute to the sentiment of a sentence.

Stopwords
Stopwords is a file containing words like (is, or, that) which have no significance in analyzing the sentiment of the sentence. When the input file is read, the stopwords are not added in the ArrayList.

Lemmatization
Using the Every word present is converted into its dictionary form using Lemma. Ex.working is converted into work. Lemma is used so that all words are present in their base forms and hence the occurance of each word in various forms is counted. This is applied using Stanford NLPCore's API

Tokenizer
Divides sentences into words. Thus the whole document is converted into a bag of words where each word is independent of each other. This is also a part of Stanford NLPCore's feature .

Abbreviation
Terms which are used in short forms are converted to their full forms. Ex. ttyl is converted to talk to you later. There are about 70 such abbreviations which are commonly used. These replacements are hard coded.

Order of Preprocessing
Tokenizer → Lemmatization → Stopwords removal → NER → Abbreviation Replacement

Training File
According to the application for which sentiment analysis is going to be used, obtain training data regarding that topic. For example if we want to do sentiment analysis on cricket then the training data should be about cricket. Training data is set of sentences regarding the matter. Two sets of sentences will be formed, one for positive and one for negative. These sentences are preprocessed according to the order mentioned. Hence two training files are formed with a list of words in it.

2.2 Derivation

Step 1:
Bayes' Theorem gives us the following formula-
P(A|B) = P(B|A).P(A) / P(B)

P(A) and P(B) are the independent probabilities of A and B respectively.
P(A|B), a conditional probability, is the probability of event A occurring provided event B is true(event B has already occurred).
P(B|A) is the probability of event B provided event A has occurred (true).

Step 2:
Using this formula for sentiment analysis-

P(Positive | Sentence) = P(Sentence | Positive).P(Positive) / P(Sentence)

Since P(Sentence) has the same value for positive as well as negative class, we can drop the term in order to simplify the equation. Thus the equation becomes
P(Positive | Sentence) = P(Sentence | Positive).P(Positive)

Step 3:
A sentence consists of words. Thus
P(Sentence | Positive) = P(word1, word2, …....wordn | Positive)

According to the assumption that all words are independent of each other
P(Sentence | Positive) = P(word1 | Positive).P(word2 | Positive). …... .P(wordn | Positive)
P(Sentence | Positive) =$\prod_{i \in n} P\ (wordi|Positive)$(Fraction of times wordi appears in all positive sentences)
P(Sentence | Positive) =$\prod_{i \in n} \frac{count(wordi,positive)}{count(totalWords,positive)}$(The totalWords are the total unique words)

P(Positive) = $\frac{count(totalWords,positive)}{count(totalWords,positive)+count(totalWords,negative)}$

Thus the total equation becomes
P(Positive | Sentence) = $\frac{count(totalWords,positive)}{count(totalWords,positive)+count(totalWords,negative)}.\prod_{i \in n} \frac{count(wordi,positive)}{count(totalWords,positive)}$

P(Positive | Sentence) is actually not probability(will not be between 0 & 1). It is simply a value which is calculated for both P(Positive | Sentence) and P(Negative | Sentence). The two values are compared. Suppose the value of P(Negative | Sentence) is higher than P(Positive | Sentence) then the sentence's sentiment is negative.

Step 4:

In the formula above the second term contributes a very small value being an important term for sentiment analysis. It is not comparable to the first term and hence to make both the terms comparable logarithm is applied to both the terms.

P(Positive | Sentence) =

$$\log\frac{count(totalWords,positive)}{count(totalWords,positive)+count(totalWords,negative)}.\log\prod_{i\in n}\frac{count(wordi,positive)}{count(totalWords,positive)}$$

Step 5:

Laplace Smoothening

If a word occurs which is not present in training file ,its count is zero and hence the numerator becomes zero making the term equal to log(0) which is not valid. Addition of +1 in the numerator avoids the value inside the logarithm to become zero.

