Evaluation of Machine Learning Based Text Classifiers

Gautham Giridharan, Goutham Kannan

Abstract

With the increasing availability of electronic documents and the rapid growth of the World Wide Web, the task of automatic categorization of documents became the key method for organizing the information and knowledge discovery. Proper classification of e-documents, online news, blogs, e-mails and digital libraries need text mining, machine learning and natural language processing techniques to get meaningful knowledge We try to solve the problem of text document classification using machine learning techniques and evaluate their performances on different text classifiers.

Proposed Solution

We have used various techniques and algorithms for text document classification which are described below:

1) Data Collection:

As classification of Test has its applications in many avenues, the same classifier can't be used for multiple applications. We have handled this problem by considering different types of data, i.e. movie reviews and restaurant reviews. We have collected imdb movie reviews. The dataset was readily available online (https://www.dropbox.com/s/xk4glpk61q3qrg2/imdb.tgz?dl=1). We have collected the restaurant reviews from YelpAPI (https://www.yelp.com/developers/documentation/v2/search_api).

Structure of Data:

Data is preprocessed and organized as folders. Data folder contains two sub folders: Train and Test, further the subfolders contain two folders namely: pos and neg, which contain .txt files. Each text file contains a single review about the movie or the restaurant.

Folder Structure:
C:\ML_Project\ML_DATA\train\pos

Labelling of Data:
The data in the pos folder is labelled as positive (0) and the neg folder is labelled as negative (1).

Document Representation

Document Representation can be divided into:

Feature Extraction:

Tokenization: A document is treated as a string, and then partitioned into a list of tokens.

Stemming: Applying the stemming algorithm that converts different word form into similar canonical form. This step is the process of conflating tokens to their root eg: connection to connect, computing to compute.

Removing Stop words: Stop words such as '**the**', '**a**' are insignificant so we can remove them.

Parts of Speech: The process of classifying words into their **parts of speech** and labeling them accordingly is known as part-of-speech tagging, **POS-tagging**, or simply tagging. Parts of speech are also known as word classes or lexical categories. The collection of tags used for a particular task is known as a **tagset.**

Eg:

[('The', 'AT'), ('grand', 'JJ'), ('jury', 'NN'), ('commented', 'VBD'), ('on', 'IN'), ('a', 'AT'), ('number', 'NN'), ... ('.', '.')]

All the words in the document will be tagged with some tag names. We will use only the tags which we want from the **tagset** and generate a list of tokens.

We have implemented different types of tokenization:

1) Normal tokenize
2) Tokenize with stemming
3) Tokenize with POS

Feature Selection:

After feature extraction the important step in pre-processing of text classification, is feature selection to construct vector space, which improves the scalability, efficiency and accuracy of the text classifier. The main idea of FS is to select subset of features from the original documents. For text classification a major problem is the high dimensionality of the feature space.

Document Frequency(DF)	$DF(t_k) = P(t_k)$
Term Frequency(TF)	$tf(f_i, d_j) = \frac{freq_{ij}}{\max_k freq_{Kj}}$

Machine Learning Techniques:

Gaussian Naive Bayes

One approach to text classification is to assign to a given document *d* the class *c** = arg max*c* *P*(*c* | *d*).We derive the *Naive Bayes* (NB) classifier by first observing that by Bayes' rule,

$$P(c \mid d) = \frac{P(c)P(d \mid c)}{P(d)},$$

The data can be classified using:

$$g_i(x) = \sum_{j=1}^{n} X_j^i log(\alpha_j|y = 1) + (1 - X_j^i) log(\alpha_j|y = 1) + log(\alpha_j)$$

Model Parameters:

$\alpha_j|y = 1$ - alpha with respect to a particular class.

