SENTIMENT ANALYSIS ON MOVIE REVIEWS

A PROJECT REPORT

Submitted by

TADI PALLAVI	314126510100
SUMAN MISHRA	314126510095
P.M.SAI SUBODH	314126510077
G.BALARAJU	314126510136

in partial fulfillment for the award of the degree
of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING

Under esteemed guidance of

Ms. J.Sharmila Rani

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Affiliated to Andhra University)

SANGIVALASA, VISAKHAPATNAM - 531162

2017-2018

A PROJECT REPORT

SENTIMENT ANALYSIS ON MOVIE REVIEWS

Submitted by

TADI PALLAVI	314126510100
SUMAN MISHRA	314126510095
P.M.SAI SUBODH	314126510077
G.BALARAJU	314126510136

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

Under esteemed guidance of

Ms. J.Sharmila Rani

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Affiliated to Andhra University)

SANGIVALASA, VISAKHAPATNAM - 531162

2017-2018

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Affiliated to Andhra University)
SANGIVALASA, VISAKHAPATNAM-531162

BONAFIDE CERTIFICATE

Certified that this project report "**SENTIMENT ANALYSIS ON MOVIE REVIEWS"** is the bonafide work of " T. PALLAVI(314126510100), SUMAN MISHRA(314126510095) , P.M SAI SUBODH(314126510077), G. BALA RAJU(314126510136)" who carried out the project work under my supervision.

Dr.R.Sivaranjini
HEAD OF THE DEPARTMENT
Professor
Department of Computer Science Engineering
ANITS

J.Sharmila Rani
PROJECT GUIDE
Assistant Professor
Department of Computer Science Engineering
ANITS

DECLARATION

This is to certify that the project work entitled "**SENTIMENT ANALYSIS ON MOVIE REVIEWS**" is a bonafide work carried out by **T.PALLAVI, SUMAN MISHRA, P.M.SAI SUBODH, G.BALARAJU** as a part of B.TECH final year 2nd semester of computer science Engineering of Andhra University, Visakhapatnam during the year 2017-18.

We **T.PALLAVI , SUMAN MISHRA, P.M.SAI SUBODH, G.BALARAJU** of final semester B.Tech, in the department of Computer Science Engineering from ANITS, Visakhapatnam, hereby declare that the project work entitled **SENTIMENT ANALYSIS ON MOVIE REVIEWS** is carried out by us and submitted in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science Engineering , under Anil Neerukonda Institute of Technology & Sciences during the academic year 2017-18 and has not been submitted to any other university for the award of any kind of degree.

TADI PALLAVI	314126510100
SUMAN MISHRA	314126510095
P.M.SAI SUBODH	314126510077
G.BALARAJU	314126510136

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement always boosted the morale. We take a great pleasure in presenting a project, which is the result of a studied blend of both research and knowledge.

We first take the privilege to thank the Head of our Department, **Dr.R.Sivaranjani**, for permitting us in laying the first stone of success and providing the lab facilities, we would also like to thank the other staff in our department and lab assistants who directly or indirectly helped us in successful completion of the project.

We feel great to thank , **Ms. J.Sharmila Rani** who is our project guide and who shared her valuable knowledge with us and made us understand the real essence of the topic and created interest in us to work day and night for the project; we also thank our B.Tech coordinator **Mr.K.Suresh & Mrs.T.Kranthi**, for their support and encouragement.

We also like to thank **Mr.P.Sainath Vital** for providing us the lab resources; we also thank our friends and college staff who extended their part of support in the successful completion of the project.

TADI PALLAVI	314126510100
SUMAN MISHRA	314126510095
P.M.SAI SUBODH	314126510077
G.BALARAJU	314126510136

ABSTRACT

Sentiment analysis is a sub-domain of opinion mining where the analysis is focused on the extraction of emotions and opinions of the people towards a particular topic from a structured,semi-structured or unstructured textual data. we try to focus our task of sentiment analysis on IMDB movie review database. Sentiment Analysis is a process of extracting information from large amount of data, and classifies them into different classes called sentiments. Python is simple yet powerful, high-level, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data by using NLTK (Natural Language Toolkit). NLTK is a library of python, which provides a base for building programs and classification of data. NLTK also provide graphical demonstration for representing various results or trends and it also provide sample data to train and test various classifier respectively. Sentiment classification aims to automatically predict sentiment polarity of users publishing sentiment data. Traditional classification algorithm can be used to train sentiment classifiers from manually labeled text data . We directly apply a classifier trained to the domain to the performance will be very low due to the difference between these domains. In this work, we develop a general solution to sentiment classification when we do not have any labels in target domain but have some labeled data in a different domain, regarded as source domain .In this project, we attempt not to only detect sarcasm in text and made pilot Model Sarcasm with Naive Bayes using TFIDF feature vectors.

Keywords— Sentiment Analysis, NLTK (Natural Language Toolkit), Python

CONTENTS

1. INTRODUCTION

1.1 PROBLEM STATEMENT

Text can be categorized in two types based on its properties in terms of text mining: ‘subjectivity’ and ‘polarity’. The focus of our project is to find the polarity of the text which means that we are interested in finding if the sentence is positive or negative. We use machine learning techniques classify such sentences and try to find answers to the following questions:

1. Machine learning techniques their purpose, which one out of them performs the best and which techniques are better than the others.
2. Advantages and disadvantages of traditional machine learning techniques for sentiment analysis.
3. The difficulticulty in the task of extracting sentiment from short comments or sentences can be as compared to the traditional topic based text classification.

1.2 MOTIVATION

Sentimental analysis is a hot topic of research. Use of electronic media is increasing day by day. Time is money or even more valuable than money therefore instead of spending times in reading and figuring out the positivity or negativity of text we can use automated techniques for sentiment analysis. Sentiment analysis is used in opinion mining. The applications of sentiment analysis are broad and powerful. The ability to extract insights from social data is a practice that is being widely adopted by organisations across the world. Shifts in sentiment on social media have been shown to correlate with shifts in the stock market.

Example – Analyzing a product based on it’s review and comments.

1.3 CONTRIBUTION

In our work, we have used a supervised learning technique to tag the reviews of a movie using the IMDB dataset present in NLTK corpora. We have used Naive Bayes classifier on movie reviews and made a Pilot Model Naive Bayes with TFIDF feature vectors on sarcastic reviews. Various stemmers and lemmatizers have also been used.

1.4 RESEARCH METHODOLOGY

Sentiment Analysis is very interesting and an attention needed one. There are many classifiers like svm's, J48 algorithm, BETREE algorithm for an English language, but there are no such which are domain independent . This research focuses mainly on analysing the movie reviews of review polarity dataset contains 2000 IMDB movie reviews , in which 1000 are positive reviews and 1000 are negative reviews .70 percentage of the data is used for training the model and remaining 30 percent is used for testing the model.

2. LITERATURE SURVEY

2.1 Introduction to Sentiment Analysis

Sentiment analysis is becoming one of the most profound research areas for prediction and classification. Automated sentiment analysis of text is used in fields where products and services are reviewed by customers and critics. Thus, sentiment analysis becomes important for businesses to draw a general opinion about their products and services. Our analysis helps concerned organizations to find opinions of people about movies from their reviews, if it is positive or negative. One can in turn formulate a public opinion about a movie. Our goal is to calculate the polarity of sentences that we extract from the text of reviews. We will model sentiment from movie reviews and try to find out how this sentiment matches with the success for these movies. In other words, if a movie review is positive, negative or neutral. But this task can be difficult and tricky. Consider a sentence “the movie interstellar was visually a treat but the story line was terrible”. Now one can clearly see how categorizing this sentence as negative, positive or neutral can be difficult. The phrases “visually a treat” and “story line was terrible” can be considered positive and negative respectively but the degree of their positiveness’ and ‘negativeness’ is somewhat ambiguous. We use a score for common positive and negative words and use this score to calculate the overall sentiment of a sentence.

Based upon the above reasons, this project applies the Naive Bayes classifier machine learning techniques to predict polarity of movie reviews as positive or negative by understanding meaning and relationship between the words. Mining the movie reviews and generating valuable metadata provides an opportunity to understand the general sentiment around movies in an independent way. The project is implemented using Python Programming Language and machine learning libraries of Python to predict sentiment of movie reviews as positive or negative using Naive Bayes classifier machine learning algorithm.

2.1 Classification

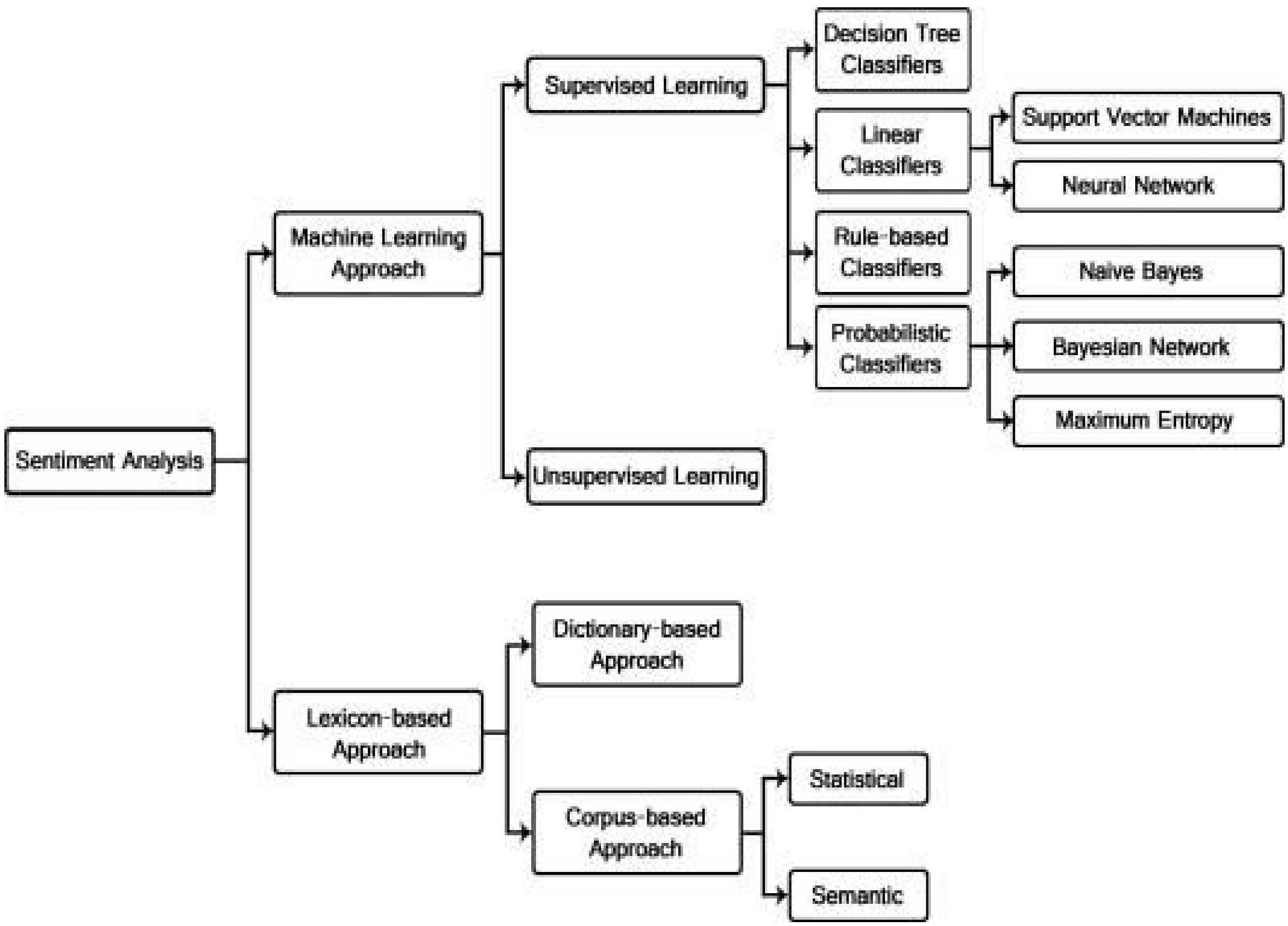


Fig 2.1 Classification

2.3 Models

There are many ways to implement Sentiment Analysis. Ultimately, it is a text classification problem and can be broken down into two main areas

Supervised models: This technique involves the construction of a "Classifier" and the problem has been studied intensively. The Classifier is responsible for categorizing texts into either a positive, negative or neutral polarity The three main classification techniques are:

i) Naive Bayes

ii) Maximum Entropy

iii) Support Vector Machines (SVM)

Unsupervised models: Unsupervised Learning has three steps.

