SENTIMENT MINING IN TWITTER USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

SHANTANU GARG [Reg No: 1031310362]
SHARAD KAKRAN [Reg No: 1031310379]

Under the guidance of

Dr. S. THENMALAR, Ph.D
(Assistant Professor(O.G), Department of Computer Sciene & Engineering)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Kancheepuram District

May 2017

SRM UNIVERSITY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**SENTIMENT MINING IN TWITTER USING MACHINE LEARNING** " is the bonafide work of " **SHANTANU GARG [Reg No: 1031310362], SHARAD KAKRAN [Reg No: 1031310379]**", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. S. THENMALAR, Ph.D
GUIDE
Assistant Professor(O.G)
Dept. of Computer Sciene & Engineering

SIGNATURE

Dr. B.Amutha
HEAD OF THE DEPARTMENT
Dept. of Computer Science & Engineering

Signature of the Internal Examiner

Signature of the External Examiner

ABSTRACT

With a rapid increase in the outreach of the internet today, people have taken the internet as a means to express their opinion on topics ranging from their preferred political party to their favourite restaurant. Social media websites such as twitter, Facebook are such places where people can express themselves. Twitter generates a vast amount of sentiment rich data which can be studied in order to identify the effects of sentiment classification. This information can be used to understand the perception of twitter users on various subjects. In this project, we use machine learning techniques such as Latent Semantic Analysis (LSA),Naive Bayes (NB),Support Vector Machines (SVM),K- Nearest Neighbors (KNN) to analyse twitter posts on various subjects. Sentiment mining deals with identifying and extracting the sentiment content of a text unit using Natural Langauge Processing (NLP), Statistics or machine learning techniques. We present a new feature for classifying the tweets in five different moods - happy, sad, fun, love and hate.

ACKNOWLEDGEMENTS

We would like to acknowledge the crucial role of the following people who helped and supported us in the preparation of this project. We would like to thank our Director **Dr C. MUTHAMIZHCELVAN**, Faculty of Engineering and Technology, SRM University, Chennai.

We would like to thank , **Dr. C.Muthamizhcelvan**, Director, Faculty of Engineering and Technology, SRM University, Chennai for providing us an opportunity to do this project under E&T department.

We would like to express our deep sense of gratitude to **Dr. B.Amutha**, Professor and Head of Department of Computer Science and Engineering, SRM University, Chennai, for giving us the opportunity to take up this project.

We would also like to thank our project co-ordinator, **Dr D. MALATHI**, Professor, Department of Computer Science and Engineering, SRM University, Chennai, for her valuable inputs during the project reviews.

We take immense pleasure in conveying our thanks to our panel members **Mr R.SRINIVASAN**, **Dr A. PANDIAN**, **Mrs SINDHU C.**, **Mrs S. INIYAN**, **Dr SUBALALITHAA C. N.**, **Mrs R. ANITHA**, Department of Computer Science and Engineering, SRM University, Chennai, for their invaluable inputs and comments during the reviews.

We take immense pleasure in conveying our thanks to our guide **Dr S. THENMALAR** Asst. Professor, Department of Computer Science and Engineering, SRMUniversity, Chennai, for her immense support and cooperation at every stage of the project completion.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

NB Naive Bayes

SVM Support Vector Machines

KNN K- Nearest Neighbors

NLP Neuro Linguistic Programming

LSA Latent Semantic Analysis

NLP Natural Langauge Processing

TDM Term-Frequency Document Matrix

DTM Document-Term Matrix

SVD Singular Value Decomposition

URL Uniform Resource Locator

CHAPTER 1

INTRODUCTION

1.1 SYSTEM ANALYSIS

System Analysis is a combined process dissecting the system responsibilities that are based on the problem domain characteristics and user requirements. The goal of the system analysis is to completely specify the technical details for the main concept in a concise and unambiguous manner.

1.2 PROBLEM DEFINITION

The problem definition can be defined as identifying the solution to efficiently support the education system of the society. To enhance and create an effective, efficient and intelligent virtual classroom which could help the students to gain more knowledge.

1.3 SYSTEM OVERVIEW

The purpose of the system analysis is to produce the brief analysis task and also to establish complete information about the concept, behaviour and other constraints such as performance measure and system optimization.

1.4 OBJECTIVE

The objective of the project are:-

- Making use of the contexts present in the data
- Finding latent patterns in the data
- Reducing the feature set

1.5 SCOPE

Datasets contain tweets which are downloaded from twitter. This dataset is divided into two sections, random tweets and context based tweets.
Rather than using all n-grams as features for modelling if use statistical technique called chi-square test for selecting the top 8000 bigrams and trigrams. This will reduce the feature set and noise present in dataset and hence increase the accuracy of the model.
LSA is used to uncover the latent patterns present in dataset. It take out the words used in similar context and allocate them some space which helps to segregate the words based on the form they are used in.

1.6 EXISTING SYSTEM

In the recent years, new modes of communication, such as text messaging and microblogging have risen and become omnipresent. While there are no boundaries to the variety of information communicated by texts and tweets, frequently these short messages are used to share sentiments and opinions that people have about the happenings in the world around them.

These informal text genres present difficulties for natural language processing exceeding those typically confronted while working with more traditional text genres.

Tweets and texts are short. The language used is very casual, with creative spelling and punctuation, slang, misspellings, new words, genre-specific terminology and abbreviations and Uniform Resource Locator (URL).

There has been very limited research done in Handling such challenges so as to automatically mine and understand the sentiments and opinions that people are communicating.

In the sentimental analysis, sentences are usually classified into three categories: - negative, positive and neutral. In this method, the polarity of a sentence is calculated, and then it is assigned to different classes accordingly. However, due to a lack of clarity often the sentences are misclassified thus affecting the overall accuracy.

The existing model uses the tweets for training models without making use of latent meaning of words in it.
In the system all features are used without performing any reduction technique which leads to poor performance of model.
The model has been enhanced and the best features are selected which are used for training a machine learning model and shows better performance .

The existing model fails to avoid the curse of dimensionality and to capture the hidden semantic in the data.

1.7 PROPOSED SYSTEM

In this project to obtain a better accuracy, we classify the tweets based on the sentiments of the writer behind them. The different emotions explored in the scope of this paper are: Happy, Sad, Fun, Love and Hate.

To reduce the adverse effect of the informal language on the test results several pre-processing techniques such as Data Cleaning, Data Splitting, Feature Selection and Feature Extraction are performed on the dataset.

Also, we study the impact of various algorithms such as LSA, SVM, KNN and NB on the accuracy of the model.

We further divide our research into two stages. In the first stage, we examine the effect of classification on unstructured tweets which were not segregated based on context. Moreover, in the second stage, we study the impact of classification techniques on context based tweets.

- We focus on using contexts of the tweets
- Finding latent patterns in data
- Reducing the size of the feature set

1.8 APPLICATIONS

Sentimental Analysis has a lot of practical applications. It can be used by the media industry to learn about the likings and dislikings of people in a movie or tv show by going through the reviews posted online. In the field of investment, Sentiment Mining can help with predicting the stock market by evaluating the public perception of stock in various forums. It can also be used by politicians to go through the review of their speeches on Twitter and analyse the positive and negative aspects of it. Sentiment Mining can be further assisted with deep learning. Systems can be developed which present a live analysis of the various scenarios in real time.

CHAPTER 2

LITERATURE SURVEY

The sentimental analysis is an emerging field of research. Today, where we have trillions of gigabytes of worth of data on the internet in the form of blogs, status updates, tweets, etc., sentiment analysis can help us in establishing and further understanding the pattern between this data. Sentiment mining involves classifying the presence of positive and negative emotions in a text document. In this model, our approach goes beyond previous work in that our model further tries to classify the mood of the author at the time of the writing. Segregating Twitter posts according to the various moods can help us in predicting the outcomes of different events like the performance of stock market or the result of elections. Also, this methodology can be used by companies and corporations in an attempt to better understand their public perception helping them save a lot of money by allowing them to focus on their marketing and communication effects accordingly [1] [2]. Sentiment analysis can be conducted on various levels ranging from parse level to fine level. In the scope of this paper, we use sentence-level sentiment mining which falls somewhere in between the parsing level and fine level [3]. Unlike the previous work is done in this field, we compare the performance of machine learning algorithms on two types of Twitter posts i.e. context-specific and random tweets.