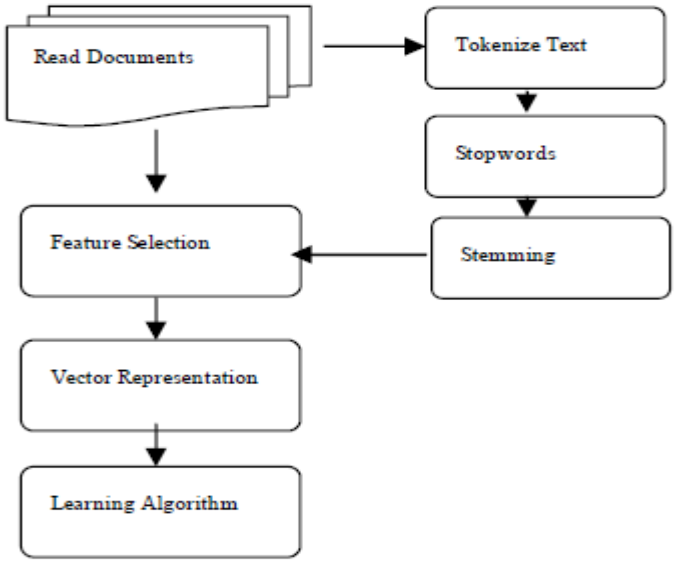
Support Vector Machines

SVM are highly effective at traditional text categorization than naïve bayes. In the two-category case, the basic idea behind the training procedure is to find a hyperplane, represented by vector ~*w*, that not only separates the document vectors in one class from those in the other, but for which the separation, or *margin*, is as large as possible. This search corresponds to a constrained optimization problem; letting $c_j \in \{1, -1\}$ (corresponding to positive and negative) be the correct class of document *dj* , the solution can be written as

$$\vec{w} := \sum_j \alpha_j c_j \vec{d}_j, \quad \alpha_j \geq 0,$$

where the α_j's are obtained by solving a dual optimization problem. Those $\vec{d}_j$ such that α_j is greater than zero are called *support vectors*, since they are the only document vectors contributing to $\vec{w}$.

Overall flow of Text Document Classification:



Performance Evaluation:

Accuracy:

Accuracy = TP +TN/ TP + TN + FP +FN

Precision:

Precision = TP/TP+FP

Recall:

Recall = TP/TP+FN

F-measure:

F-measure = 2.0 * Precision * Recall / Precision + Recall

Confusion Matrix:

[[TP FP]
[FN TN]]

Implementation Details

- I have used NumPy extensively to implement all the formulas and algorithms.
- I have used matplotlib to plot the graphs.
- I have used in built functions to verify the results with the implemented functions.
- I encapsulated all the work in a modular way using files and parameters

Program Design Issues:

- There was a problem of **"overflow error"** due to the fact that some of the some the computed values were out of range. This occurred for both the Logistic regression

- All the data are classified into a single class if we don't use proper learning rate and number of iterations.

- The code takes so much time to execute for both the dataset as the number of features in both the data set are large.

- There was a problem computing the Euclidean distance between two vectors for Gaussian Kernel function when I tried using NumPy functions, so I used used used pdist,squareform from SciPy to compute the Euclidean distance between the feature vectors.

- There was a problem of **"math value domain error"** due to the fact that some of the log values were undefined. I solved this problem using **Laplace Smoothing**. This occurred particularly in Bernoulli and Binomial Naïve Bayes case.

- All the data were classified into a single class if we don't use appropriate values for the parameters in gradient descent.

Results and Discussion:

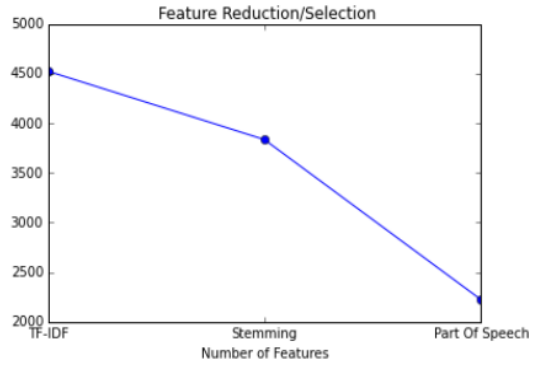
Movie reviews (IMDB reviews)