1. Implement POS tagging(Part of Speech), then, two consecutive words are extracted to identify if their tags conform to given patterns.

2.Estimate the sentiment orientation (SO) of the extracted phrase.

3.Compute the average SO of all phrases that were extracted in terms of positive or negative.

2.4 Applications and Benefits

Sentiment analysis has many applications and benefits to your business and organization. It can be used to give your business valuable insights into how people feel about your product brand or service.

i) When applied to social media channels, it can be used to identify spikes in sentiment, thereby allowing you to identify potential product advocates or social media influencers.

ii) It can be used to identify when potential negative threads are emerging online regarding your business, thereby allowing you to be proactive in dealing with it more quickly.

Iii) Sentiment analysis could also be applied to your corporate network, for example, by applying it to your email server, emails could be monitored for their general “tone”. For example,Tone Detector is an Outlook Add-in that determines the “tone” of your email as you type. Like an emotional spell checker for all of your outgoing email.

2.5 Sentiment Analysis Methods

The sentiment classification approaches can be classified in

(i) Machine Learning : The machine learning approach is used for predicting the polarity of sentiments based on trained as well as test data sets

(ii) Lexicon based : Lexicon based approach does not need any prior training in order to mine the data. It uses a predefined list of words, where each word is associated with a specific sentiment.

(iii) Hybrid approach : combination of both the machine learning and the lexicon based approaches has the potential to improve the sentiment classification performance.

2.6 Feature Extraction

Text Analysis is a main application field for mechanism learning processes. However the raw information, an order of symbols cannot be fed straight into the algorithms themselves as maximum of them expect arithmetical feature paths with a fixed size somewhat than the raw text forms with variable length. In imperative to address this, sickie-learn offers utilities for the most mutual ways to extract numerical structures out of texts, as follows:

• Tokenizing the strings and giving an integer id for each imaginable token, for example by using white-spaces & punctuation as symbolic separators.

• Counting the existences of tokens in each document.

• Regulating and weighting with diminishing importance tokens that occur in the majority of samples / forms.

3. SYSTEM REQUIREMENT SPECIFICATION

3.1 Softwares used in the project are:

Language: Python.

Software used: PyCharm, Python27, Python GUI, NLTK tool Kit.

Operating System: Windows 10, Linux

3.2 Hardwares used in the project are:

Processor: intel Multi Core processor

RAM: 4 GB or above

Hard Disk:500 GB or above

3.2.1 User Interface

The work of the user can enter a review and get the polarity of the review

3.2.2 Hardware Interace

Monitor: The outputs are displayed on the monitor screen.

3.2.3 Software Interface

PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for the Python language. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django. PyCharm is cross-platform, with Windows, macOS and Linux versions.

4. EXISTING SYSTEM

4.1 Existing Sentiment technique

Naive Bayes Algorithm: Bayes theorem named after Rev. Thomas Bayes. It works on conditional probability. Conditional probability is the probability that something will happen, ***given*** *that something else* has already occurred. Using the conditional probability, we can calculate the probability of an event using its prior knowledge. Naive Bayes model is easy to build and particularly useful for very large data sets

Below is the formula for calculating the conditional probability

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

where

- P(c) is the probability of hypothesis c being true. This is known as the prior probability.
- P(x) is the probability of the evidence(regardless of the hypothesis).
- P(x|c) is the probability of the evidence given that hypothesis is true.
- P(c|x) is the probability of the hypothesis given that the evidence is there.

Working:

Let’s understand it using an example. Below I have a training data set of weather and corresponding target variable ‘Play’ (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let’s follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

4.2 Disadvantages

The first disadvantage is that the Naive Bayes classifier makes a very strong assumption on the shape of your data distribution, i.e. any two features are independent given the output class. Due to this, the result can be (potentially) very bad hence, a “naive” classifier

Another problem happens due to data scarcity. For any possible value of a feature, you need to estimate a likelihood value by a frequentist approach. This can result in probabilities going towards 0 or 1, which in turn leads to numerical instabilities and worse results

5. Proposed System

5.1 Classifier

5.1.1 Naive Bayes Classification

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 .

Being relatively robust, easy to implement, fast, and accurate, naive Bayes classifiers are used in many different fields. Some examples include the diagnosis of diseases and making decisions about treatment processes the classification of RNA sequences in taxonomic studies , and spam filtering in e-mail clients .

However, strong violations of the independence assumptions and non-linear classification problems can lead to very poor performances of naive Bayes classifiers.

We have to keep in mind that the type of data and the type problem to be solved dictate which classification model we want to choose. In practice, it is always recommended to compare different classification models on the particular dataset and consider the prediction performances as well as computational efficiency.

In the following sections, we will take a closer look at the probability model of the naive Bayes classifier and apply the concept to a simple toy problem. Later, we will use a publicly available SMS (text message) collection to train a naive Bayes classifier in Python that allows us to classify unseen messages as spam or ham.

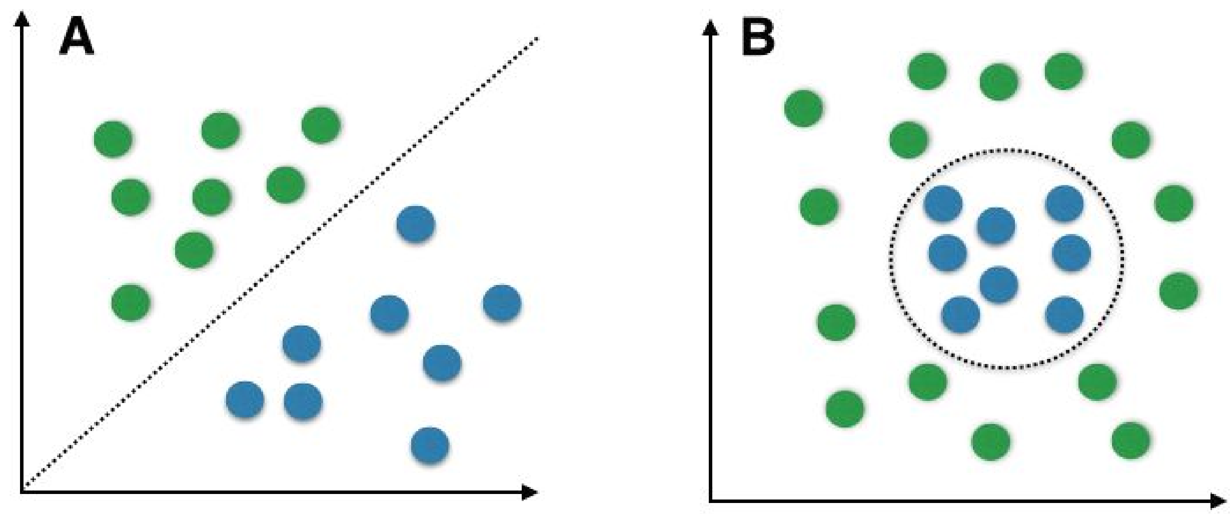


Figure 5.1.1 Linear (A) vs non-linear problems (B).

Random samples for two different classes are shown as colored spheres, and the dotted lines indicate the class boundaries that classifiers try to approximate by computing the decision boundaries.

A non-linear problem (B) would be a case where linear classifiers, such as naive Bayes, would not be suitable since the classes are not linearly separable. In such a scenario, non-linear classifiers (e.g.,instance-based nearest neighbor classifiers) should be preferred.

5.1.2 Posterior Probabilities

In order to understand how naive Bayes classifiers work, we have to briefly recapitulate the concept of Bayes' rule. The probability model that was formulated by Thomas Bayes (1701-1761) is quite simple yet powerful; it can be written down in simple words as follows:

posterior probability = conditional probability · prior probability evidence posterior probability=conditional probability · prior probability evidence

Bayes' theorem forms the core of the whole concept of naive Bayes classification. The posterior probability, in the context of a classification problem, can be interpreted as: "What is the probability that a particular object belongs to class ii given its observed feature values?" A more concrete example would be: “What is the probability that a person has diabetes given a certain value for a pre-breakfast blood glucose measurement and a certain value for a post-breakfast blood glucose measurement?”

$$P(\text{diabetes} \mid \mathbf{x}_i), \mathbf{x}_i = [90\text{mg/dl}, 145\text{mg/dl}]$$

Let

- $\mathbf{x}_i$ be the feature vector of sample $i, i \in \{1,2,...,n\}$,
- ω_j be the notation of class $j, j \in \{1,2,...,m\}$,
- and $P(\mathbf{x}_i \mid \omega_j)$ be the probability of observing sample $\mathbf{x}_i$ given that is belongs to class ω_j.

The general notation of the posterior probability can be written as

$$P(\omega_j \mid \mathbf{x}_i) = \frac{P(\mathbf{x}_i \mid \omega_j) \cdot P(\omega_j)}{P(\mathbf{x}_i)}$$

The objective function in the naive Bayes probability is to maximize the posterior probability given the training data in order to formulate the decision rule.

To continue with our example above, we can formulate the decision rule based on the posterior probabilities as follows:

$$\text{person has diabetes if} \; P(\text{diabetes} \mid \mathbf{x}_i) \geq P(\text{not-diabetes} \mid \mathbf{x}_i), \text{else classify person as healthy.}$$

5.1.3 Class-conditional Probabilities

One assumption that Bayes classifiers make is that the samples are *i.i.d.*

The abbreviation *i.i.d.* stands for “independent and identically distributed” and describes random variables that are independent from one another and are drawn from a similar probability distribution. Independence means that the probability of one observation does not affect the probability of another observation . One popular example of *i.i.d.*variables is the classic coin tossing: The first coin flip does not affect the outcome of a second coin flip and so forth. Given a fair coin, the probability of the coin landing on “heads” is always 0.5 no matter of how often the coin if flipped.