There are many different techniques and methods presented in the literature by many researchers for the sentimental analysis of Twitter posts. There have been massive amount of research conducted in the area of sentiment classification. There are many different techniques and methods presented in the literature by many researchers for the sentimental analysis of Twitter posts. In general, most of the research was done in classifying larger pieces of texts such as movie reviews or product reviews [4]. Twitter is slightly different from reviews as the maximum characters allowed per post is 140. Also, unlike reviews tweets are more casual in nature and are less thoughtfully composed. Twitter is an excellent medium for companies to collect feedback to study consumer behaviour. Previously researchers have built sentiment classifiers that can determine positive, negative and neutral sentiment for a sentence [5]. In a similar research,

Divya et al. used lexicons and Naive Bayes classifiers to implement mood classification in twitter data [6].

Barbosa et al. in their research designed a two-step automatic sentiment analysis method for the purpose of classification of tweets. For the purpose of the study, they adopted a noisy data set. They classified the tweets into two categories: - subjective and objective following that they further classified the subjective tweets into positive and negative [7].

Johan et al. in their research paper modelled public mood and emotion using sentiment analysis on Twitter. They used a psychometric aggregator to extract the six mood states from the Twitter data. Next, they computed a six-dimensional vector for each day in their timeline. They observed that events in the social, cultural and economic sphere could have an enormous impact on the public mood [8].

Mike Thelwall and Kevan Buckley in their research paper used two new approaches, lexicon extension and mood setting to improve the accuracy of their algorithm of topic-specific lexical Sentiment strength detection for the social web. It was Sentiment Mining in Twitter Using Machine Learning observed that both the methods were able to improve sentiment analysis performance on the corpora when the topic focus is tightest [9].

CHAPTER 3

SENTIMENT MINING

The opinions of others have a significant influence in our daily decision-making process. These decisions range from buying a product such as a smart phone to making investments to choosing a school and all decisions that affect various aspects of our daily life. Before the Internet, people would seek opinions on products and services from sources such as friends, relatives, or consumer reports.

However, in the Internet era, it is much easier to collect diverse opinions from different people around the world. People look to review sites (e.g., CNET, Epinions.com), e-commerce sites (e.g., Amazon, eBay), online opinion sites (e.g., TripAdvisor, Rotten Tomatoes, Yelp) and social media (e.g., Facebook, Twitter) to get feedback on how a particular product or service may be perceived in the market.

Similarly, organizations use surveys, opinion polls, and social media as a mechanism to obtain feedback on their products and services. Sentiment analysis or opinion mining is the computational study of opinions, sentiments, and emotions expressed in text. The use of sentiment analysis is becoming more widely leveraged because the information it yields can result in the monetization of products and services.

For instance, by obtaining consumer feedback on a marketing campaign, an organization can measure the campaign's success or learn how to adjust it for greater success. Product feedback is also helpful in building better products, which can have a direct impact on revenue, as well as comparing competitor offerings.

This project will describe the various, types of sentiment classification, explore how to convert unstructured text into structured opinions, and address the current challenges in the end. Unlike the any of previous work is done in this field, we compare the performance of machine learning algorithms on two types of Twitter posts i.e. context-specific and random tweets.

3.1 SENTIMENT CLASSIFICATION LEVELS

Sentiment analysis can occur at different levels: feature level document level or sentence level.

3.1.1 FEATURE LEVEL CLASSIFICATION

In this method, the aim is to recognise and extricate object features that have been mentioned on by the writer and conclude whether the opinion is negative, positive, or neutral. Feature synonyms are grouped together, and a feature-based report of multiple reviews is created.

3.1.2 DOCUMENT LEVEL CLASSIFICATION

In this method, sentiments are derived from the entire article, and a full opinion is classified based on the overall sentiments of the writer. The aim is to rank a review as neutral, positive or negative. For Instance: In the sentence I purchased an iPhone a few days ago. It is such a superior phone, though a little bulky. The touch screen is great. The call quality is clear too. I just adore it! Is the review classification negative or positive? Document level classification performs best when the document is written by a single person and communicates a sentiment/opinion on a single topic.

3.1.3 SENTENCE LEVEL CLASSIFICATION

This method normally comprises of two levels. In the first level classification of a sentence based on its subjectivity into one of the two classes: subjective and objective is done. and in the second level classification of individual sentences based on its sentiment into two categories: negative and positive is done.

A Subjective sentence represents personal beliefs, opinions, emotions, or feelings while an Objective sentence presents some actual knowledge. Subjective sentence identification can be accomplished by various techniques, for example, Naive Bayesian Classi-

fication. However, simply realising that sentences have a positive or negative opinion is not adequate. It is an intermediary action that helps in separating sentences without any views and also in determining if sentiments regarding entities and their aspects are negative or positive. A subjective sentence might contain many opinions, and subjective, and factual clauses. For Instance: In the sentence iPhone sales are doing great in this weak economy. sentiment classification at both the sentence and document levels is beneficial, although they do not discover what people like or dislike, nor do they help in identifying opinion targets.

3.2 NATURAL LANGUAGE PROCESSING NLP

NLP stands for Natural language processing, is the field of study that focuses on the communications between human language and computers.

NLP is a process for computers to analyse, understand, and derive meaning from human languages in an intelligent and valuable way. By using NLP, developers can organise and arrange knowledge to execute tasks such as automatic translation, summarization, relationship extraction, named entity recognition, sentiment analysis, topic segmentation, and speech recognition.

Besides common word processor operations that treat the text as a mere series of symbols, NLP also examines the hierarchical composition of language: many words make a phrase, many phrases make a sentence and, eventually, sentences communicates opinions. NLP systems analyse languages for their meaning and help in fulfilling roles such as converting speech to text, automatically translating between languages and correcting grammar.

NLP is applied to parse text, enabling machines to learn how humans speak. This interaction facilitates real-world applications like sentiment analysis, automatic text summarization, named entity recognition, topic extraction, part-of-speech tagging, stemming and relationship extraction. NLP is usually used for automated question answering, machine translation, and text mining.

NLP is characterised as a dilemma in computer sciences. Human language is seldom

precise. For understanding human languages it is vital to understand the concepts of a language and how different words are combined to form meaning. Although language is one of the simplest tasks for humans to master, the vagueness of language is what causes natural language processing a complex problem for computers to comprehend.

3.3 TEXT MINING

Text mining is an analytical field that acquires high-quality knowledge from text. It is extensively used in the industry when the data is unstructured. Text mining provides derived information in the form of numbers (indices), a summary of the text, categories or clusters.

Text Mining process is divided into various stages. The first step involves gathering unstructured data from websites, blogs, user comments, social media websites and emails. The next step is text parsing. This step requires extraction of words, stemming, parts of speech tagging, word filtering, synonyms, and tokenization. Following text parsing, text filtering is performed. It involves building stop word dictionary, removing stop words and removing irrelevant terms. The fourth step comprises of Building Document-Term Matrix (DTM) or Term-Frequency Document Matrix (TDM), calculating Singular Value Decomposition (SVD) and computing frequency term counts. Next, the text mining algorithms are applied to the data. Finally, Analysis, Insights and Recommendations are derived.

Text Mining is applied in various fields of sentimental analysis. It is used in analysing open-ended survey comments, customer insurance/warranty claims, feedback forms and sentiment of users against a particular campaign/reviews/product using social media data. It can also be used for automatic processing of emails, images and messages.

CHAPTER 4

SOFTWARE AND HARDWARE REQUIREMENTS

The output specifications detail is given at the astronomical point of the testing trip. The capability and execution distributed to programming as a serious facet of framework building are refined by fitting a complete information portrayal as a utilitarian portrayal of framework conduct, an indication of execution stipulations and description imperatives, appropriate approval criteria.