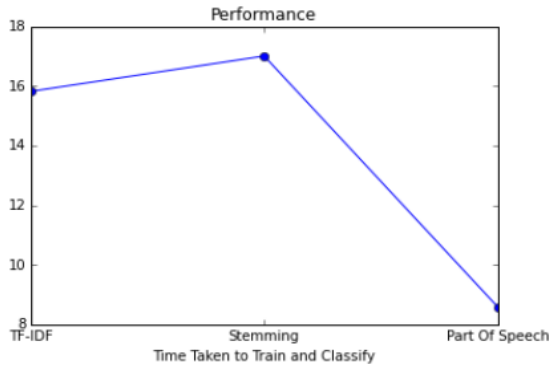
Logistic Regression

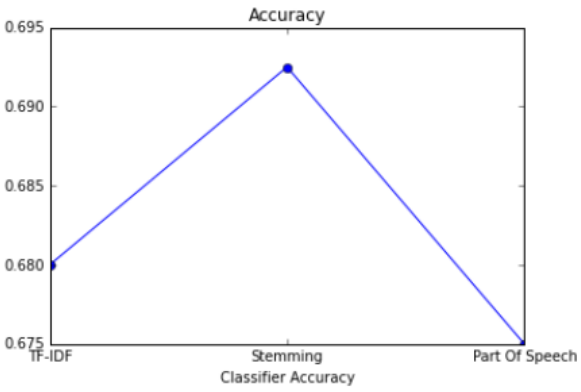
The following table shows the performance using different types of tokenizer on Logisitic Regression classifier.

Tokenizers	***Features***	***Accuracy***	***Precision***	***Recall***	***F-Measure***
Tokenize(TF-IDF)	4524	0.68	0.653	0.765	0.705
Tokenize with Stemming	3835	0.6925	0.665	0.775	0.715
Tokenize with POS	2224	0.675	0.653	0.745	0.696

The following graphs show the feature reduction, time taken to train and classify and classifier accuracy based on the different types of tokenizers.





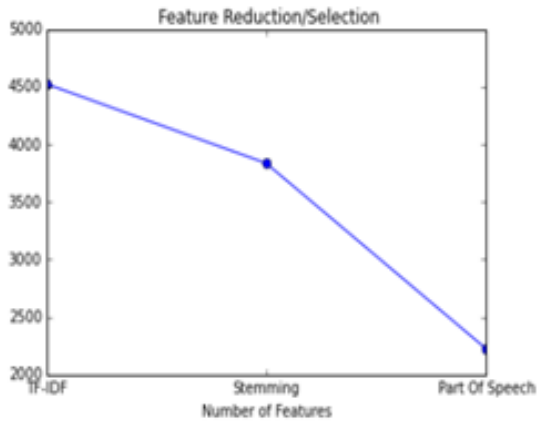


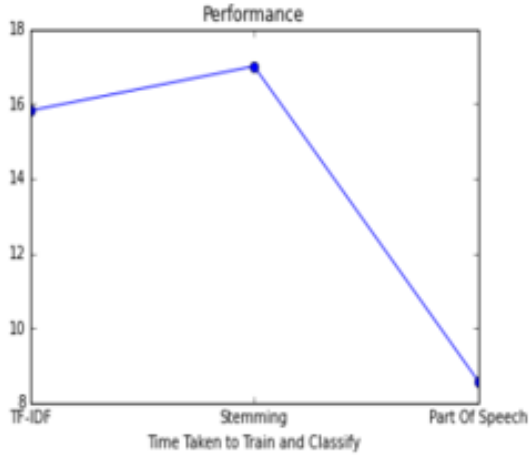
Gaussian Naïve Bayes

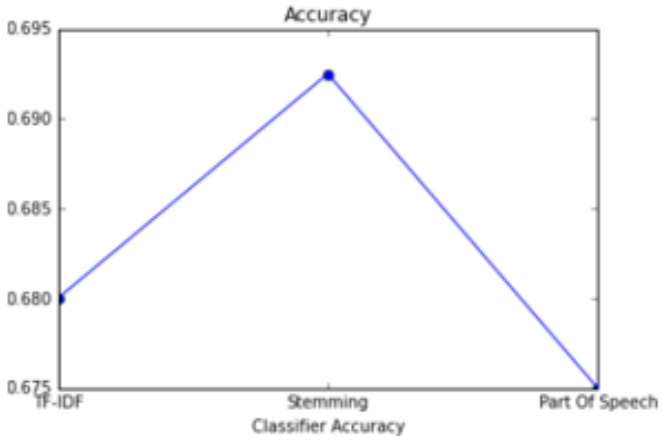
The following table shows the performance using different types of Tokenizer on Gaussian Naïve Bayes classifier.

Tokenizers	***Features***	***Accuracy***	***Precision***	***Recall***	***F-Measure***
Tokenize(TF-IDF)	4524	0.6525	0.634	0.72	0.674
Tokenize with Stemming	3835	0.6825	0.674	0.705	0.689
Tokenize with POS	2224	0.6575	0.690	0.57	0.624

The following graphs show the feature reduction, time taken to train and classify and classifier accuracy based on the different types of tokenizers.