An additional assumption of naive Bayes classifiers is the conditional independence of features. Under this naive assumption, the class-conditional probabilities or (likelihoods) of the samples can be directly estimated from the training data instead of evaluating all possibilities of xx. Thus, given a dd-dimensional feature vector xx, the class conditional probability can be calculated as follows:

P(x∣ωj)=P(x1∣ωj)⋅P(x2∣ωj)⋅…⋅P(xd∣ωj)=∏k=1dP(xk∣ωj)P(x∣ωj)=P(x1∣ωj)⋅P(x2∣ωj)⋅…⋅P(xd∣ωj)=∏k=1dP(xk∣ωj)

Here, P(x∣ωj)P(x∣ωj) simply means: “How likely is it to observe this particular pattern xx given that it belongs to class ωjωj?” The “individual” likelihoods for every feature in the feature vector can be estimated via the maximum-likelihood estimate, which is simply a frequency in the case of categorical data:

P^(*xi*∣*ωj*)=*Nxi*,*ωjNωj*(*i*=(1,...,*d*))P^(xi∣ωj)=Nxi,ωjNωj(i=(1,...,d))

- *Nxi*,*ωj*Nxi,ωj: Number of times feature *xi*xi appears in samples from class *ωj*ωj.
- *Nωj*Nωj: Total count of all features in class *ωj*ωj.

5.1.4 Multinomial Naive Bayes - A Toy Example

After covering the basics concepts of a naive Bayes classifier, the posterior probabilities and decision rules, let us walk through a simple toy example based on the training set shown in Figure 5.1.2.

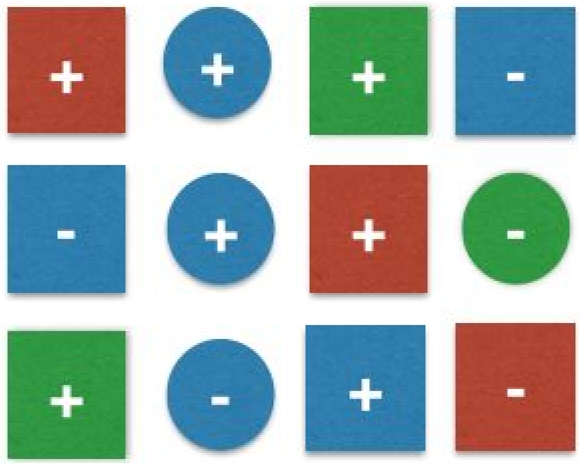


Fig 5.1.2 Example for Multinomaial NB , the toy example

5.1.5 Maximum-Likelihood Estimates

The decision rule can be defined as

Classify sample as + ifP(ω=+∣x=[blue, square])≥P(ω=-∣x=[blue, square])else classify sample as−.Classify sample as + ifP(ω=+∣x=[blue, square])≥P(ω=-∣x=[blue, square])else classify sample as−.

Under the assumption that the samples are i.i.d, the prior probabilities can be obtained via the maximum-likelihood estimate (i.e., the frequencies of how often each class label is represented in the training dataset):

P(+)=712=0.58P(-)=512=0.42P(+)=712=0.58P(-)=512=0.42

Under the naive assumption that the features “color” and “shape” are mutually independent, the class-conditional probabilities can be calculated as a simple product of the individual conditional probabilities.

Via maximum-likelihood estimate, e.g., P(blue∣−)P(blue∣−) is simply the frequency of observing a “blue” sample among all samples in the training dataset that belong to class −−.

P(x∣+)=P(blue∣+)⋅P(square∣+)=37⋅57=0.31P(x∣−)=P(blue∣−)⋅P(square∣−)=35⋅35=0.36P(x∣+)=P(blue∣+)⋅P(square∣+)=37⋅57=0.31P(x∣−)=P(blue∣−)⋅P(square∣−)=35⋅35=0.36

Now, the posterior probabilities can be simply calculated as the product of the class-conditional and prior probabilities:

P(+∣x)=P(x∣+)⋅P(+)=0.31⋅0.58=0.18P(−∣x)=P(x∣−)⋅P(−)=0.36⋅0.42=0.15P(+∣x)=P(x∣+)⋅P(+)=0.31⋅0.58=0.18P(−∣x)=P(x∣−)⋅P(−)=0.36⋅0.42=0.15

Classification

Putting it all together, the new sample can be classified by plugging in the posterior probabilities into the decision rule:

If *P*(+∣x)≥*P*(-∣x)classify as +,else classify as −If P(+∣x)≥P(-∣x)classify as +,else classify as −

P^(*xi*∣*ωj*)=*Nxi*,*ωj*+*αNωj*+*αd*(*i*=(1,...,*d*))P^(xi∣ωj)=Nxi,ωj+αNωj+αd(i=(1,...,d))

where

- *Nxi*,*ωj*Nxi,ωj: Number of times feature *xi*xi appears in samples from class *ωj*ωj.
- *Nωj*Nωj: Total count of all features in class *ωj*ωj.
- *α*α: Parameter for additive smoothing.
- *d*d: Dimensionality of the feature vector x=[*x*1,...,*xd*]x=[x1,...,xd].

5.1.6 Proposed System

1. Dataset is uploaded with social media data and divided into three categories: positive, negative, and neutral.
2. Feature extraction algorithm is applied on the dataset.
3. Features are extracted in the form of Eigen vectors and Eigen values.
4. For instance selection, genetic algorithm is applied. Population size is selected and initialize the genetic algorithm operators are initialized.Selection operator is used to initialize the data. Crossover operator is used to divide the data into two categories according to Eigen values and vector range .Mutation operator is applied for end movement modification.

5.2 Architecture

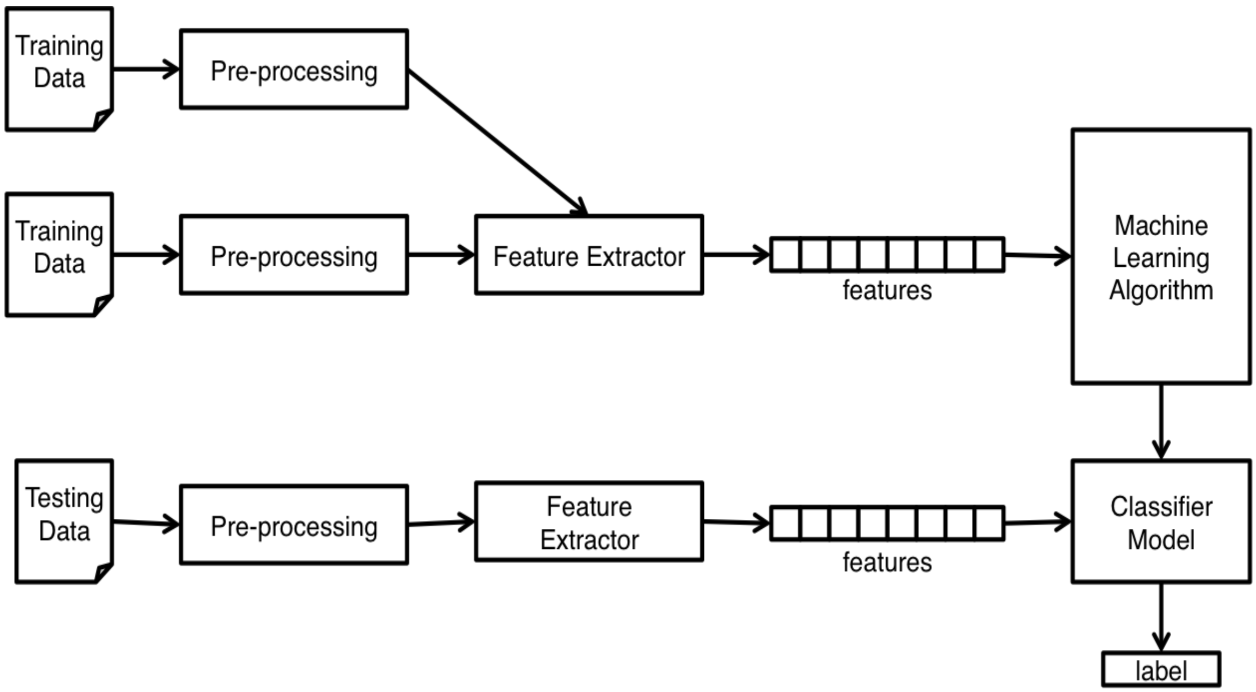


Fig 5.2 Architecture

5.3 Processing Steps

5.3.1 Naive Bayes and Text Classification

This section will introduce some of the main concepts and procedures that are needed to apply the naive Bayes model to text classification tasks. Although the examples are mainly concerning a two-class problem — classifying text messages as *spam* or *ham* — the same approaches are applicable to multi-class problems such as classification of documents into different topic areas (e.g., “Computer Science”, “Biology”, “Statistics”, “Economics”, “Politics”, etc.).

5.3.2 The Bag of Words Model

One of the most important sub-tasks in pattern classification are *feature extraction* and *selection*; the three main criteria of good features are listed below:

- Salient. The features are important and meaningful with respect to the problem domain.
- Invariant. Invariance is often described in context of image classification: The features are insusceptible to distortion, scaling, orientation, etc. A nice example is given by C. Yao and others in Rotation-Invariant Features for Multi-Oriented Text Detection in Natural Images .
- Discriminatory. The selected features bear enough information to distinguish well between patterns when used to train the classifier.

Prior to fitting the model and using machine learning algorithms for training, we need to think about how to best represent a text document as a feature vector. A commonly used model in Natural Language Processing is the so-called bag of words model. The idea behind this model really is as simple as it sounds. First comes the creation of the vocabulary — the collection of all different words that occur in the training set and each word is associated with a count of how it occurs. This vocabulary can be understood as a set of non-redundant items where the order doesn't matter. Let D1D1 and D2D2 be two documents in a training dataset:

- D1D1: "Each state has its own laws."
- D2D2: "Every country has its own culture."

Based on these two documents, the vocabulary could be written as

V={each:1,state:1,has:2,its:2,own:2,laws:1,every:1,country:1,culture:1}V={each:1,state:1,has:2,its:2,own:2,laws:1,every:1,country:1,culture:1}

The vocabulary can then be used to construct the dd-dimensional feature vectors for the individual documents where the dimensionality is equal to the number of different words in the vocabulary (d=|V|d=|V|). This process is called vectorization.

Table 1. Bag of words representation of two sample documents D1D1 and D2D2.

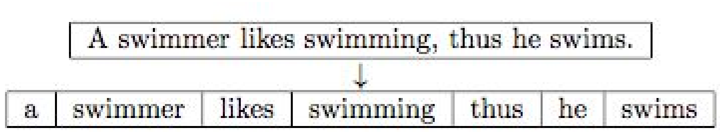
	each	state	has	its	own	laws	every	country	culture
x*D*1xD1	1	1	1	1	1	1	0	0	0
x*D*2xD2	0	0	1	1	1	0	1	1	1
ΣΣ	1	1	2	2	2	1	1	1	1

Given the example in Table 1 one question is whether the 1s and 0s of the feature vectors are binary counts (1 if the word occurs in a particular document, 0 otherwise) or absolute counts (how often the word occurs in each document). The answer depends on which probabilistic model is used for the naive Bayes classifier: The Multinomial or Bernoulli model — more on the probabilistic models in Section Multi-variate Bernoulli Naive Bayes and Section Multinomial Naive Bayes.

5.3.3 Tokenization

Tokenization describes the general process of breaking down a text corpus into individual elements that serve as input for various natural language processing algorithms. Usually, tokenization is accompanied by other optional processing steps, such as the removal of stop words and punctuation characters, stemming or lemmatizing, and the construction of n-grams. Below is an example of a simple but typical tokenization step that splits a sentence into individual words, removes punctuation, and converts all letters to lowercase.

Table 2. Example of tokenization.



5.3.4 Stop Words

Stop words are words that are particularly common in a text corpus and thus considered as rather un-informative (e.g., words such as so, and, or, the, …”). One approach to stop word removal is to search against a language-specific stop word dictionary. An alternative approach is to create a stop list by sorting all words in the entire text corpus by frequency. The stop list — after conversion into a set of non-redundant words — is then used to remove all those words from the input documents that are ranked among the top n words in this stop list.