4.1 FUNCTIONAL REQUIREMENTS

It provides a definition as a Function of the System and its parts. A function is explained as a combination of various input, the working, and the output. The functional requirement may be calculated by technical specifications, data changes and processing and other special functionalities that define what a tool is expected to achieve. Behavioural requirement describes the cases where the tool uses the Functional Requirements are captured for use-cases.

4.2 NON-FUNCTIONAL REQUIREMENTS

It is a necessity that indicates criteria that can be utilised to judge the operation of a framework, as opposed to particular practices. Non-utilitarian prerequisites characterise how a framework should be. Non-useful prerequisites are frequently called characteristics of a framework.

4.3 INTEROPERABILITY

Interoperability is that the capacities to create frameworks and associations collaborate. Whereas the term was initially characterised for information innovation or frameworks building administrations to require into consideration information trade, much expansive definition considers social, political, and authoritative variables that impact framework to framework execution. Associate in Nursing trip of building national administrations for a shopper once the individual elements are indeed extraordinary and administered by the different association.

4.4 FLEXIBILITY

It is utilised as a trait of various sorts of frameworks. It additionally alludes to outline that can adjust when extreme changes happen. It has been characterised contrastingly in many fields of building design, science, financial matters, and so forth. Flexibility is the capacity of a framework to react to potential inner or outside changes influencing its qualities conveyance, in a convenient and practical way. Along these lines, adaptability for a building framework is the situation with which the framework can react to instability in a way to manage or increment its esteem conveyance.

4.5 SCALABILITY

It is the capability of a framework, system, or procedure to handle a developing live of labour during an adept manner or its ability to be distended to oblige that development. As an example, it will touch to the capability of a framework to make its combination yield below Associate in Nursing distended load once assets are enclosed. A much akin to significance is inferred once the word is used as a district of a money setting, wherever skillfulness of a company infers that the hidden set up of action offers the potential for financial development within the organisation. A framework whose execution enhances within the wake of as well as instrumentation, comparatively to the talents other is claimed to be a filmable framework.

4.6 HARDWARE REQUIREMENTS

Processor	:	Core 2 Duo and above
Hard Disk	:	500GB and above
Configuration	:	Optical Mouse with advanced capabilities
Keyboard	:	110 keys enhanced
RAM	:	4 GB or above

4.7 SOFTWARE REQUIREMENTS

Operating System	:	Linux
Softwares Required	:	R Studio, Python ide environment
Back End	:	Python

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The proposed system architecture for the mood classifier is shown in Figure 5.1. There are five main steps in the process, and they are as follows:

1. Creation of Dataset
2. Segregation of Dataset in two categories i.e. context-specific tweets and random tweets
3. Pre-Processing the data
4. Modelling
5. Evaluating the accuracy of each algorithm

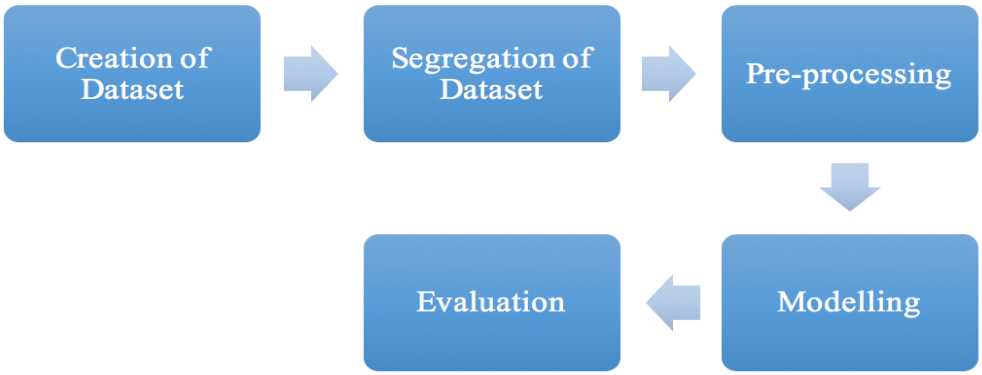

Figure 5.1: System Architecture

5.1.1 CREATION OF DATASET

For the purpose of this research, we create our own dataset. We download tweets on various random topics using the API provided by Twitter.

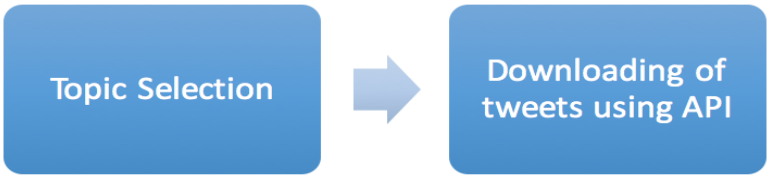

Figure 5.2: Creation of dataset

5.1.2 SEGREGATION OF DATASET

Next, we segregate the tweets into two categories for a better understanding of the importance of context in classification. We divide the tweets into context-specific tweets and random tweets. Context-specific tweets are the tweets relating to a particular topic. In this paper, we use the tweets on the topic of Mothers day. Random tweets are a collection of tweets ranging from a wide range of topics.

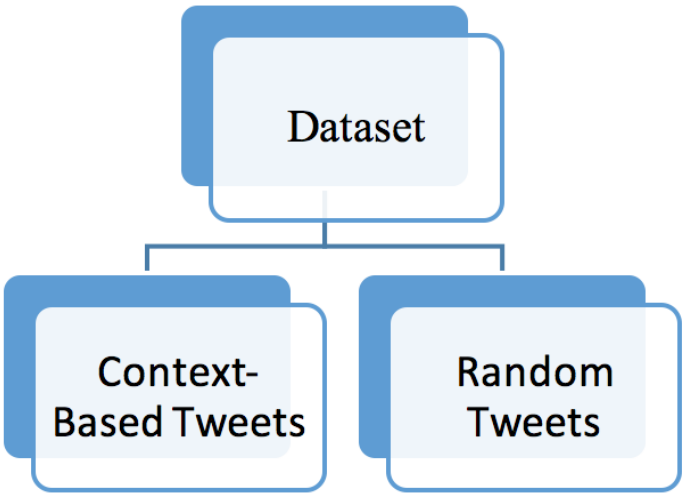

Figure 5.3: Segregation of tweets

5.1.3 PRE-PROCESSING

After we have segregated the data into different categories, we perform a pre-processing step. In this step we perform four functions that are as follows:

- Data Cleaning - Twitter data has many disturbances, and hence it required to be cleaned. The data is cleaned it by removing numbers and a few punctuations. Some of the punctuations were not removed as many people express their emo-

tions using punctuations. Twitter usernames and website data appearing in tweets were also removed.

- Data Splitting - After cleaning up the data, it is divided into train and test data. Train data is used for training a model, and test data are used for testing the accuracy of the model. The ratio of training to test data is 7:3.

- Feature Extraction - Features are the foundation to train a model. Here we used the bag of words technique and also extracted bigrams and trigrams from the data. Stopwords are also employed in features.

- Feature Selection - A Large number of features add noise to the model and give poor performance. To increase the speed and accuracy, we selected top 5000 features from the feature set. A statistical technique called chi-square is used to test for selecting best features.

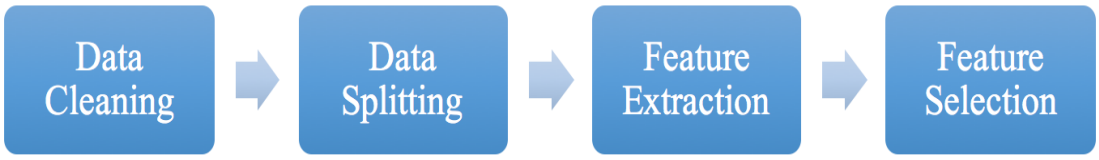

Figure 5.4: Steps involved in Pre-processing of the tweets

Twitter posts contain a lot of misspellings and slang words. Data cleaning is performed to get rid of such instances. In Data splitting the dataset is divided into training data and test data. Training data is used to train the model, and experimental data is used to check the performance of the model.