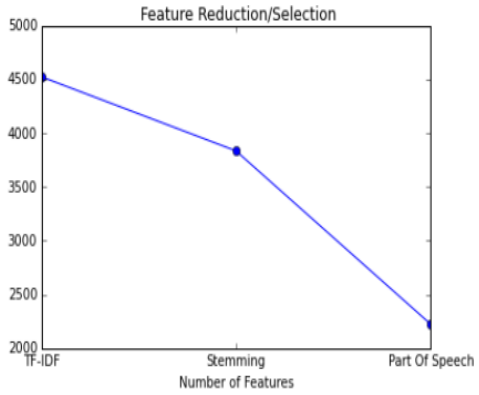


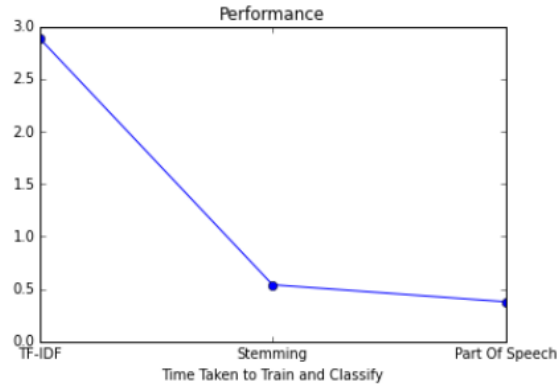
Support Vector Machines

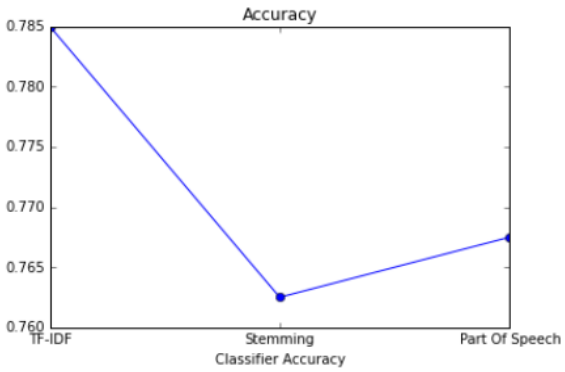
The following table shows the performance using different types of Tokenizer on Support Vector Machines classifier.

Tokenizers	***Features***	***Accuracy***	***Precision***	***Recall***	***F-Measure***
Tokenize(TF-IDF)	4524	0.785	0.739	0.88	0.803
Tokenize with Stemming	3835	0.7625	0.692	0.945	0.799
Tokenize with POS	2224	0.7675	0.753	0.795	0.773

The following graphs show the feature reduction, time taken to train and classify and classifier accuracy based on the different types of tokenizers.







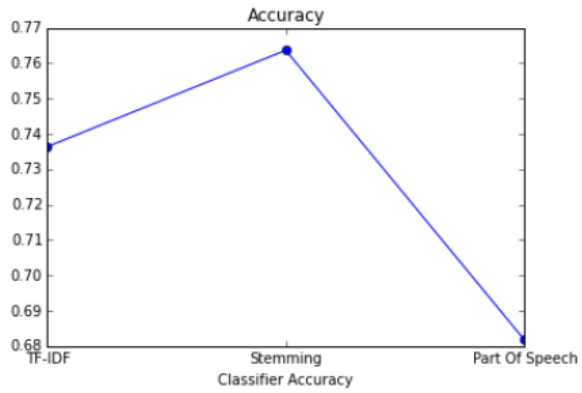
Restaurant Reviews (YelpAPI)