The Final equation is

P(Positive | Sentence) =

$$\log\frac{count(totalWords,positive)}{count(totalWords,positive)+count(totalWords,negative)}+\sum_{i\in n}log\frac{count(wordi,positive)+1}{count(totalWords,positive)}$$

Total Positive Probability = log[(number of words in positive file)/(number of words in positive and negative file)] + Σ log[(occurrence of word[i] in positive file + 1)/(number of words in positive file)]

P(Negative | Sentence) =

$$\log\frac{count(totalWords,negative)}{count(totalWords,positive)+count(totalWords,negative)}+\sum_{i\in n}log\frac{count(wordi,negative)+1}{count(totalWords,negative)}$$

Total Negative Probability = log[(number of words in negative file)/(number of words in positive and negative file)] + Σ log[(occurrence of word[i] in negative file + 1)/(number of words in negative file)]

2.3 Example

Let us take an example. “ Shyam is a good boy”
We have trained our training file.
By preprocessing the sentence ,
'Shyam' is recognised by the NER as a name and hence removed.
'is','a' are in the stopwords list and hence removed.
So the word 'good' and 'boy' will contribute to the sentiment of the sentence.
Suppose there are 600 words in the positive file and 400 words in the negative file. The word 'good' occurs 20 times in the positive file and 10 times in negative file. Similarly 'boy' occurs 10 and 12 times respectively.

P(Positive | 'good','boy') = log(600/(600+400)) + (log((20 + 1)/600) + log((10 + 1)/600)) =

P(Negative | 'good','boy') = log(400/(600+400)) + (log((10 + 1)/600) + log((12 + 1)/600)) =

The value of positive probability is more than the negative probability. Thus, the sentiment of the sentence is positive.

2.4 Tweaks

Another example is “Shyam is not a good boy”.
The processed words are 'not' , 'good' and 'boy'.
In this case the meaning of good has changed because of the word not.
So whenever the word 'not' appears, the probability of the word after it is taken in the opposite file.
Also the word is not counted for the purpose of probability.
For the example “Shyam is not a good boy”.

P(Positive | 'good','boy') = log(600/(600+400)) + (log((10 + 1)/600) + log((10 + 1)/600)) =
For the word 'good' for positive probability, its count in the negative file(10) is taken into formula.

P(Negative | 'good','boy') = log(400/(600+400)) + (log((20 + 1)/600) + log((12 + 1)/600)) =

Since the value of negative probability is more than the positive probability, the sentiment of the sentence is negative. Words like doesn't are broken by the lemma into doesn and t. The word doesn is removed as it belongs to the stop words file. 't' is considered as an abbreviation and converted into 'not'.

3. Maximum Entropy

Maximum entropy modeling is a framework for integrating information from many heterogeneous information sources for classification. The data for a classification problem is described as a (potentially large) number of features. These features can be quite complex and allow the experimenter to make use of prior knowledge about what types of information are expected to be important for classification. Each feature corresponds to a constraint on the model. We then compute the maximum entropy model, the model with the maximum entropy of all the models that satisfy the constraints. If we chose a model with less entropy, we would add `information’ constraints to the model that are not justified by the empirical evidence available to us. Choosing the maximum entropy model is motivated by the desire to preserve as much uncertainty as possible. On the engineering level, an added benefit is that the person creating a Maxent model only needs to inform the training procedure of the event space, and need not worry about independence between features.