Dataset	Fun	Happy	Hate	Love	Sad
Training	655	1042	465	1346	1058
Test	281	447	199	577	454

Table 5.1: Statistics of the dataset used for Random tweets

Dataset	Happy	Love	Sad
Training	1556	953	978
Test	648	327	426

Table 5.2: Statistics of the dataset used for Context-specific tweets

Feature extraction involves extracting features from the entire dataset. First, we make a bag of words. Bag of words requires the counting of instances of each word in a particular sentence. Next, we use POS. POS is part of Speech, here we tag each sentence to its respective part of the expression. To increase the frequency, we also used bi-grams and tri-grams. Traditionally, researchers remove the stop words, punctuation symbols, and emoticons to reduce the size of data. We observed that removing them resulted in a decreased accuracy. Hence, in this paper, we did not remove stop words, punctuation symbols, and emoticons.

To save our data from running into a curse of dimensionality, we reduced the size of data by using a statistical technique called the Chi-square test.

5.1.4 MODELLING

After pre-processing the data, we applied different machine learning techniques on it. To understand the effect of different machine learning algorithms on accuracy, we followed two approaches. In the first method, features are trained using three different algorithms- KNN, SVM, NB. Here, in our research, the optimum value of the number of neighbours in the KNN algorithm is 20. In the second approach, we used LSA algorithm to study the importance of the context in a sentence. Following the use of LSA, classification is done using KNN, SVM and NB algorithms on the result.

CHAPTER 6

CLASSIFICATION TECHNIQUES

6.1 LATENT SEMANTIC ANALYSIS

Latent Semantic Analysis is also known as Latent Semantic Index (LSI). It is used extensively in text-based sentiment classification. Its purpose is to analyse text documents and understand the intrinsic meaning of those documents. The LSA algorithm follows a two-point approach to classify a document. The first step is called the bag of words approach. Here, the documents are represented as âĂIJbag of wordsâĂİ. A count matrix is prepared where the number of instances of each word is calculated. In this step, the sequence of words is not important. In the second phase, the algorithm looks for patterns of words that usually occur together in a particular kind of document. Next, we seek to identify the frequency of a particular word in each document following which we can assign weights to each word. The most popular weighting technique is called TFIDF. The equation as given by Britinger [10] is

$$TFIDFi, j = (Ni, j/N*, j) * log(D/Di) \tag{6.1}$$

In the above formula, TFIDF is the term frequency in the document, Ni,jÂădenotes the number of occasions the word i appears in the document j, N*,jÂădenotes the number of total words in the document j, D is the number of documents, and Di denotes the fraction of texts in which the word i appears.

6.2 NAIVE BAYES

Naive Bayes is a compilation of classification algorithms based on the Bayes theorem. The universal principle that each algorithm in the Naive Bayes family shares is that

every feature classified is independent of the value of any other characteristic. Naive Bayes is a simple algorithm that can sometimes outperform more sophisticated algorithms. The conditional probability for the Naive Bayes according to russels [11]is defined as:

$$P(Xjyj) =_m i = 1P(xijyj) \tag{6.2}$$

In this formula, is the feature vector defined as X=fx1 ,x2 ,....xmg and yj are the class label.

6.3 SUPPORT VECTOR MACHINE

Support Vector Machines aka SVM is a machine learning problem, widely used in classification problems. SVM's are based on the idea of finding the right hyperplane that can divide a dataset into two classes. SVM uses the discriminative function according to Peters [12] is defined as:

$$g(X) = wT_(X) + bb \tag{6.3}$$

In this formula, x is the feature vector, w is the weights vector and b is the bias vector. () is the non-linear mapping from input space to high dimensional feature space. w and b are learned automatically on the training set.

6.4 K-NEAREST NEIGHBOUR

K-Nearest Neighbours algorithm or the KNN algorithm is a non-parametric lazy learning algorithm. This algorithm stores all available cases and classifies new cases based on the similarity measure. It measures the distance between a query scenario and a set of other scenarios in the dataset. KNN algorithm is measured by a distance function. The equation according to Simmons [13] is defined as:

$$dist(x, y) = ni = 1(xi) - (yi) \tag{6.4}$$

In this formula, We can compute the distance between two scenarios using some distance function d(x,y), where x,y are scenes composed of N features.

CHAPTER 7

RESULTS

7.1 METRICS USED

7.1.1 ACCURACY

Accuracy is often the starting point for analyzing the quality of a predictive model, as well as an obvious criterion for prediction. Accuracy measures the ratio of correct predictions to the total number of cases evaluated. It may seem obvious that the ratio of correct predictions to cases should be a key metric. A predictive model may have high accuracy, but be useless.

7.1.2 PRECISION

Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called precision at n or Pn.

7.1.3 RECALL

Recall in information retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved. n binary classification, recall is called sensitivity. So it can be looked at as the probability that a relevant document is retrieved by the query.

7.2 ALGORITHMS

In this paper, we used KNN, LSA, SVM and NB algorithms to classify the two categories of tweets. For the first category, we sorted the data into five different moods - happy, sad, hate, love and fun. In the second group to understand the effect of context, we classified the tweets into three different moods such as happy, love, sad.

All these classifiers had almost similar accuracy for this feature vector. It was observed that all the classifiers performed better on the context based tweets.

7.2.1 LATENT SEMANTIC ANALYSIS

We used LSA to find the deeper meaning of the words used in tweets and use them in some context. But due to limited length of tweets LSA did not provide any better results as such. LSA can use aÂăterm-document matrixÂăwhich describes the occurrences of terms in documents; it is aÂăsparse matrixÂăwhose rows correspond toÂătermsÂăand whose columns correspond to documents. Here KNN and SVM are implemented over the results returned by LSA.

7.2.2 K-NEAREST NEIGHBOUR

KNN assumes that the data is in a feature space. More exactly, the data points are in a metric space. The data can be scalars or possibly even multidimensional vectors. Since the points are in feature space, they have a notion of distance âĂŞ This need not necessarily be Euclidean distance although it is the one commonly used. Since euclidean distance is calculated for finding the k nearest neighbors, tf idf vectors were created from features. We are also given a single number "k" . This number decides how many neighbors (where neighbors is defined based on the distance metric) influence the classification. If k=1 , then the algorithm is simply called the nearest neighbor algorithm.

Our optimal value for k is 8. After trying many values using grid search , for neighbhors 8 accuracy was the highest.

	Accuracy	Precision	Recall
KNN	.72	.68	.72
LSA + KNN	.69	.67	.69

Table 7.1: KNN Metric Calculation for Random Tweets.

	Accuracy	Precision	Recall
KNN	.93	.91	.93
LSA + KNN	.91	.90	.91

Table 7.2: KNN Metric Calculation for Context specific Tweets.

7.2.3 SUPPORT VECTOR MACHINE

In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. It selects the best hyper plane which has the maximum distance between hyperplane and data points. Since there are more than two classes, polynomial SVM is used which performed better than linear SVM.

	Accuracy	Precision	Recall
SVM	.71	.69	.71
LSA + SVM	.71	.69	.70

Table 7.3: SVM Metric Calculation for Random Tweets.

	Accuracy	Precision	Recall
SVM	.93	.92	.92
LSA + SVM	.90	.91	.90

Table 7.4: SVM Metric Calculation for Context-Specific Tweets.

7.2.4 NAIVE BAYES

NB is probably the most basic algorithm to train on text data. Most of the time it is very efficient, in our case also it shows very good results. It makes an assumption on the feature set that all the features are independent of each other. Multinomial naive bayes is used for text classification. Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. Whereas simple naive Bayes would model a document as the presence and absence of particular words, multinomial

naive bayes explicitly models the word counts and adjusts the underlying calculations to deal with in.

	Accuracy	Precision	Recall
NB	.70	.67	.71

Table 7.5: NB Metric Calculation on Random Tweets.

	Accuracy	Precision	Recall
NB	.93	.91	.93

Table 7.6: NB Metric Calculation on Context specific Tweets.