Table 3. Example of stop word removal.

A swimmer likes swimming, thus he swims.

↓

swimmer	likes	swimming	,	swims	.

5.3.5 Stemming and Lemmatization

Stemming describes the process of transforming a word into its root form. The original stemming algorithm was developed my Martin F. Porter in 1979 and is hence known as Porter stemmer .

Table 4. Example of Porter Stemming.

A swimmer likes swimming, thus he swims.

↓

a	swimmer	like	swim	,	thu	he	swim	.

Stemming can create non-real words, such as “thu” in the example above. In contrast to stemming, lemmatization aims to obtain the canonical (grammatically correct) forms of the words, the so-called lemmas. Lemmatization is computationally more difficult and expensive than stemming, and in practice, both stemming and lemmatization have little impact on the performance of text classification .

Table 4. Example of Lemmatization.

A swimmer likes swimming, thus he swims.

↓

A	swimmer	like	swimming	,	thus	he	swim	.

5.3.6 N-Grams In the n-gram model, a token can be defined as a sequence of n items. The simplest case is the so-called unigram (1-gram) where each word consists of exactly one word, letter, or symbol. All previous examples were unigrams so far. Choosing the optimal number n depends on the language as well as the particular application.

- unigram (1-gram):

a	swimmer	likes	swimming	thus	he	swims

- bigram (2-gram):

a swimmer	swimmer likes	likes swimming	swimming thus	...

- trigram (3-gram):

a swimmer likes	swimmer likes swimming	likes swimming thus	...

5.3.7 Multi-variate Bernoulli Naive Bayes : The Multi-variate Bernoulli model is based on binary data: Every token in the feature vector of a document is associated with the value 1 or 0. The feature vector has *m*m dimensions where *m*m is the number of words in the whole vocabulary. The Bernoulli trials can be written as

P(x∣*ωj*)=∏*i*=1*mP*(*xi*∣*ωj*)*b*⋅(1−*P*(*xi*∣*ωj*))(1−*b*)(*b*∈0,1).P(x∣ωj)=∏i=1mP(xi∣ωj)b⋅(1−P(xi∣ωj))(1−b)(b∈0,1).

Let *P*^(*xi*∣*ωj*)P^(xi∣ωj) be the maximum-likelihood estimate that a particular word (or token) *xi*xi occurs in class *ωj*ωj. *P*^(*xi*∣*ωj*)=*dfxi*,*y*+1*dfy*+2P^(xi∣ωj)=dfxi,y+1dfy+2

where

- *dfxi*,*y*dfxi,y is the number of documents in the training dataset that contain the feature *xi*xi and belong to class *ωj*ωj.
- *dfy*dfy is the number of documents in the training dataset that belong to class *ωj*ωj.

5.4 Pilot Model: Sarcasm with Naive Bayes using TFIDF feature vectors

5.4.1 Multinomial Naive Bayes

Term Frequency

A alternative approach to characterize text documents — rather than binary values — is the term frequency (tf(t, d)). The term frequency is typically defined as the number of times a given term t (i.e., word or token) appears in a document d (this approach is sometimes also called raw frequency). In practice, the term frequency is often normalized by dividing the raw term frequency by the document length.

$$\text{normalized term frequency}=\text{tf}(t,d)nd\text{normalized term frequency}=\text{tf}(t,d)nd$$

where

- $\text{tf}(t,d)\text{tf}(t,d)$: Raw term frequency (the count of term tt in document dd).
- $ndnd$: The total number of terms in document dd.

The term frequencies can then be used to compute the maximum-likelihood estimate based on the training data to estimate the class-conditional probabilities in the multinomial model:

$$P\hat{}(xi \mid \omega j)=\sum tf(xi,d\in\omega j)+\alpha\sum Nd\in\omega j+\alpha\cdot VP\hat{}(xi \mid \omega j)=\sum tf(xi,d\in\omega j)+\alpha\sum Nd\in\omega j+\alpha\cdot V$$

where

- $xixi$: A word from the feature vector xx of a particular sample.
- $\sum tf(xi,d\in\omega j)\sum tf(xi,d\in\omega j)$: The sum of raw term frequencies of word xixi from all documents in the training sample that belong to class ωjωj.
- $\sum Nd\in\omega j\sum Nd\in\omega j$: The sum of all term frequencies in the training dataset for class ωjωj.
- $\alpha\alpha$: An additive smoothing parameter ($\alpha=1\alpha=1$ for Laplace smoothing).
- VV: The size of the vocabulary (number of different words in the training set).

The class-conditional probability of encountering the text xx can be calculated as the product from the likelihoods of the individual words (under the naive assumption of conditional independence).

$$P(x|\omega j)=P(x1|\omega j)\cdot P(x2|\omega j)\cdot \ldots \cdot P(xn|\omega j)=\prod i=1mP(xi|\omega j)P(x|\omega j)=P(x1|\omega j)\cdot P(x2|\omega j)\cdot \ldots \cdot P(xn|\omega j)=\prod i=1mP(xi|\omega j)$$

5.4.2 Term Frequency - Inverse Document Frequency (Tf-idf)

The term frequency - inverse document frequency (Tf-idf) is another alternative for characterizing text documents. It can be understood as a weighted term frequency, which is especially useful if stop words have not been removed from the text corpus. The Tf-idf approach assumes that the importance of a word is inversely proportional to how often it occurs across all documents. Although Tf-idf is most commonly used to rank documents by relevance in different text mining tasks, such as page ranking by search engines, it can also be applied to text classification via naive Bayes.

$$\text{Tf-idf}=\text{tfn}(t,d)\cdot \text{idf}(t)\text{Tf-idf}=\text{tfn}(t,d)\cdot \text{idf}(t)$$

Let tfn(d,f)tfn(d,f) be the normalized term frequency, and idfidf, the inverse document frequency, which can be calculated as follows

$$\text{idf}(t)=\log(\text{ndnd}(t)),\text{idf}(t)=\log(\text{ndnd}(t)),$$

where

- *nd*nd: The total number of documents.
- *nd*(*t*)nd(t): The number of documents that contain the term *t*t.

5.4.3 Performances of the Multi-variate Bernoulli and Multinomial Model

Empirical comparisons provide evidence that the multinomial model tends to outperform the multi-variate Bernoulli model if the vocabulary size is relatively large . However, the performance of machine learning algorithms is highly dependent on the appropriate choice of features. In the case of naive Bayes classifiers and text classification, large differences in performance can be attributed to the choices of stop word removal, stemming, and token-length. In practice, it is recommended that the choice between a multi-variate Bernoulli or multinomial model for text classification.

6.DESIGN

6.1 Structure Chart

Structure chart in software engineering and organizational theory, is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules in a tree. Each module is represented by a box, which contains the module's name. A structure chart is top-down modular design tool, constructed of squares representing the different modules in the system, and lines that connect them. The lines represent the connection and or ownership between activities and sub-activities as they are used in organization chart.

Structure chart of this project: Each module is represented by a box, which contains the module's name. A structure chart is top-down modular design tool, constructed of squares representing the different modules in the system, and lines that connect them. The lines represent the connection and or ownership between activities and sub-activities as they are used in organization chart.

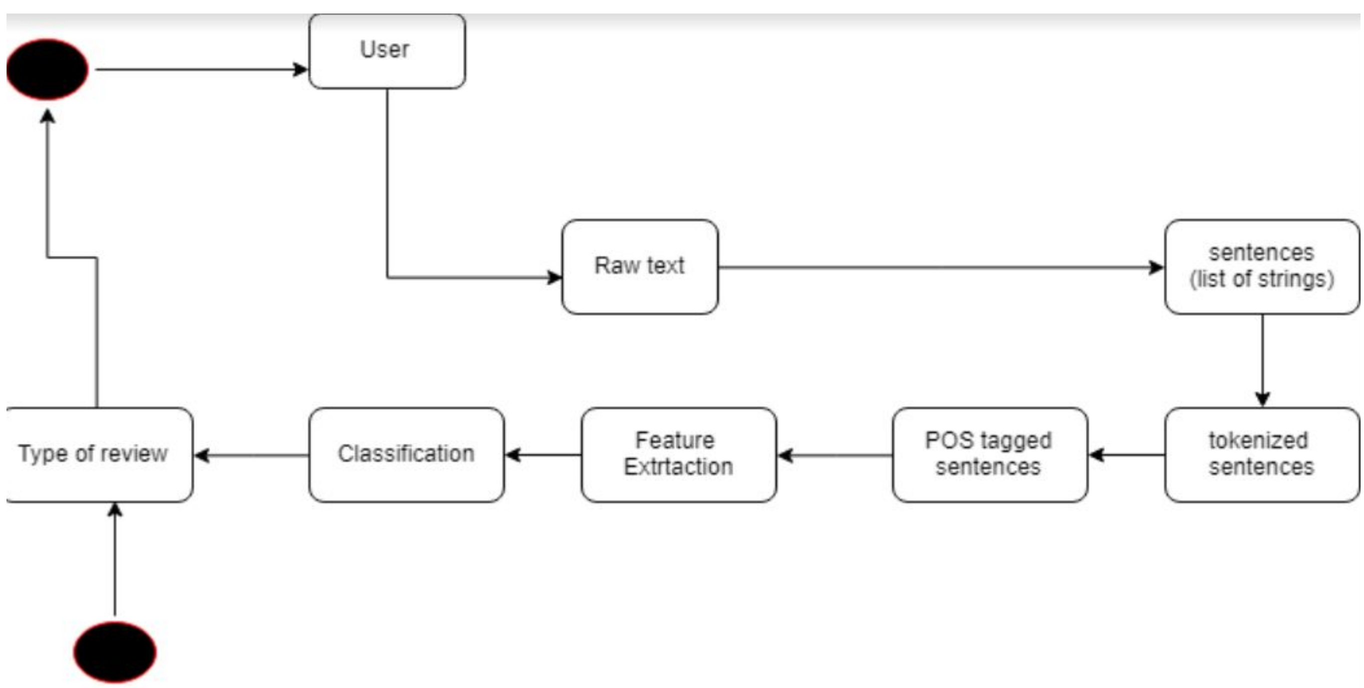


Fig 6.1 Structure chart

Description A raw text consisting of code-mixed sentences is given as the input. Sentence segmentation will segment the sentences from the raw text and gives it for Tokenizer. The tokenizer will create a list of list of strings and gives it to the PoS tagger. The PoS tagger will tag all the words based upon their parts of speech in a list of list of tuples and gives it for the feature extractor , which ives the sentences with features for classification , the classifier gives the review of the sentence as output to the user.

6.2 UML DIAGRAMS

The uml is a language. It provides vocabulary and the results for combining words in that vocabulary for the purpose of communication. A modelling language is language whose vocabulary and rules flows on the conceptual and physical representation of a system. A modelling language such as uml is a standard language for software blue prints.

The uml is a language for visualizing, specifying, constructing and documenting. The software intensive articrafts of a system.

UML diagram are classified into two categories:

1. Structural or static

2. Dynamic or behavioural

Structural Model contains

Classes, object, use case, component and deployment.

Behavioural Model contains:

Collaboration, State chart and activity.

6.2.1 Sequence Diagram:

Interaction diagram is called is sequence diagram. Interaction diagram describes patterns of communication among a set of interaction objects. An object interacts with another object by sending messages, arguments may be passed along with a message and they are found to be parameters of executing methods in the receiving objects.