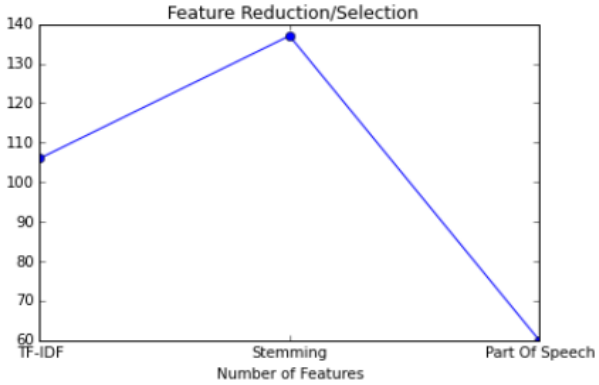
Logistic Regression

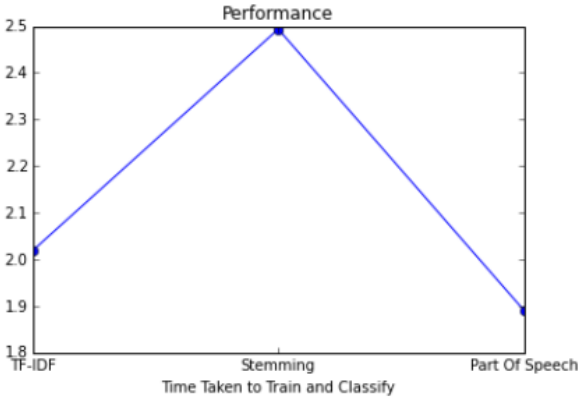
The following table shows the performance using different types of Tokenizer on Logisitic Regression classifier.

Tokenizers	*Features*	*Accuracy*	*Precision*	*Recall*	*F-Measure*
Tokenize(TF-IDF)	106	0.736	0.718	0.807	0.760
Tokenize with Stemming	137	0.763	0.746	0.824	0.783
Tokenize with POS	60	0.681	0.677	0.736	0.705

The following graphs show the feature reduction, time taken to train and classify and classifier accuracy based on the different types of tokenizers.





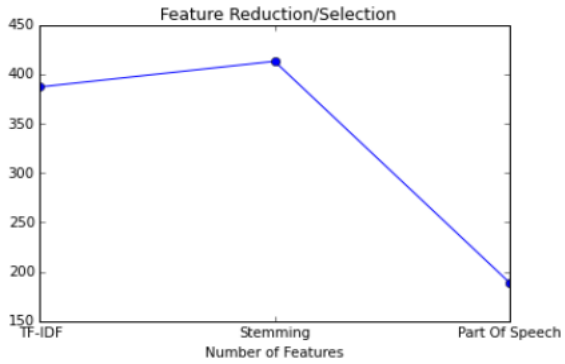


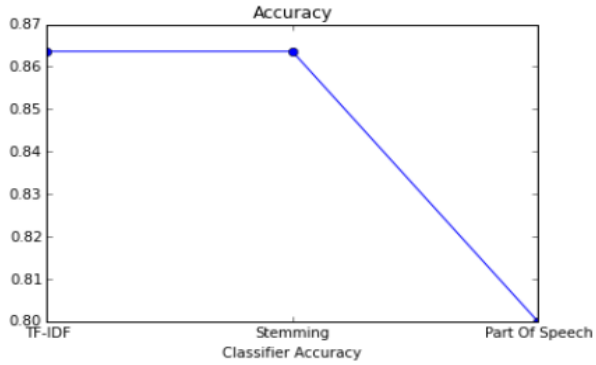
Gaussian Naïve Bayes

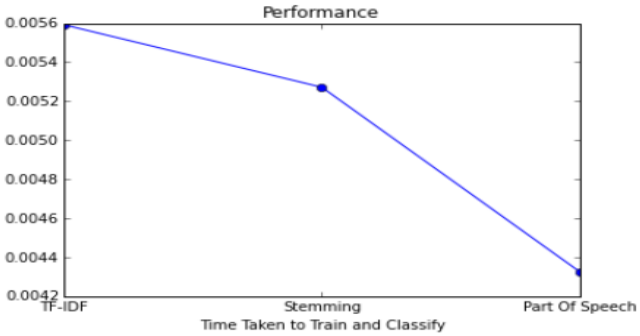
The following table shows the performance using different types of tokenizer on Gaussian Naïve Bayes classifier.

Tokenizers	***Features***	***Accuracy***	***Precision***	***Recall***	***F-Measure***
Tokenize(TF-IDF)	387	0.863	0.862	0.877	0.869
Tokenize with Stemming	413	0.863	0.977	0.754	0.851
Tokenize with POS	189	0.8	0.761	0.894	0.822

The following graphs show the feature reduction, time taken to train and classify and classifier accuracy based on the different types of tokenizers.