Goal

Our goal is to calculate p where p is value, variable of class with maximum entopy.

$$H(p) = -\sum_{x \in A \times B} p(x) \log p(x)$$

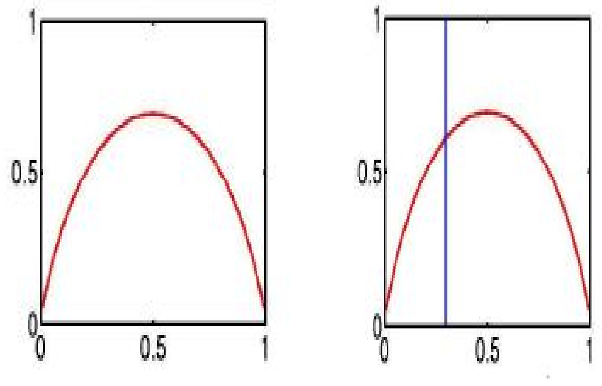
$$x = (a, b), \quad where \quad a \in A \wedge b \in B$$

From training data , collect(a , b) pairs:

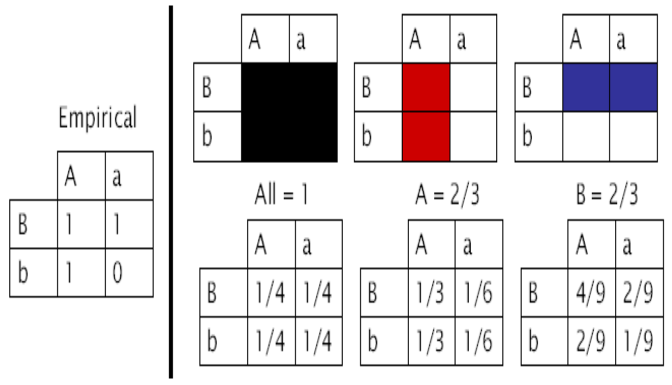
- a : things to be predicted (e.g. , a class in a classification problem)
- b : the context
- Ex : POS tagging:
  - a=NN
  - b=the words in a window and previous two tags

3.1 Coin-flip example

- Toss a coin: p(H)=p1, p(T)=p2.
- Constraint: p1 + p2 = 1
- Question: what's your estimation of p=(p1, p2)?
- Answer: choose the p that maximizes H(p)



3.2 Overlapping Feature



Empirical

	A	a
B	1	1
b	1	0

	A	a
B	⬛	⬛
b	⬛	⬛

All = 1

	A	a
B	1/4	1/4
b	1/4	1/4

	A	a
B	🟥	
b	🟥	

A = 2/3

	A	a
B	1/3	1/6
b	1/3	1/6

	A	a
B	🟦	🟦
b		

B = 2/3

	A	a
B	4/9	2/9
b	2/9	1/9

3.3 Feature

Feature (a.k.a. feature function, Indicator function) is a binary-valued function on events:

$$f_j : \varepsilon \rightarrow \{0,1\}, \quad \varepsilon = A \times B$$

A: the set of possible classes (e.g., tags in POS tagging)
B: space of contexts (e.g., neighboring words/ tags in POS tagging)
Ex:

$$f_j(a,b) = \begin{cases} 1 & if \ a = DET \,\&\, curWord(b) = "that" \\ 0 & o.w. \end{cases}$$

Finite training sample of events: S

Observed probability of x in S: $\tilde{p}(x)$

The model p's probability of x: $p(x)$

The jth feature: f_j

Observed expectation of : $E_{\tilde{p}} f_j = \sum_{x \in \varepsilon} \tilde{p}(x) f_j(x)$

Model expectation of : $E_p f_j = \sum_{x \in \varepsilon} p(x) f_j(x)$

3.4 Constraints

Model's feature expectation = observed feature expectation

Where, $E_p f_j = E_{\tilde{p}} f_j$

$$E_{\tilde{p}} f_j = \sum_{x \in \varepsilon} \tilde{p}(x) f_j(x) = \frac{\sum_{i=1}^{N} f_j(x)}{N}$$

$$f_j(a,b) = \begin{cases} 1 & if \ a = DET \,\&\, curWord(b) = "that" \\ 0 & o.w. \end{cases}$$