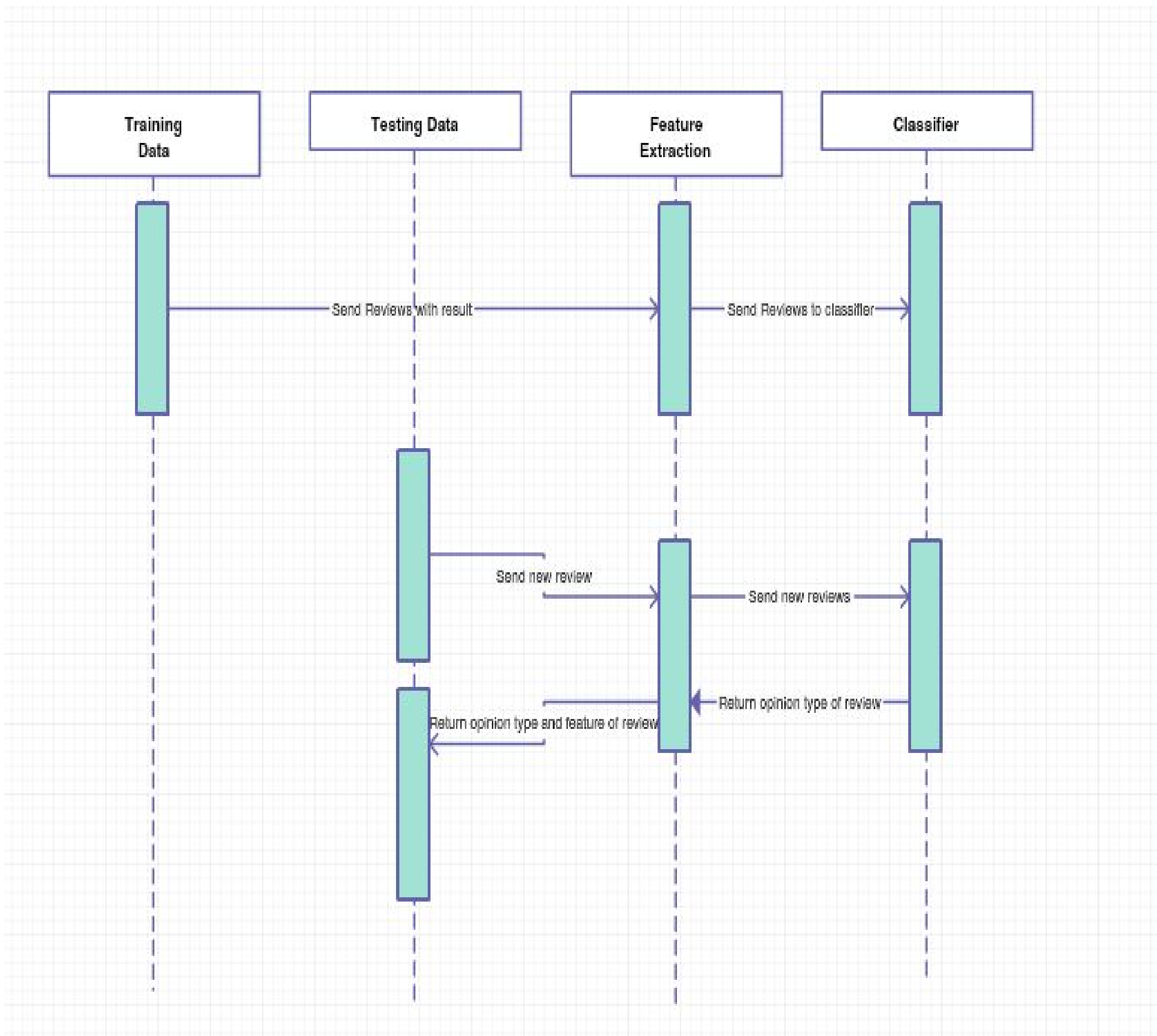


Fig 6.2.1 Sequence Diagram

6.2.2 Activity Diagram

An Activity diagram describes the work flow in steps between activities and actions with support for interaction and concurrency behaviour of the system. These are similar to state diagram because activities of the state of doing something. These can show activities, that are conditional or parallel.

The Objects identified are Hard Disk, Main Memory, Processor.

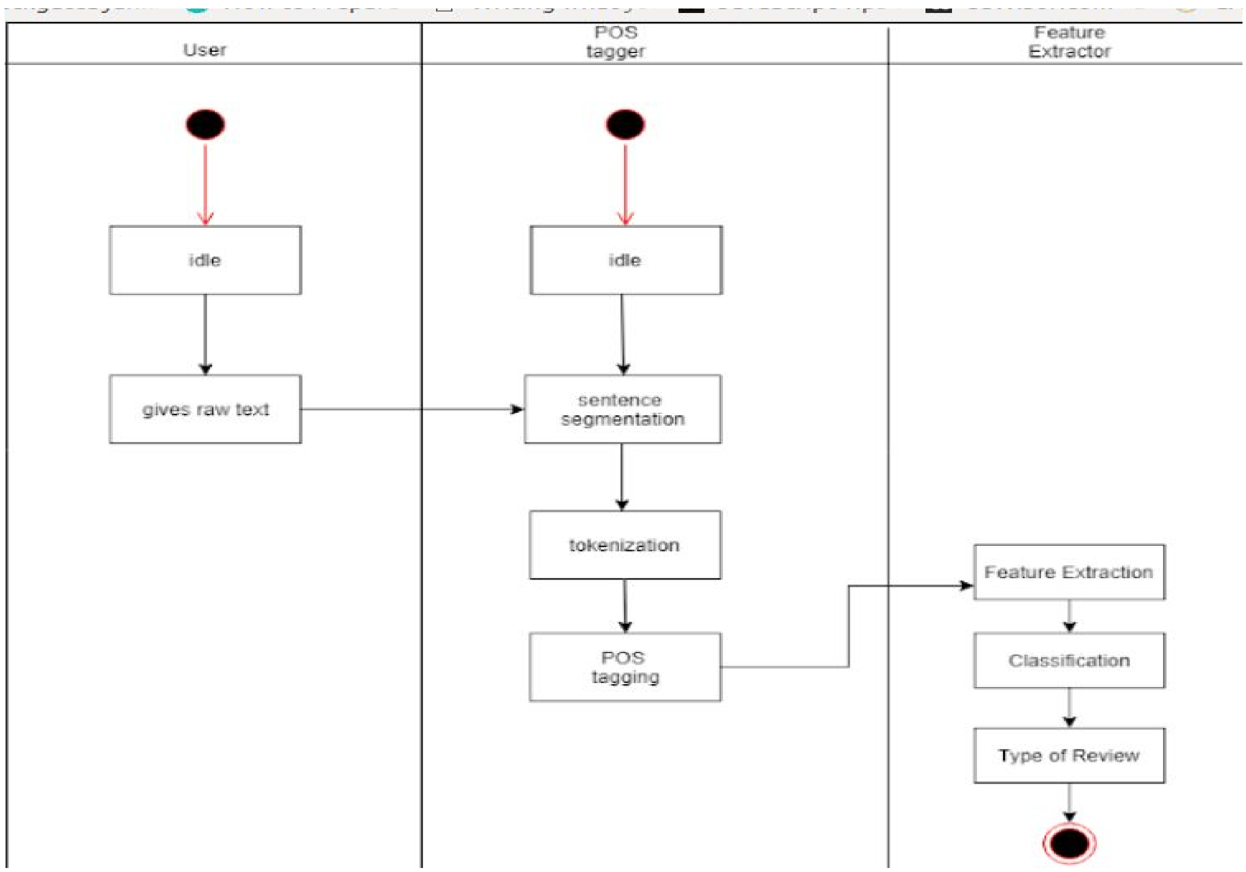


Fig 6.2.2 Activity Diagram

6.2.3 State Chart Diagram

It describes the dynamic nature of the diagram how the objects changes its state. It has various States and Transitions.

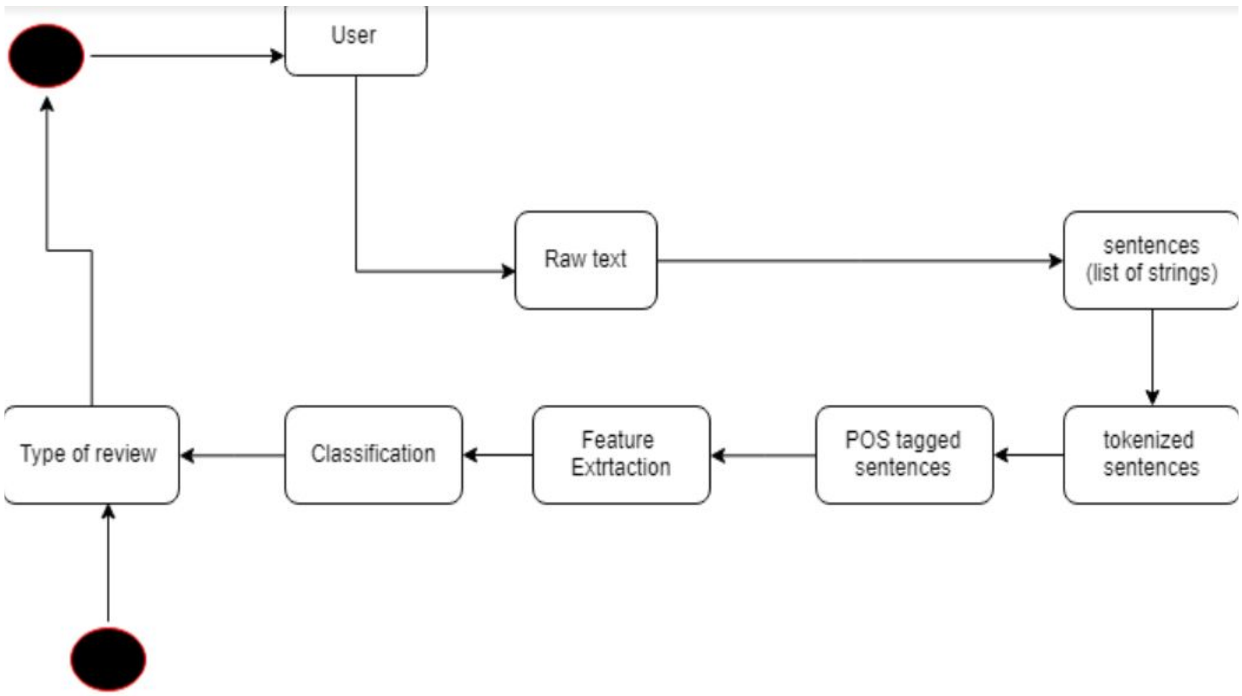


Fig 6.2.3 State Chart Diagram

6.2.4 Deployment Diagram

It provides different perspective of the application. It captures the configuration of runtime elements of the application. This diagram is more useful when a system is build and ready to be deployed.