7.2.5 COMPARISON RESULTS

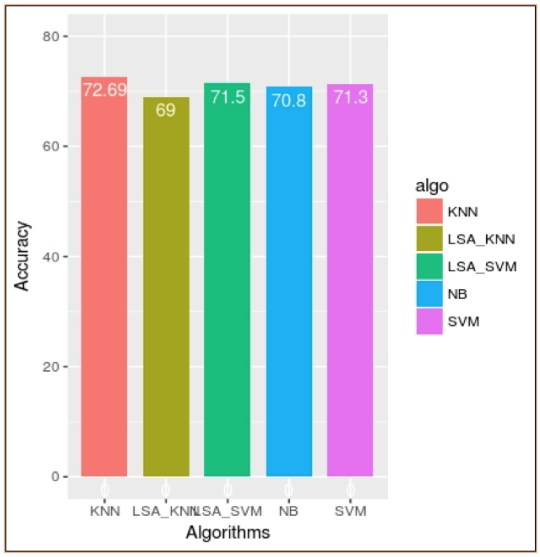

Figure 7.1: Performance on Random Tweets

	Accuracy	Precision	Recall
KNN	.72	.68	.72
NB	.70	.67	.71
SVM	.71	.69	.71
LSA + KNN	.69	.67	.69
LSA + SVM	.71	.69	.70

Table 7.7: The performance of Algorithms on Random Tweets.

In the case of random tweets, it was observed that KNN algorithm performed the best with an accuracy of 72.69 %.

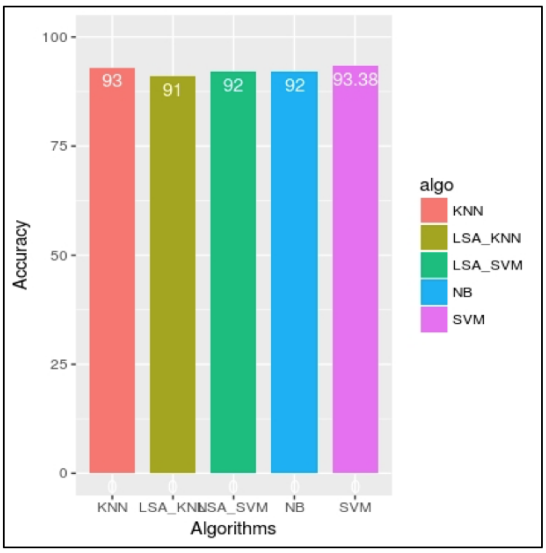

Figure 7.2: Performance on Context Specific Tweets

	Accuracy	Precision	Recall
KNN	.93	.91	.93
NB	.93	.91	.93
SVM	.93	.92	.92
LSA + KNN	.91	.90	.91
LSA + SVM	.90	.91	.90

Table 7.8: The performance of algorithms on Context specific Tweets.

Regarding context-specific tweets, it was found that the SVM algorithm performed the best with an accuracy of 93.38 % followed in closely by the KNN algorithm at 93 %.

CHAPTER 8

CONCLUSION

To better understand the effect of context on efficiency in the proposed system, we performed classification on context-specific tweets and random tweets. In both the approaches, we used KNN, SVM, NB for classifying. It was seen that context-specific tweets were more accurate than random tweets. In this paper, we applied LSA algorithm to make better use of context-based classification. But, due to the limited length of twitter posts, we observed that LSA algorithm did not perform as expected. Classification accuracy of the vector is tested using different classifiers like Naive Bayes, K-Nearest Neighbour.

APPENDIX A

SAMPLE CODE

A.1 SAMPLE CODE FOR TRAINING DATA

```
import time
import pandas as pd
import re
import nltk.classify.util
from sklearn import metrics
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.pipeline import make_pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import Normalizer
from sklearn.neighbors import KNeighborsClassifier
from nltk import pos_tag
from sklearn.svm import LinearSVC, SVC
from sklearn.feature_selection import SelectKBest, chi2

##########################################################

############################
#  Load the raw text dataset.
##########################################################

############################
```

```
print("Loading dataset...")

#raw_text_dataset = pd.read_csv("love_hate.csv")

#train_data = raw_text_dataset[0:1800]
#test_data = raw_text_dataset[1800:2586]

train_data = pd.read_csv("train_data.csv")
test_data = pd.read_csv("test_data.csv")

print(len(train_data))
print(len(test_data))

X_train_raw = train_data['content']
y_train_labels = train_data['sentiment']
X_test_raw = test_data['content']
y_test_labels = test_data['sentiment']



def review_to_words( review_text ):

#
# 2. Remove non-letters
letters_only = re.sub("[^a-zA-Z]", " ", review_text)
#
# 3. Convert to lower case, split into individual words
words = letters_only.lower().split()
#
# 4. In Python, searching a set is much faster than

searching
```

```
#    a list, so convert the stop words to a set
stops = nltk.corpus.stopwords.words('english')
stops += ['.', '://', 'http', 'com', '...']
# 5. Remove stop words
meaningful_words = [w for w in words if not w in stops]

# 6. Join the words back into one string separated

by space,
# and return the result.
return( " ".join( meaningful_words ))



print("Cleaning and parsing the training set

tweets...\n")
num_reviews = X_train_raw.size
clean_train_reviews = []
for i in range( 0, num_reviews ):
# If the index is evenly divisible by 1000, print a

message
if( (i+1)%1000 == 0 ):
print( "Review %d of %d\n" % ( i+1, num_reviews ))
clean_train_reviews.append( review_to_words(

X_train_raw[i] ))

postag_train_reviews = []
for d in clean_train_reviews:
data_tokens = nltk.wordpunct_tokenize(d)
temp = ["%s_%s" % (w,t) for w, t in pos_tag(data_tokens)]
```

```
postag_train_reviews.append(" ".join(temp))


print("Cleaning and parsing the testing set tweets...\n")
test_tweets = X_test_raw.size
clean_test_reviews = []
for i in range( 0 , test_tweets ):
# If the index is evenly divisible by 1000, print a

message
if( (i+1)%1000 == 0 ):
print( "Review %d of %d\n" % ( i+1, test_tweets ))
clean_test_reviews.append( review_to_words(

X_test_raw[i] ))


postag_test_reviews = []
for d in clean_test_reviews:
data_tokens = nltk.wordpunct_tokenize(d)
temp = ["%s_%s" % (w,t) for w, t in pos_tag(data_tokens)]
postag_test_reviews.append(" ".join(temp))




############################################################

############################
#  Use LSA to vectorize the articles.
############################################################
```

```
##########################

# Tfidf vectorizer:
#   – Strips out âĂIJstop wordsâĂİ
#   – Filters out terms that occur in more than half

of the docs (max_df=0.5)
#   – Filters out terms that occur in only one

document (min_df=2).
#   – Selects the 10,000 most frequently occuring

words in the corpus.
#   – Normalizes the vector (L2 norm of 1.0) to

normalize the effect of
#     document length on the tf−idf values.
vectorizer = TfidfVectorizer( min_df=2,

max_features=20000 ,stop_words=None, ngram_range=

(1,3),
use_idf=True)



# Build the tfidf vectorizer from the training data

("fit"), and apply it
# ("transform").

X_train_tfidf =
```

```
vectorizer.fit_transform(postag_train_reviews)

print("  Actual number of tfidf features: %d" %

X_train_tfidf.get_shape()[1])


print("\nPerforming dimensionality reduction using LSA")
t0 = time.time()

# Project the tfidf vectors onto the first 150

principal components.
# Though this is significantly fewer features than

the original tfidf vector,
# they are stronger features, and the accuracy is higher.
svd = TruncatedSVD(800)
lsa = make_pipeline(svd, Normalizer(copy=False))

# Run SVD on the training data, then project the

training data.
X_train_lsa = lsa.fit_transform(X_train_tfidf)


print("  done in %.3fsec" % (time.time() - t0))

explained_variance = svd.explained_variance_ratio_.sum()
print("  Explained variance of the SVD step:

{}%".format(int(explained_variance * 100)))
```

```
# Now apply the transformations to the test data as well.
X_test_tfidf = vectorizer.transform(postag_test_reviews)
X_test_lsa = lsa.transform(X_test_tfidf)

print(" train tfidf size",X_train_tfidf.size)
print("test tfidf size" ,X_test_tfidf.size)



############################################################

############################
#  Run classification of the test articles
############################################################

############################

print("\nUsing KNN on LSA vectors...")

# Time this step.
t0 = time.time()

# Build a k-NN classifier. Use k = 5 (majority

wins), the cosine distance,
# and brute-force calculation of distances.
knn_lsa = KNeighborsClassifier(n_neighbors=20,

algorithm='brute', metric='cosine')
knn_lsa.fit(X_train_lsa, y_train_labels)
```

```
# Classify the test vectors.
p = knn_lsa.predict(X_test_lsa)


# Measure accuracy

print(metrics.accuracy_score(y_test_labels, p))
print(metrics.classification_report(y_test_labels, p))
# Calculate the elapsed time (in seconds)
elapsed = (time.time() - t0)
print("    done in %.3fsec" % elapsed)



print("\nUsing LinearSVC on LSA features")
lsvc = LinearSVC()
lsvc.fit(X_train_lsa,  y_train_labels)
result = lsvc.predict(X_test_lsa)
print(metrics.accuracy_score(y_test_labels, result))
#print(metrics.confusion_matrix(y_test_labels,result))
print(metrics.classification_report(y_test_labels,

result))

print("\nUsing LIBSVC on LSA features")
libsvc = SVC(kernel='linear')
libsvc.fit(X_train_lsa,  y_train_labels)
result = libsvc.predict(X_test_lsa)
print(metrics.accuracy_score(y_test_labels, result))
print(metrics.classification_report(y_test_labels,

result))
```

```
print("\nUsing MNB on tfidf")
mnb = MultinomialNB()
mnb.fit(X_train_tfidf,  y_train_labels)
result = mnb.predict(X_test_tfidf)
print(metrics.accuracy_score(y_test_labels, result))
print(metrics.classification_report(y_test_labels,

result))
```