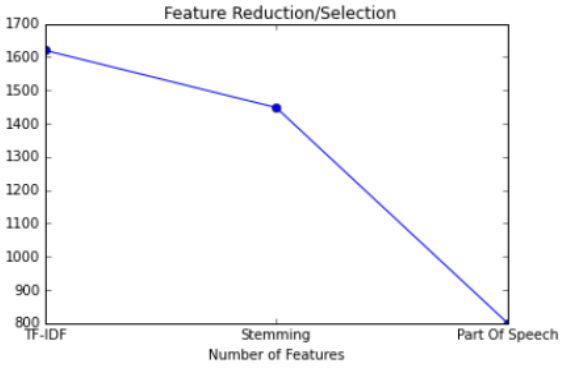


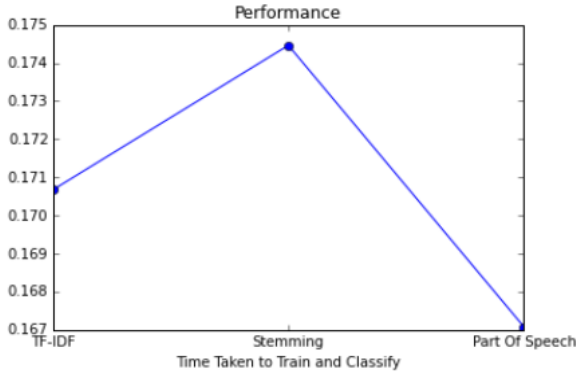
Support Vector Machines:

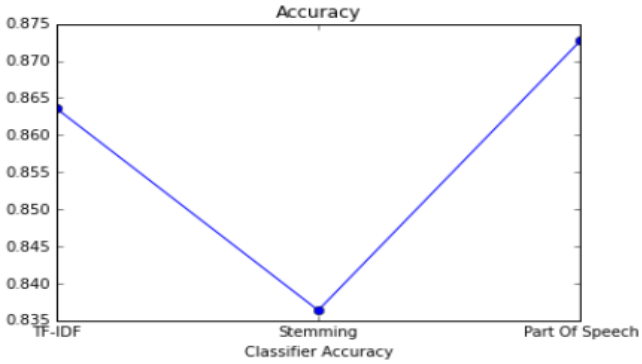
The following table shows the performance using different types of tokenizer on Support Vector Machines classifier.

Tokenizers	***Features***	***Accuracy***	***Precision***	***Recall***	***F-Measure***
Tokenize(TF-IDF)	1620	0.863	0.838	0.912	0.873
Tokenize with Stemming	1448	0.836	0.854	0.824	0.839
Tokenize with POS	801	0.872	0.890	0.859	0.875

The following graphs show the feature reduction, time taken to train and classify and classifier accuracy based on the different types of tokenizers.







Following tables show the top positive and neagtive coeffients found in the text documents(Yelp and IMDb data sets)

Positive Coefficients	Values
u'great'	0.76481850215725622
u'excellent'	0.76048804020699867
u'superb'	0.66125758551728364
u'worth'	0.59417533424060176
u'liked'	0.54575595599943894
u'loved'	0.5341933356702695
u'enjoyed'	0.47802820013861308

Negative Coefficients	Values
u'boring'	-0.78126164334045312
u'bad'	-0.76048804020699867
u'terrible'	-0.53879153203649954
u'poor'	-0.52289306645282474
u'awful'	-0.54575595599943894
u'waste'	-0.52289306645282474
u'almost	-0.50515419092424174

Conclusion:

- We found that there is a reduction in the number of features when we use different types of tokenizer.
- Tokenize with part of speech performs better when we consider the dimensionality reduction.
- As far as classifier accuracy is concerned Tokenize with Stemming works better.
- Performance with tokenize with stemming is better when compared to other tokenizer functions.
- We have varied different parameters in the CountVectorizer like min-df, max-df, ngram-range and found different results.
- For different datasets, we find that Support Vector Machines classifier performs well when used with Tokenize with part of speech tokenizer.

Future Works:

- Neural Networks and Genetic Algorithm could be implemented for classification of text documents.
- Other concepts of feature extraction based on sentiments could also be applied so that we can capture the exact emotion of the reviewer.

References:

1. Lillian Lee, Bo Pang, “Thumbs up? Sentiment Classification using Machine Learning Techniques”,

http://www.cs.cornell.edu/home/llee/papers/sentiment.pdf

2. Baharum Baharudin, Lam Hong Lee, and Khairullah Khan.A review of machine learning algorithms for text-documents classification. Journal of advances in information technology,1(1):4–20, 2010

https://www.researchgate.net/publication/43121576_A_Review_of_Machine_Learning_Algorithms_for_Text-Documents_Classification