3.5 Lagrangian Multipliers

Minimize A(p): $A(p) = -H(p) - \sum_{j=0}^{k} \lambda_j (E_p f_j - d_j)$

$$A'(p) = 0$$

$$\Rightarrow \delta(\sum_{x} p(x) \log p(x) - \sum_{j=0}^{k} \lambda_j ((\sum_{x} p(x) f_j(x)) - d_j) / \delta p(x) = 0$$

$$\Rightarrow 1 + \log p(x) - \sum_{j=0}^{k} \lambda_j f_j(x) = 0$$

$$\Rightarrow \log p(x) = \sum_{j=0}^{k} \lambda_j f_j(x) - 1$$

$$\Rightarrow p(x) = e^{\sum_{j=0}^{k} \lambda_j f_j(x) - 1} = e^{\sum_{j=1}^{k} \lambda_j f_j(x) + \lambda_0 - 1}$$

$$\Rightarrow p(x) = \frac{e^{\sum_{j=1}^{k} \lambda_j f_j(x)}}{Z} \quad where\ Z = e^{1-\lambda_0}$$

$$p(x) = \frac{e^{\sum_{j=1}^{k} \lambda_j f_j(x)}}{Z}$$

$$\pi = \frac{1}{Z} \quad \lambda_j = \ln \alpha_j$$

For calculating lambda two types of algorithm's are used-

3.6 GIS

3.6.1Requirements for running GIS:

Obey form of model and constraints:

$$p(x) = \frac{e^{\sum_{j=1}^{k} \lambda_j f_j(x)}}{Z} \qquad E_p f_j = d_j$$

An additional constraint:

$$C = \max_{x \in \varepsilon} \sum_{j=1}^{k} f_j(x) \qquad \forall x \in \varepsilon \quad \sum_{j=1}^{k} f_j(x) = C$$

Add a new feature f_{k+1}:

```
Step1: Compute $d_j$, j=1, …, k+1
Step2: Initialize $\lambda_j^{(1)}$ (any values, e.g., 0)
Step 3: Repeat until converge
Step 4:         For each j
Step 5:                 Compute $E_{p^{(n)}} f_j = \sum_{x \in \varepsilon} p^{(n)}(x) f_j(x)$
                        Where $p^{(n)}(x) = \frac{e^{\sum_{j=1}^{k+1} \lambda_j^{(n)} f_j(x)}}{Z}$
Step 6:                 Update $\lambda_j^{(n+1)} = \lambda_j^{(n)} + \frac{1}{C}(\log \frac{d_i}{E_{p^{(n)}} f_j})$
```

3.6.2 Approximation for calculating feature expectation

$$E_p f_j = \sum_{x \in \varepsilon} p(x) f_j(x) = \sum_{a \in A, b \in B} p(a,b) f_j(a,b)$$

$$= \sum_{a \in A, b \in B} p(b) p(a \mid b) f_j(a,b)$$

$$\approx \sum_{a \in A, b \in B} \tilde{p}(b) p(a \mid b) f_j(a,b)$$

$$= \sum_{b \in B} \tilde{p}(b) \sum_{a \in A} p(a \mid b) f_j(a,b)$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{a \in A} p(a \mid b_i) f_j(a, b_i)$$

3.6.3 Properties of GIS Algorithm

- $L(p^{(n+1)}) >= L(p^{(n)})$
- The sequence is guaranteed to converge to p*.
- The converge can be very slow.
- The running time of each iteration is O(NPA):

N: the training set size
P: the number of classes

A: the average number of features that are active for a given event (a, b).