NODES ARE:

1.User

2.POS Tagger

3.I/O function

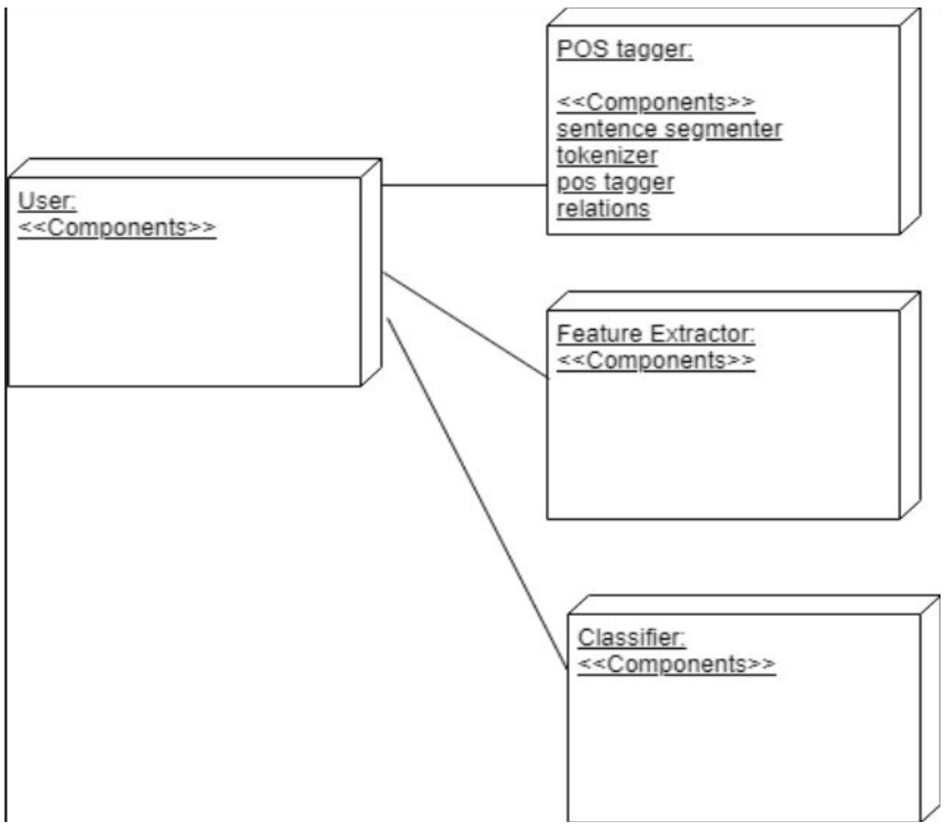


Fig 6.2.4 Deployment Diagram

7.METHODOLOGY

7.1 Supervised Learning

Remember the time when you used to go to school? The time when you first learnt what an apple looked like? The teacher probably showed a picture of an apple and told you what it was, right? And you could identify the particular fruit ever since then.That’s exactly how supervised learning works. As you can see in the image below:

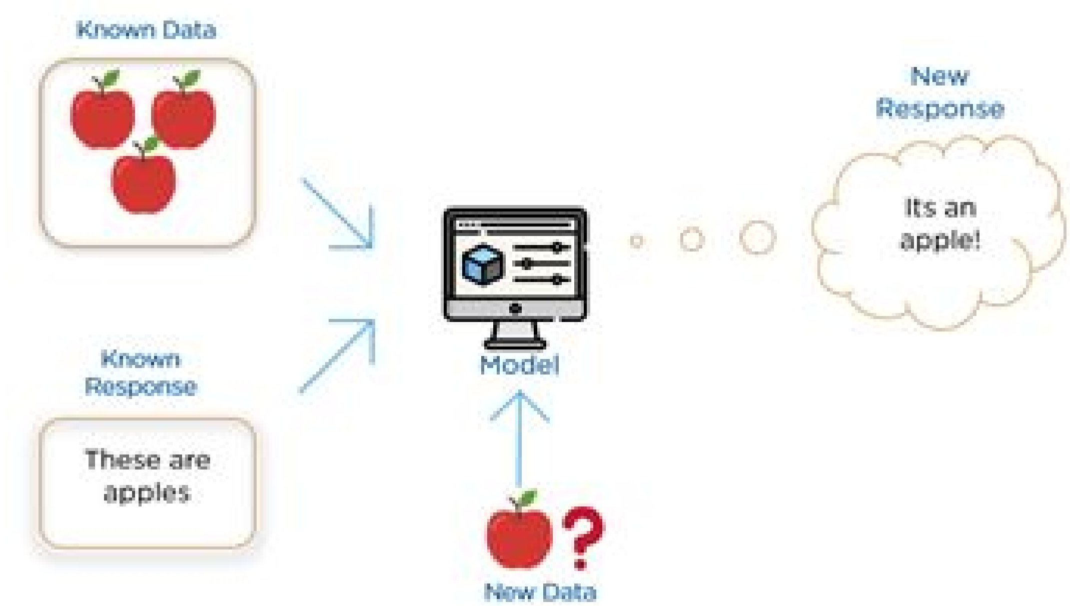


Fig 7.1 Supervised learning example

Step 1 -You provide the system with data that contains photos of apples and let it know that these are apples. This is called labelled data.

Step 2-The model learns from the labelled data and the next time you ask it to identify an apple, it can do it easily

7.2 **Unsupervised Learning**

If somebody gives you a basket full of different fruits and asks you to separate them, you will probably do it based on their colors, shape and size, right .*Unsupervised learning works in the same way.* As you can see in the image

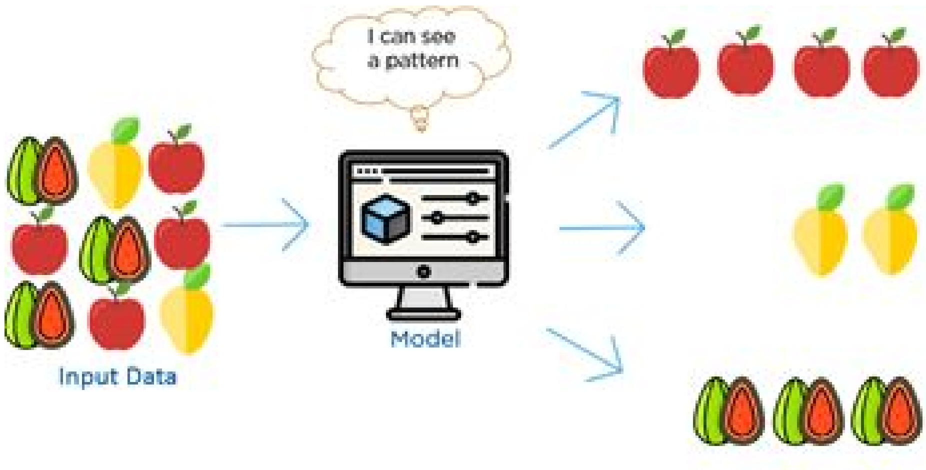


Fig 7.2 Unsupervised learning example

Step1-You provide the system with a data that contains photos of different kinds of fruits and ask it to segregate it. Remember, in case of unsupervised learning you don't need to provide labelled data.

Step 2-The system will look for patterns in the data. Patterns like shape, color and size and group the fruits based on those attributes.

1. Type of input data –In case of Supervised Learning, the input data is labelled and in case of Unsupervised Learning, the input data is non labelled.

2. Feedback –In case of Supervised Learning, the system learns from the output and keeps it in mind while in case of unsupervised learning, there is no feedback involved.

3. Function –Supervised Learning is generally used to predict data whereas, Unsupervised Learning is used to find hidden structure in the data.

8.IMPLEMENTATION

8.1 Sample Code for Sentiment Classification:

```
from msg import *

import os

import csv

import re

import nltk

import nltk.classify.util

from nltk.classify import NaiveBayesClassifier

from nltk.corpus import movie_reviews

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

from nltk.corpus import wordnet as wn

from nltk.corpus import sentiwordnet as swn

from preprocessing import *

from pos_tagging import *

from negation import NegatingWordReader

from modifier import ModifierWordReader

nltk.download('stopwords')

nltk.download('movie_reviews')
```

```
def create_word_features(words):

useful_words = [word for word in words if word not in stopwords.words("english")]

my_dict = dict([(word, True) for word in useful_words])

return my_dict

neg_reviews = []

for fileid in movie_reviews.fileids('neg'):

        words = movie_reviews.words(fileid)

        neg_reviews.append((create_word_features(words), "negative"))

print(len(neg_reviews))

pos_reviews = []

for fileid in movie_reviews.fileids('pos'):

words = movie_reviews.words(fileid)

pos_reviews.append((create_word_features(words), "positive"))

print(len(pos_reviews))



tweets = list()

f = open('testdata.manual.2009.06.14.csv','rt')

concat = ''

try:

reader = csv.reader(f)
```

```
for row in reader:

concat += row[5]

tweets.append(row)

finally:

f.close()

def is_adjective(tag):

if tag == 'JJ' or tag == 'JJR' or tag == 'JJS':

return True

else:

return False

def is_adverb(tag):

if tag == 'RB' or tag == 'RBR' or tag == 'RBS':

return True

else:

return False

def is_noun(tag):

if tag == 'NN' or tag == 'NNS' or tag == 'NNP' or tag == 'NNPS':

return True

else:

return False
```

```
def is_verb(tag):

if tag == 'VB' or tag == 'VBD' or tag == 'VBG' or tag == 'VBN' or tag == 'VBP' or tag == 'VBZ':

return True

else:

return False

def is_valid(token):

if is_noun(token[1]) or is_adverb(token[1]) or is_verb(token[1]) or is_adjective(token[1]):

return True

else:

return False

nwr = NegatingWordReader('NegatingWordList.txt')

mwr = ModifierWordReader('BoosterWordList.txt')

def filter_tweet(tweet):

return map(lambda x : x[0], filter(lambda token : is_valid(token), tweet))

def get_sentiment_from_level(i):

if i == 4:

return 'Positive'

elif i == 2:

return 'Neutral'
```

```
else:

return 'Negative'

def get_first_synset(word):

synsets = swn.senti_synsets(word)

if len(synsets) > 0:

return synsets[0]


else:

return None


def get_synsets(tweet):

return filter(lambda x: x is not None ,map(lambda x : get_first_synset(x),tweet))


def get_posScore_from_synsets(sentisynsets):

scores = map(lambda sentisynset: sentisynset.pos_score(), sentisynsets)

if len(scores) > 0:

return reduce(lambda a,x: a + x, scores)

else:

return 0

def get_negScore_from_synsets(sentisynsets):
```

```
scores = map(lambda sentisynset: sentisynset.neg_score(), sentisynsets)
if len(scores) > 0:
return reduce(lambda a,x: a + x, scores)
else:
return 0
def get_tweet_sentiment_from_score(posScore, negScore):
if posScore > negScore:
return 'Positive'
elif posScore == negScore:
return 'Neutral'
else:
return 'Negative'


def get_sentiment_from_tweet(tweet):
tweet = filter_tweet(tweet)
sentisynsets = get_synsets(tweet)
posScore = get_posScore_from_synsets(sentisynsets)
negScore = get_negScore_from_synsets(sentisynsets)

sentiment = get_tweet_sentiment_from_score(posScore, negScore)
```

```
return posScore, negScore, sentiment

tweets_tagged = map(lambda tweet: pos_tagging(preprocess(tweet,dicoSlang)), tweets)
real_sentiments = map(lambda tweet: get_sentiment_from_level(int(tweet[0])),tweets)
predicted_sentiments = map(lambda tweet: get_sentiment_from_tweet(tweet)[2], tweets_tagged)

train_set = neg_reviews[:750] + pos_reviews[:750]
test_set = neg_reviews[750:] + pos_reviews[750:]
print(len(train_set), len(test_set))
classifier = NaiveBayesClassifier.train(train_set)
accuracy = nltk.classify.util.accuracy(classifier, test_set)
print(accuracy * 100)
while True:
print 'enter a sentence'
user_input = raw_input()
if user_input == "":
break
words = word_tokenize(user_input)
words = create_word_features(words)
```

```
print(classifier.classify(words))
```