A.2 SAMPLE CODE FOR CLASSIFICATION OF TWEETS

```
import time
import pandas as pd
import re
import nltk.classify.util
from sklearn import metrics
from sklearn.feature_extraction.text import

TfidfVectorizer, CountVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.pipeline import make_pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import Normalizer
from sklearn.neighbors import KNeighborsClassifier
from nltk import pos_tag
from sklearn.svm import LinearSVC, SVC
from sklearn.feature_selection import SelectKBest, chi2

#####################################################
```

```
##########################
#  Load the raw text dataset.
#########################################################

##########################

print("Loading dataset...")

#raw_text_dataset = pd.read_csv("love_hate.csv")

#train_data = raw_text_dataset[0:1800]
#test_data = raw_text_dataset[1800:2586]

train_data = pd.read_csv("train_data.csv")
test_data = pd.read_csv("test_data.csv")

print(len(train_data))
print(len(test_data))

X_train_raw = train_data['content']
y_train_labels = train_data['sentiment']
X_test_raw = test_data['content']
y_test_labels = test_data['sentiment']


def review_to_words( review_text ):

#
# 2. Remove non-letters
letters_only = re.sub("[^a-zA-Z]", " ", review_text)
#
# 3. Convert to lower case, split into individual words
```

```
words = letters_only.lower().split()
#
# 4. In Python, searching a set is much faster than

searching
#   a list, so convert the stop words to a set
stops = nltk.corpus.stopwords.words('english')
stops += ['.', '://', 'http', 'com', '...']
# 5. Remove stop words
meaningful_words = [w for w in words if not w in stops]

# 6. Join the words back into one string separated

by space,
# and return the result.
return( " ".join( meaningful_words ))


print("Cleaning and parsing the training set

tweets...\n")
num_reviews = X_train_raw.size
clean_train_reviews = []
for i in range( 0, num_reviews ):
# If the index is evenly divisible by 1000, print a

message
if( (i+1)%1000 == 0 ):
print( "Review %d of %d\n" % ( i+1, num_reviews ))
clean_train_reviews.append( review_to_words(

X_train_raw[i] ))
```

```
postag_train_reviews = []
for d in clean_train_reviews:
data_tokens = nltk.wordpunct_tokenize(d)
temp = ["%s_%s" % (w,t) for w, t in pos_tag(data_tokens)]
postag_train_reviews.append(" ".join(temp))


print("Cleaning and parsing the testing set tweets...\n")
test_tweets = X_test_raw.size
clean_test_reviews = []
for i in range( 0 , test_tweets ):
# If the index is evenly divisible by 1000, print a

message
if( (i+1)%1000 == 0 ):
print( "Review %d of %d\n" % ( i+1, test_tweets ))
clean_test_reviews.append( review_to_words(

X_test_raw[i] ))


postag_test_reviews = []
for d in clean_test_reviews:
data_tokens = nltk.wordpunct_tokenize(d)
temp = ["%s_%s" % (w,t) for w, t in pos_tag(data_tokens)]
postag_test_reviews.append(" ".join(temp))




#########################################################
```

```
##########################
#  Use LSA to vectorize the articles.
##########################################################

##########################

# Tfidf vectorizer:
#   – Strips out âĂIJstop wordsâĂİ
#   – Filters out terms that occur in more than half

of the docs (max_df=0.5)
#   – Filters out terms that occur in only one

document (min_df=2).
#   – Selects the 10,000 most frequently occuring

words in the corpus.
#   – Normalizes the vector (L2 norm of 1.0) to

normalize the effect of
#     document length on the tf–idf values.
vectorizer = TfidfVectorizer( min_df=2,

max_features=20000 ,stop_words=None, ngram_range=
(1,3),
use_idf=True)

# Build the tfidf vectorizer from the training data

("fit"), and apply it
# ("transform").
```

```
X_train_tfidf =
vectorizer.fit_transform(postag_train_reviews)

print("  Actual number of tfidf features: %d" %
X_train_tfidf.get_shape()[1])

print("\nPerforming dimensionality reduction using LSA")
t0 = time.time()

# Project the tfidf vectors onto the first 150
principal components.
# Though this is significantly fewer features than
the original tfidf vector,
# they are stronger features, and the accuracy is higher.

svd = TruncatedSVD(800)
lsa = make_pipeline(svd, Normalizer(copy=False))

# Run SVD on the training data, then project the
training data.
X_train_lsa = lsa.fit_transform(X_train_tfidf)


print("  done in %.3fsec" % (time.time() - t0))

explained_variance = svd.explained_variance_ratio_.sum()
print("  Explained variance of the SVD step:
{}%".format(int(explained_variance * 100)))


# Now apply the transformations to the test data as well.
X_test_tfidf = vectorizer.transform(postag_test_reviews)
```

```
X_test_lsa = lsa.transform(X_test_tfidf)

print(" train tfidf size",X_train_tfidf.size)
print("test tfidf size" ,X_test_tfidf.size)


##########################################################
##########################
#  Run classification of the test articles
##########################################################
##########################

print("\nUsing KNN on LSA vectors...")

# Time this step.
t0 = time.time()

# Build a k-NN classifier. Use k = 5 (majority
wins), the cosine distance,
# and brute-force calculation of distances.
knn_lsa = KNeighborsClassifier(n_neighbors=20,
algorithm='brute', metric='cosine')
knn_lsa.fit(X_train_lsa, y_train_labels)

# Classify the test vectors.
p = knn_lsa.predict(X_test_lsa)


# Measure accuracy

print(metrics.accuracy_score(y_test_labels, p))
print(metrics.classification_report(y_test_labels, p))
```

```
# Calculate the elapsed time (in seconds)
elapsed = (time.time() - t0)
print("    done in %.3fsec" % elapsed)



print("\nUsing LinearSVC on LSA features")
lsvc = LinearSVC()
lsvc.fit(X_train_lsa,  y_train_labels)
result = lsvc.predict(X_test_lsa)
print(metrics.accuracy_score(y_test_labels, result))
#print(metrics.confusion_matrix(y_test_labels,result))
print(metrics.classification_report(y_test_labels,
result))

print("\nUsing LIBSVC on LSA features")
libsvc = SVC(kernel='linear')
libsvc.fit(X_train_lsa,  y_train_labels)
result = libsvc.predict(X_test_lsa)
print(metrics.accuracy_score(y_test_labels, result))
print(metrics.classification_report(y_test_labels,
result))

print("\nUsing MNB on tfidf")
mnb = MultinomialNB()
mnb.fit(X_train_tfidf,  y_train_labels)
result = mnb.predict(X_test_tfidf)
print(metrics.accuracy_score(y_test_labels, result))
print(metrics.classification_report(y_test_labels,
result))
```