3.7 IIS

3.7.1 Algorithm

```
Step 1: Compute $d_j$, j=1, …, k+1 and

Step 2 : Initialize $\lambda_j^{(1)}$ (any values, e.g., 0)
Step 3 : Repeat until converge
Step 4:        For each j
Step 5:        Let $\Delta\lambda_j$ be the solution to

Step 6:             Update $\lambda_j^{(n+1)} = \lambda_j^{(n)} + \Delta\lambda_j$
```

$$\forall x \in \varepsilon \quad \sum_{j=1}^{k} f_j(x) = C$$

$$\Delta\lambda_j = \frac{1}{C}(\log\frac{d_i}{E_{p^{(n)}} f_j})$$

3.8 Maximum Entropy Implementation

Given Implementation of Maximum Entropy is done by using StanfordNLP API.
Implementation of maxent is as follows:

getMaxEntropySentiment()

Step 1 : ColumnDataClassifier cdc = **new** ColumnDataClassifier(<propertie path>);

Step 2 : Classifier cl = **null**;

Step 3 : cl = IOUtils.*readObjec FromFile*("<Trained file path>");

Step 4: Datum<String, String> d = cdc.makeDatumFromLine(<input String>);

Step 5: **if** (cl.classOf(d).equals("0"))
<To do Task For negative Sentiment>

```
Step 6:   else
          <To do Task For negative Sentiment>
```

3.9 Explanation

Step 1:
Classifier is created with required properties. Since API contains so many properties it is necessary to tell what is needed for the process. Otherwise it will load all the properties and make the program slow.

Step 2:
A null classifier is made for the input data. With the help of this it is possible to compare the given input with the trained classifier.

Step 3:
Classifier is defined which is declared in previous Step.

Step 4:
Given input is converted into datum features.

Step 5:
If the sentiment of the given output is "0" , task for the negative sentiment is have to be done.

Step 6:
Else task for positive sentiment is done.

4. Support Vector Machine

SVM(Support Vector Machine) have been proven as one of the most powerful learning algorithms for text categorization. SVM is a useful technique for data classification. Although SVM is considered easier to use than Neural Networks, users not familiar with it often get unsatisfactory results at first. A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one target value" (i.e. the class labels) and several attributes" (i.e. the features or observed variables). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes.

4.1 Preprocessing of text

The input data is initially processed and then Svm classifier is applied to it.
Different features techniques like unigrams, bigrams, unigrams + bigrams, unigrams + POS tagging, Position and unigrams + Position. Tokenization, stop word removal, TF-IDF and POS tagging is used as part of pre-processing. Also new approach sentiment fuzzy classification algorithm is used to improve result. Movie and Twitter review are classified using natural language techniques which are synonym using WordNet and word sense disambiguation. The results show that accuracy is increased by 5% using machine learning ensemble classifiers consist from Random Forest, Decision Tree, Extremely Randomized Trees and Ada Boost. WordNet synset increases accuracy. Machine learning approach is supervised learning approach because classifier is trained on dataset whereas semantic approach is unsupervised because it measures how far a word is related to positive or negative. Both approaches have its pros and cons. In this paper supervised machine learning approach is more accurate then semantic orientation but it is time consuming process to train model. Semantic orientation approach is less accurate but it is efficient. Tweet sentiment analysis model consists of feature selection

module, sentiment identification module and sentiment aggregation & scoring module. In feature extraction, they are extracted opinion words using lexicon list. They have used Wilson opinion lexicon list for semantic orientation. It is used to predict prior public opinion. The aim of paper is to find best effective features which provide better result and also provide better feature selection method. They have also express that how unigram feature set can be reduced to get better result. As a pre-processing sop word removal, stemming, pos tagging is performed. Mutual information (MI), Information gain (IG), Chi-square (λ2) and TF-IDF feature selection method is used to extract feature.

4.2 Important terms for SVM

- Confusion Matrix
- Quadratic Programming for Linear Data
- MMH :- Maximum Marginal Hyperplane
- Lagrange Multipliers Theory and optimal Laggrange Coefficient
- Hyperplane Equation : W*X + b =0

4.3 SVM Implementation

Support vector machine is non probabilistic algorithm which is used to separate data linearly and nonlinearly. Here dataset D = {Xi, Yi} where Xi is set of tuples and Yi is associated class label of tuples. Class labels are -1 and +1 for no and yes category respectively. The goal of SVM is to separate negative and positive training example by finding n-1 hyper plane.
Quadratic Programming (QP) problem is needed to be solved in linear data. This problem is transformed using the Lagrange Multipliers theory and Optimal Lagrange coefficients sets are obtained. A separating hyper plane is written as:

W*X + b = 0

where W = { w1, w2, w3, … , wn}, wn is weight vector of n attributes and b is bias. Distance from separating hyper plane to any point on H1 is 1/|W| and Distance from separating hyper plane to any point on H2 is 1/|W| so maximum margin is 2/|W|. The MMH is rewritten as the decision boundary according to Lagrangian formulation.

$$D(X)^T = \sum_{i=1}^{n} y_i a_i X_i X^T + b_0$$

Where XT is test tuple, αi and b0 are numeric parameters, yi is class label of support vector Xi. so If sign is positive of MMH equation then XT comes in positive category. If sign is negative of MMH equation then XT comes in negative category. SVM classifier formula is defined as following.

$$F(x) = \sum_{i=1}^{n} a_i k(x, x_i) + b$$

4.4 Sentiment Analysis Implementation using LIBSVM

4.4.1 Variable and constant used in LIBSVM

```
-s svm_type : set type of SVM (default 0)
	0 -- C-SVC
	1 -- nu-SVC
	2 -- one-class SVM
	3 -- epsilon-SVR
	4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
	0 -- linear: u'*v
	1 -- polynomial: (gamma*u'*v + coef0)^degree
	2 -- radial basis function: exp(-gamma*|u-v|^2)
	3 -- sigmoid: tanh(gamma*u'*v + coef0)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking: whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates: whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)
```

C : float, Penalty parameter C of the error term.
Kernel : string,
Specifies the kernel type to be used in the algorithm. It must be one of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable. If none is given, ‘rbf’ will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n samples, n samples).
Degree : int, Degree of the polynomial kernel function (‘poly’). Ignored by all other kernels.
Gamma : float, Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’. If gamma is ‘auto’ then 1/n_features will be used instead.
Coef0 : float, Independent term in kernel function. It is only significant in ‘poly’ and ‘sigmoid’.
Probability : boolean, Whether to enable probability estimates. This must be enabled prior to calling fit, and will slow down that method.
Shrinking : boolean, Whether to use the shrinking heuristic.
Tol : float, Tolerance for stopping criterion.
Cache size : float, Specify the size of the kernel cache (in MB).
Class weight : {dict, ‘balanced’}, Set the parameter C of class i to class_weight[i]*C for SVC. If not given, all classes are supposed to have weight one. The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n samples / (n classes * np.bincount(y))
verbose : bool, Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in libsvm that, if enabled, may not work properly in a multithreaded context.
Max iter : int , Hard limit on iterations within solver, or -1 for no limit.
Decision function shape : ‘ovo’, ‘ovr’ or None, default=None
Whether to return a one-vs-rest (‘ovr’) decision function of shape (n samples, n classes) as all other classifiers, or the original one-vs-one (‘ovo’) decision function of libsvm which has shape (n samples, n classes * (n classes - 1) / 2). The default of None will currently behave as ‘ovo’ for backward compatibility and raise a deprecation warning, but will change ‘ovr’ in 0.19.
Random state : int seed, The seed of the pseudo random number generator to use when shuffling the data for probability estimation.

4.4.2 Files provide by LIBSVM

- svm-predict.java
- svm-train.java
- svm-scale.java
- svm-toy.java

4.4.3 Requirements to run LIBSVM

- dictionary
- training file
- trained model
- svm-train
- svm-predict

4.4.4 Making the dictionary

Goal

Give unique id to every word present in the training file and save it as dictionary.