8.2 **Sample Code Sarcasm**

```
import numpy as np
import csv
import re
def preprocessing(csv_file_object):
data=[]
length=[]
remove_hashtags = re.compile(r'#\w+\s?')
remove_friendtag = re.compile(r'@\w+\s?')
remove_sarcasm = re.compile(re.escape('sarcasm'),re.IGNORECASE)
remove_sarcastic = re.compile(re.escape('sarcastic'),re.IGNORECASE)
for row in csv_file_object:
if len(row[0:])==1:
temp=row[0:][0]
temp=remove_hashtags.sub('',temp)
if len(temp)>0 and 'http' not in temp and temp[0]!='@' and '\u' not in temp:
temp=remove_friendtag.sub('',temp)
temp=remove_sarcasm.sub('',temp)
temp=remove_sarcastic.sub('',temp)
temp=' '.join(temp.split()) #remove useless space
if len(temp.split())>2:
data.append(temp)
length.append(len(temp.split()))
```

```
data=list(set(data)) #remove duplicate tweets
data = np.array(data)
return data, length
print 'Extracting data'
### POSITIVE DATA ####
csv_file_object_pos = csv.reader(open('twitDB_sarcasm.csv', 'rU'),delimiter='\n')
pos_data, length_pos = preprocessing(csv_file_object_pos)
#print pos_data
### NEGATIVE DATA ####
csv_file_object_neg = csv.reader(open('twitDB_regular.csv', 'rU'),delimiter='\n')
neg_data, length_neg = preprocessing(csv_file_object_neg)
print 'Number of sarcastic tweets :', len(pos_data)
print 'Average length of sarcastic tweets :', np.mean(length_pos)
print 'Number of non-sarcastic tweets :', len(neg_data)
print 'Average length of non-sarcastic tweets :', np.mean(length_neg)
#save the tweets as binary .npy files
np.save('posproc',pos_data)
np.save('negproc',neg_data)
```

8.3 Sample Code for Plotting results from multinomial and Bernouli NB

```
'''BernoulliNB gave slightly better results than MultinomialNB on just TF-IDF feature vector.'''

import numpy as np

#Load the binary files of sarcastic and non-sarcastic tweets

sarcasm=np.load("posproc.npy")

neutral=np.load("negproc.npy")

#Print sample data

print ("10 sample sarcastic lines:")

print (sarcasm[:10])

print ("10 sample non-sarcastic lines:")

print (neutral[:10])

#Stats

sarcasm_size=len(sarcasm)

print ("Total sarcastic lines = "+str(sarcasm_size))

neutral_size=len(neutral)

print ("Total non-sarcastic lines = "+str(neutral_size))


#Import necessary libraries

from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.decomposition import TruncatedSVD
```

```
from sklearn.pipeline import Pipeline

import matplotlib.pyplot as plt


dataset_pos = sarcasm

dataset_neg = neutral

print ("Total length of dataset = "+str(len(dataset_pos)+len(dataset_neg)))


##Plotting data using LSI(SVD) since PCA doesn't support sparse data.


pipeline = Pipeline([

('vect', CountVectorizer()),

('tfidf', TfidfTransformer()),

])

X = pipeline.fit_transform(dataset_pos)

Y = pipeline.fit_transform(dataset_neg)


data2DX = TruncatedSVD(n_components=2).fit_transform(X)

data2DY = TruncatedSVD(n_components=2).fit_transform(Y)


#Red - Sarcastic, Green - Regular.
```

```
plt.scatter(data2DX[:,0], data2DX[:,1], c=np.array([1, 0, 0]))

plt.scatter(data2DY[:,0], data2DY[:,1], c=np.array([0, 1, 0]))

plt.show() #not required if using ipython notebook
```

For preprocessing the data

```
import numpy as np

import csv

import re

def preprocessing(csv_file_object):

    data=[]

    length=[]

    remove_hashtags = re.compile(r'#\w+\s?')

    remove_friendtag = re.compile(r'@\w+\s?')

    remove_sarcasm = re.compile(re.escape('sarcasm'),re.IGNORECASE)

    remove_sarcastic = re.compile(re.escape('sarcastic'),re.IGNORECASE)


    for row in csv_file_object:

        if len(row[0:])==1:

            temp=row[0:][0]

            temp=remove_hashtags.sub('',temp)

            if len(temp)>0 and 'http' not in temp and temp[0]!='@' and '\u' not in temp:
```

```
            temp=remove_friendtag.sub('',temp)
            temp=remove_sarcasm.sub('',temp)
            temp=remove_sarcastic.sub('',temp)
            temp=' '.join(temp.split())
            if len(temp.split())>2:
                data.append(temp)
                length.append(len(temp.split()))
    data=list(set(data))
    data = np.array(data)
    return data, length
print 'Extracting data'
csv_file_object_pos = csv.reader(open('twitDB_sarcasm.csv', 'rU'),delimiter='\n')
pos_data, length_pos = preprocessing(csv_file_object_pos)
csv_file_object_neg = csv.reader(open('twitDB_regular.csv', 'rU'),delimiter='\n')
neg_data, length_neg = preprocessing(csv_file_object_neg)
print 'Number of  sarcastic tweets :', len(pos_data)
print 'Average length of sarcastic tweets :', np.mean(length_pos)
print 'Number of  non-sarcastic tweets :', len(neg_data)
print 'Average length of non-sarcastic tweets :', np.mean(length_neg)
```

```
np.save('posproc',pos_data)

np.save('negproc',neg_data)
```

For Sentiment analysis :

Sample Input:

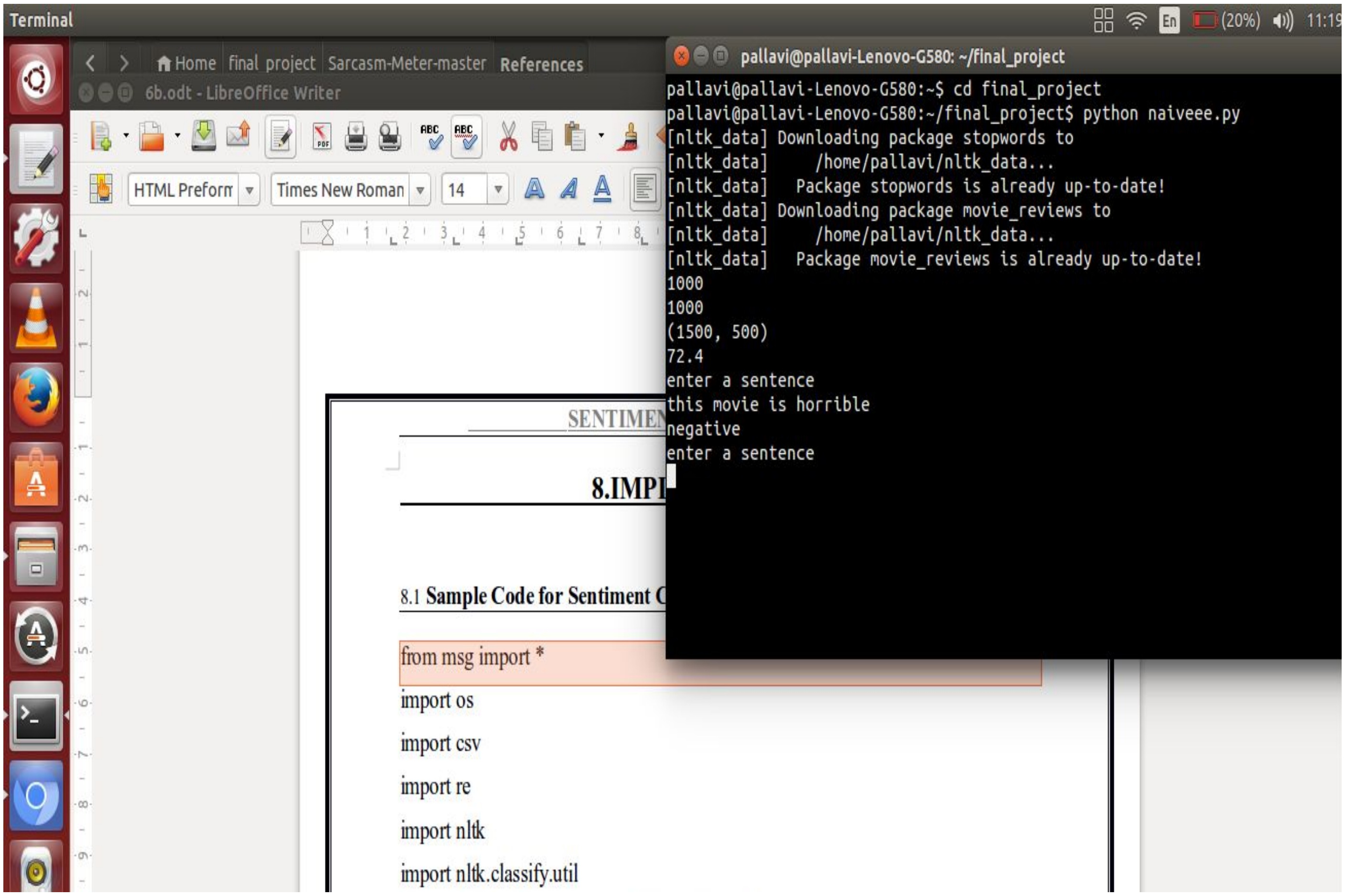
User Input of a movie review

Sample Output:

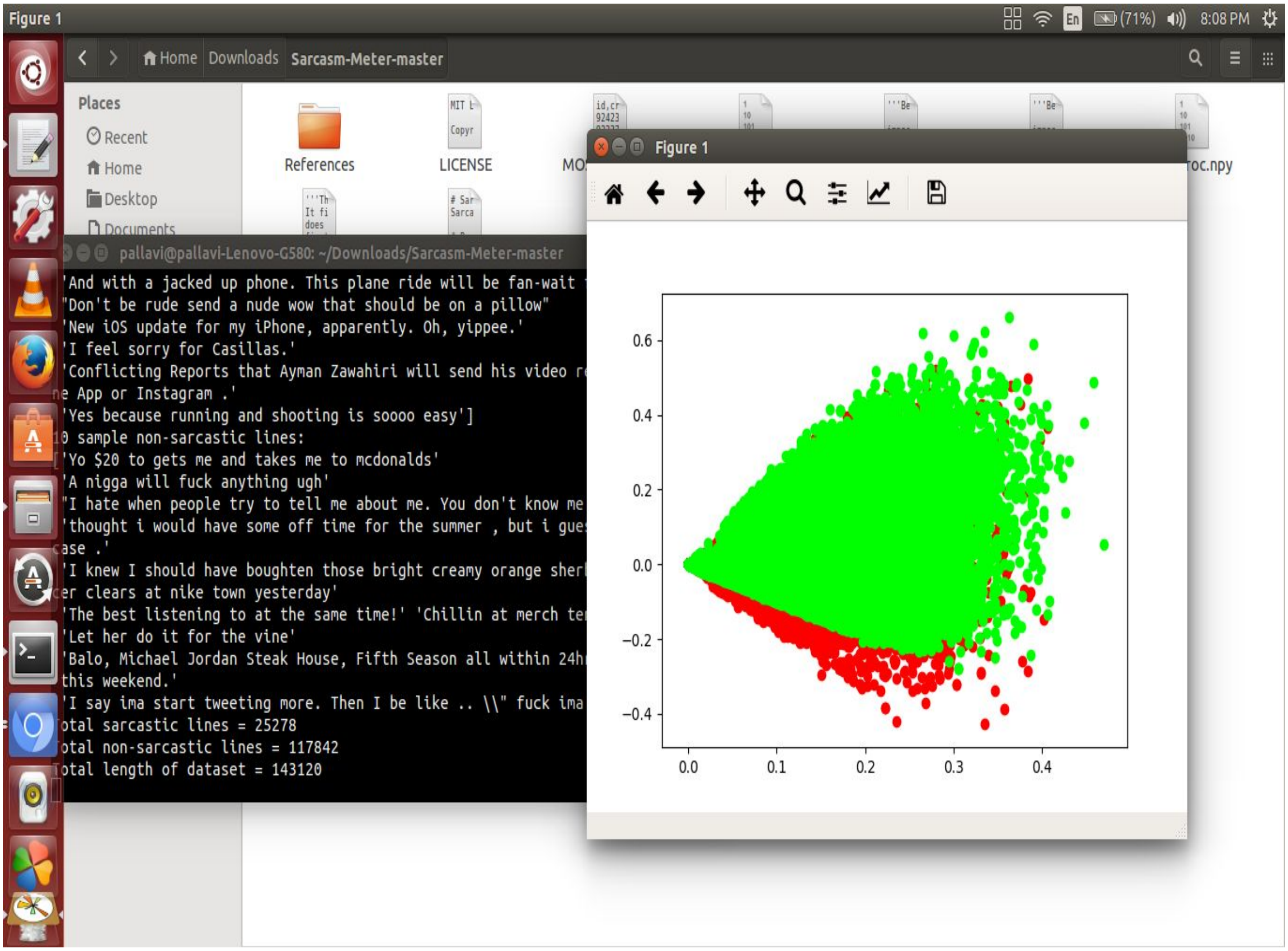
Sentiment of the entered review

9.RESULTS

9.1 Capturing Starts , Sentiment analysis of a sentence or a paragraph.



9.2 YTPilot model: Naive Bayes with TFIDF feature vectors



10. TESTING

10.1 Testing model with positive reviews

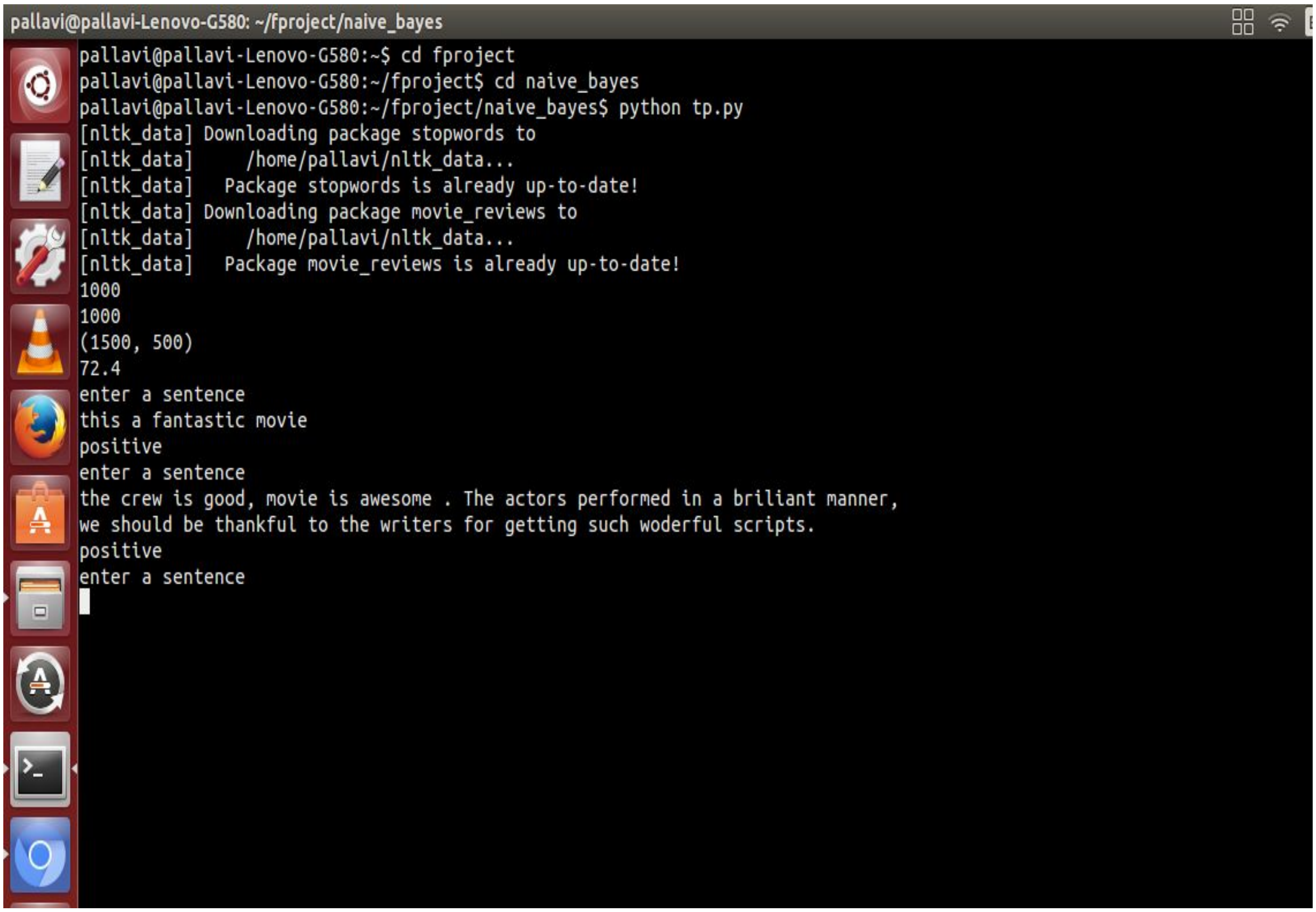


Fig 10.1 Testing for Positive review

10.2 Testing model with negative reviews

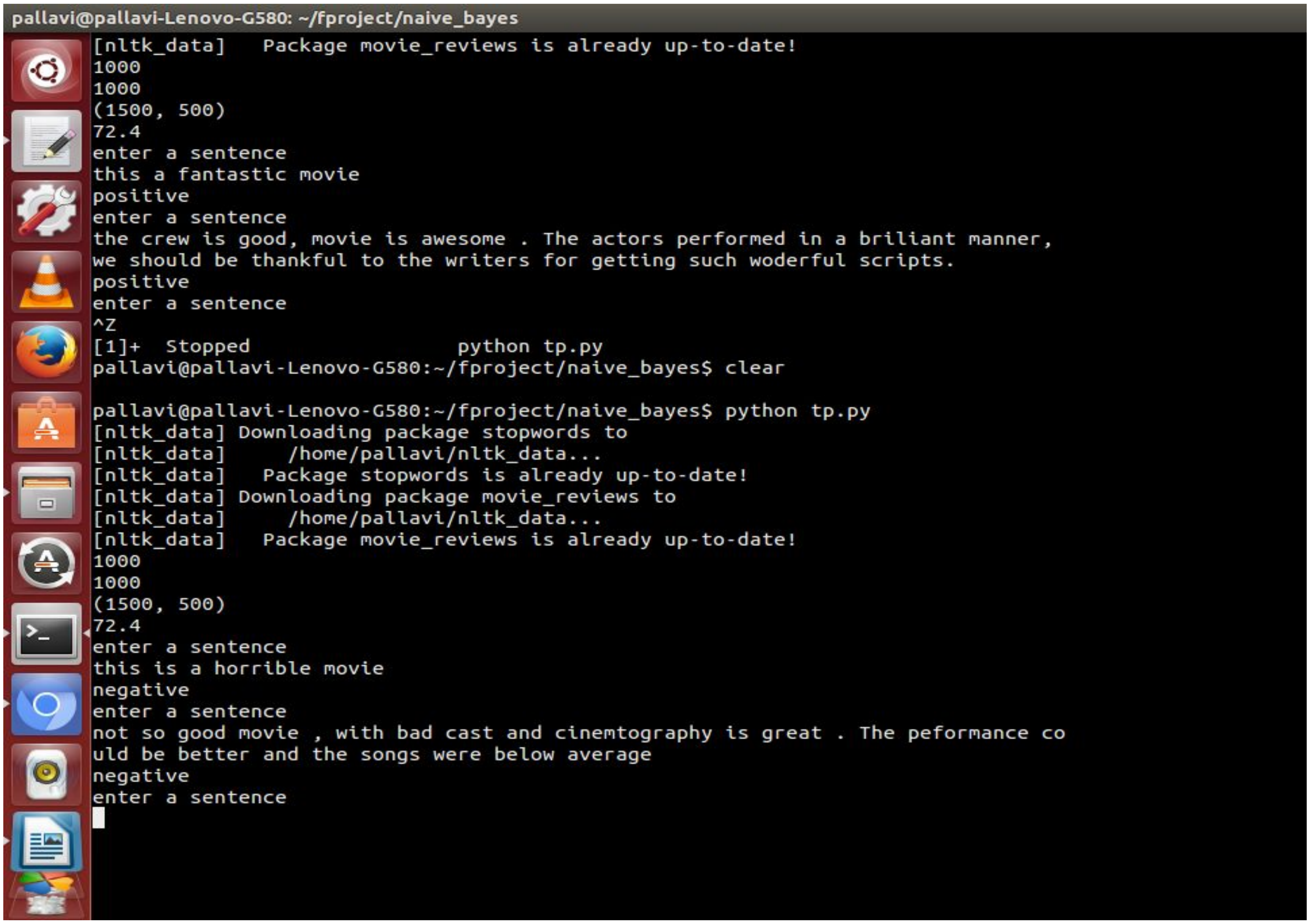


Fig 10.2 Testing the model for negative review

11.CONCLUSION

In this, we proposed a model with a classifier which is domain specific for the task of Sentiment analysis on movie reviews. We developed a supervised system in order to tackle the problem effectively. The proposed system is able to successfully assign sentiment to the user given movie review. We also developed a pilot model to distinguish the accuracies between Multinomial and Bernoulli Naive Bayes for sarcastic and non sarcastic reviews.Experiments show that our system is quite generic that shows encouraging performance levels.We have seen that Sentiment Analysis can be used for analyzing opinions in blogs, articles, Product reviews, Social Media websites, Movie-review websites where a third person narrates his views. We also studied NLP and Machine Learning approaches for Sentiment Analysis. We have seen that is easy to implement Sentiment Analysis via SentiWordNet approach than via Classier approach. We have seen that sentiment analysis has many applications and it is important field to study. Sentiment analysis has Strong commercial interest because Companies want to know how their products are being perceived and also Prospective consumers want to know what existing users think.

REFERENCES

WEB SITES:

1.http://pythonforengineers.com/build-a-sentiment-analysis-app-with-movie-reviews/

2.https://www.programiz.com/python-programming/methods/built-in/map

3.http://vpython.org/contents/docs/vector.html

4.http://www.nltk.org/book/ch07.html

5.http://www.nltk.org/book/ch01.html

6.https://link.springer.com/article/10.1023/A:1007673816718

7.https://en.wikipedia.org/wiki/Metaphone

8.https://taku910.github.io/crfpp/

BOOKS

- NLTK Book by Steven Bird textbook.
- Think Python by Allen.B.Downey 2nd edition.
- Machine Learning by Vinod Chandra 2nd edition.