APPENDIX B

SCREENSHOTS

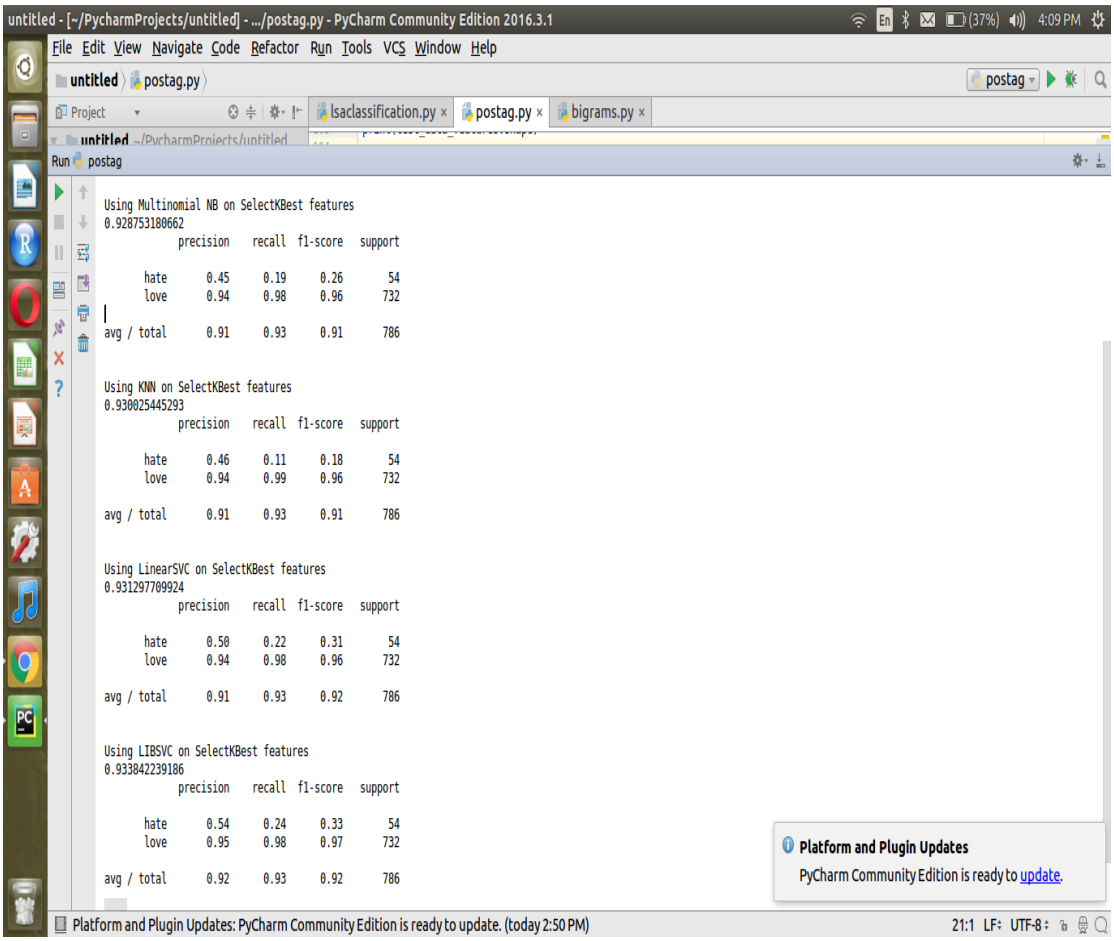

Figure B.1: Context specific tweets without LSA algorithm

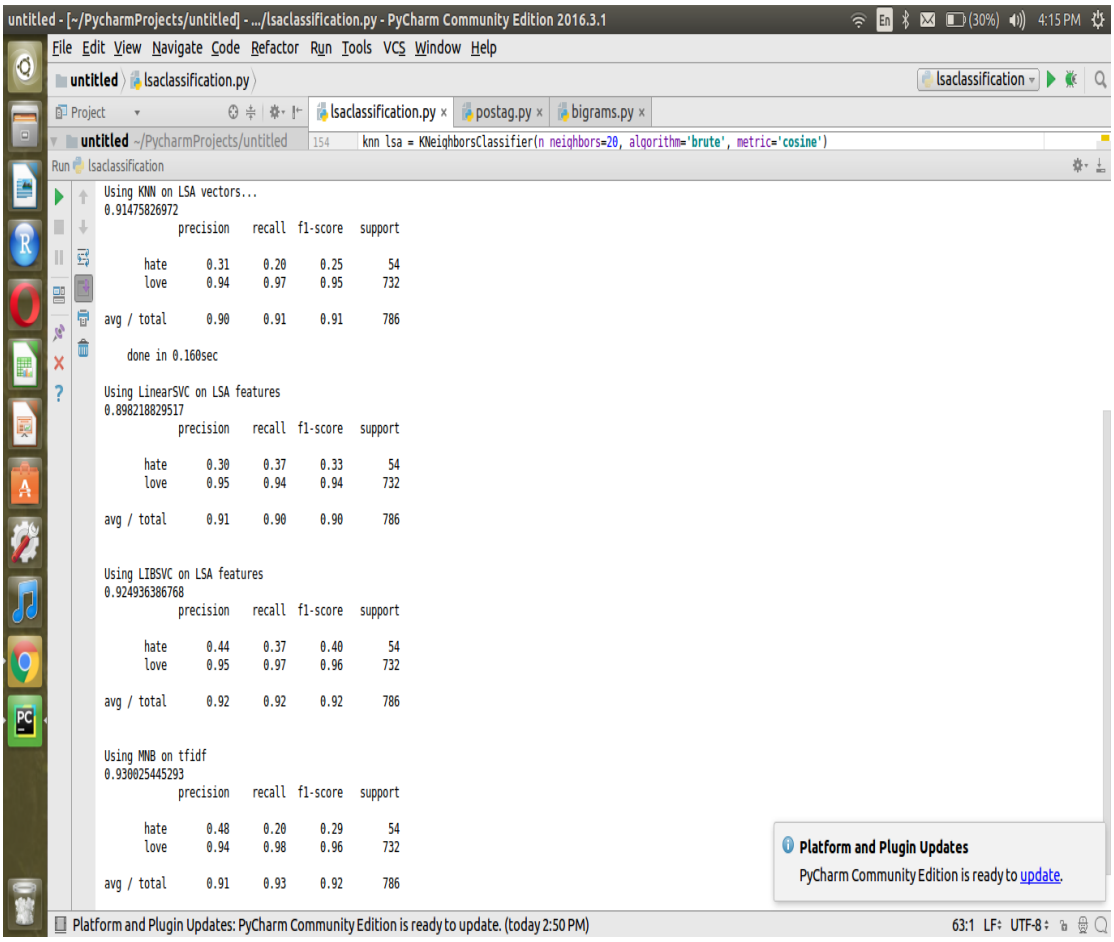

Figure B.2: Context specific tweets with LSA algorithm

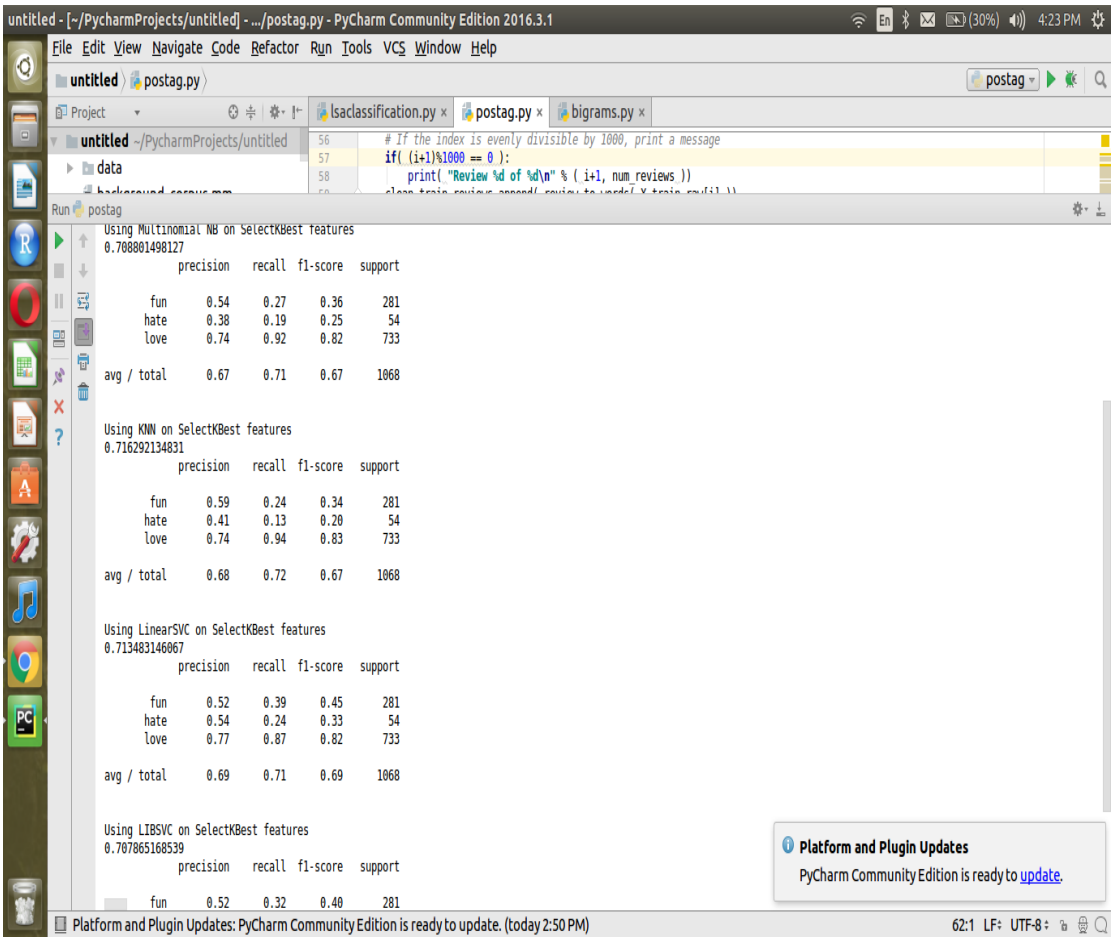

Figure B.3: Random tweets without LSA algorithm

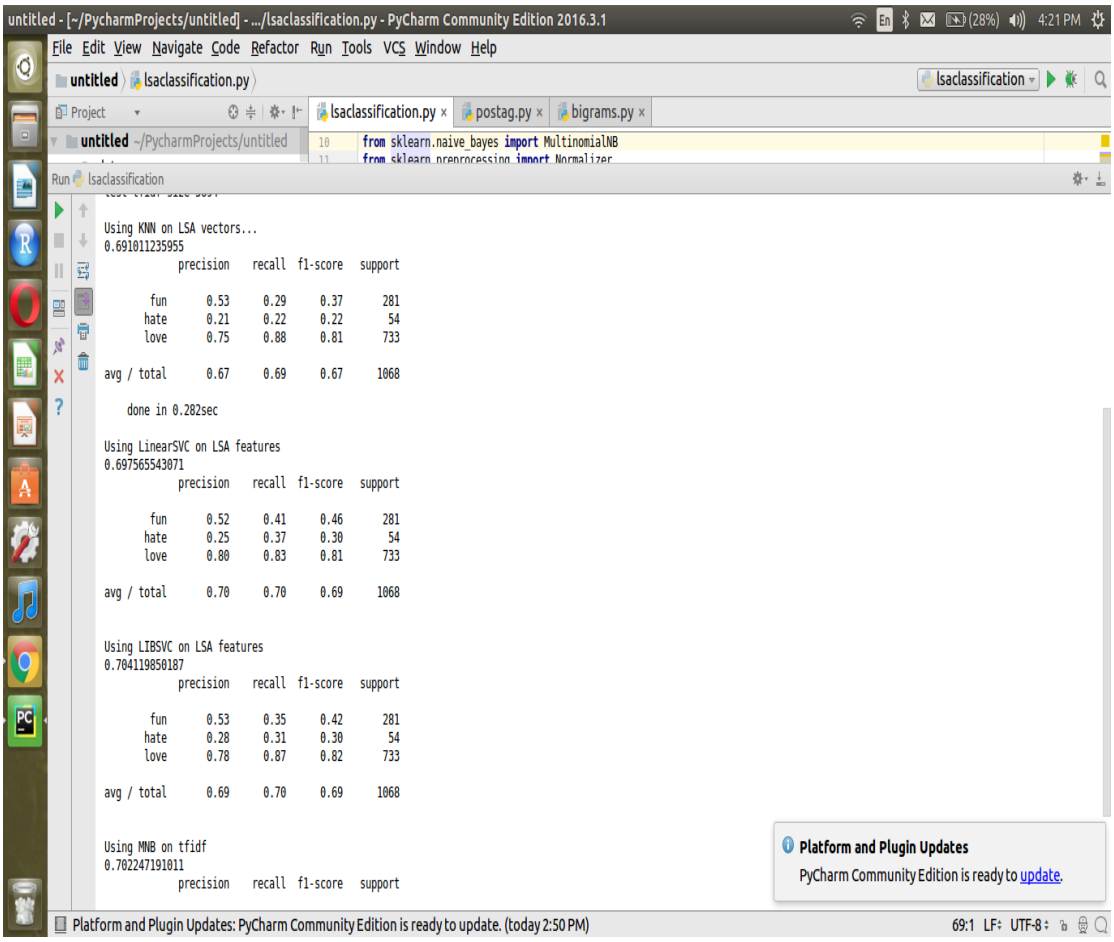

Figure B.4: Random tweets with LSA algorithm

REFERENCES

[1] J. Bollen, H. Mao, and X. J. Zeng, "Twitter mood predicts the stock market," in *Handbook of statistical analysis and datamining applications. Academic Press*, 2009.

[2] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpw, "Predicting elections with twitter: What 140 characters reveal about political sentiment," in *Association for Advancement of Artificial Intelligence*, vol. 48, pp. 146–170, 2013.

[3] Y. Mejova, "Sentiment analysis: An overview," in *International Journal of Engineering Research and Technology*, vol. 4, IJERT, 2013.

[4] B. Pang, L. Lee, and S. vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, vol. 4, pp. 169–174, IEEE, 2007.

[5] A. Pak and P. Paroubek, "Twitter as corpus for sentiment analysis and opinion minin," in *Journal of Computational Science*, vol. 4, pp. 236–270, 2008.

[6] L. D. Vani and D. Suneetha, "Mood classification of social media text," in *Fourth International conference on computing communications and Networking techonologies*, vol. 2, ICCCNT, 2013.

[7] L. Barbosa and J. feng, "Robust sentiment detection on twitter from biased and noisy data," in *Science in Information Technology (ICSITech), International Conference*, vol. 4, pp. 66–74, IEEE, 2015.

[8] J. Bollen, H. Mao, and A. Pepe, "Modeling public mood and emotion:twitter sentiment and socio-economic phenomena," in *Association for Advancement of Artificial Intelligence*, vol. 28, pp. 158–172, 2011.

[9] M. Thelwall and K. Buckley, “Topic-based sentiment analysis for the social web: The role of mood and issue-related words,” in *Journal of the American Society for Information Science and Technology*, vol. 8, pp. 146–170, 2010.

[10] S. Britinger, “Introduction to latent semantic analysis,” 2015.

[11] P. Russels, “Artificial intelligence: A modern approach,” 2008.

[12] A. Peters, “Support vector machines algorithm,” 2010.

[13] S. Simmons, “Nearest neighbor pattern classification,” 2013.