Algorithm

```
makeDictionary():
Step 1: path=<Input Path>
Step 2: read=readFile(path)
Step 3: read=swapSpace(read)
 Step 4: words=DistinctWords(read)
Step 5: write(words)
```

4.4.5 Making training data

Goal

First character indicating the sentence is positive or negative .
Positive : 1
Negative : 0
Later on replacing words with there unique id from dictionary.
If the word does not have any unique id in the dictionary than it will be ignored.
Then sentence will be sorted according to the unique id.
Occurrence of word is written next to word`s unique id separated by ":".
This format is to follow for all the sentence present in the training file.

Algorithm

```
trainModel():

Step 1: path=<Input File>

Step 2: read=readFile(path)

Step 3: Text= readFile(<Dictionary file path>)

Step 4: for   i=0 to i<read size

Step 5:             x=read(i).split()
Step 6:     s=<"0" for negative "1" for positive>

Step 7:     for  j=0 to j<Text size

Step 8:         temp=find(x[j],Text)

Step 9:         if(temp is not equals to "0")
                    temp2+=temp+":"+feature(x,j)+" "

Step  10:       if(feature(x,j)>1)
                    x=removeIndex(x,j)

Step 11:    if(temp2 is not equals to null)
                write(s + ascending(temp2))
```

4.4.6 Running LIBSVM

Input
Input should be in the same format as training file.
Give any prediction in the start
Positive :1
Negative :0

Format : Argv will be
    <input file path> <model file path> <output file path>

Output
You will get output in the output file as
0 : positive
1 : negative

5. Results

10,000 sentences were tested for sentiment analysis using the three algorithms. Results obtained are as follows-

5.1 Efficiency

The percentage of sentences that are correctly classified as positive or negative

Naive Bayes' – 87%
Maximum Entropy – 83%
Support Vector Machine – 89%

5.2 Round Trip Time (RTT)

The time taken to process Sentiment of one sentence. Results obtained are as follows-

Naive Bayes' – 9 seconds

Maximum Entropy – 1 seconds
Support Vector Machine – 3 seconds

6. Future Scope

In Naive Bayes', Conditional independence of words can be dropped ans N – grams can be used.
Also adjectives could have higher weightage than other words. Also adverbs could be used along with adjectives.
Sentiment Analysis needs to move beyond a one dimensional positive or negative scale.
In order to capture various emotions expressed by humans such as anxiety, excitement, fear, joy,etc in the form of text, we need a multi-dimensional scale.
With time organizations will become more aware about the advantages sentiment analysis can provide for their businesses.
The ultimate goal is to make sentiment analysis more accurate and useful for industry applications.

7.Abbreviations

• NER	: Name Entity Recognition
• NLP	: Natural language processing
• Entropy	: Transformation
• GIS	: Generalized Iterative Scaling
• IIS	: Improved Iterative Scaling
• POS	: Part Of Speech
• TF-IDF	: Term Frequency–Inverse Document Frequency

8. Sources

Speech and Language Processing. Daniel Jurafsky & James H. Martin.

http://scikit-learn.org/stable/modules/naive_bayes.html

https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/

http://sebastianraschka.com/Articles/2014_naive_bayes_1.html#3_3_multivariate

https://www.brandwatch.com/blog/understanding-sentiment-analysis/

https://people.cs.pitt.edu/~milos/courses/cs2750-Spring03/lectures/class11.pdf

https://www.cs.nyu.edu/~yann/2010f-G22-2565-001/diglib/lecture03a-svm-2010.pdf

http://www.ijcaonline.org/archives/volume146/number13/jadav-2016-ijca-910921.pdf

https://www.ijircce.com/upload/2014/january/16K_Sentiment.pdf

http://acl-arc.comp.nus.edu.sg/archives/acl-arc-090501d4/data/pdf/anthology-PDF/P/P02/P02-1002.pdf faculty.washington.edu/fxia/courses/LING572/MaxEnt.ppt

http://web.stanford.edu/class/cs276a/projects/reports/nmehra-kshashi-priyank9.pdfhttp://web.stanford.edu/class/cs276a/projects/reports/nmehra-kshashi-priyank9.pdf