Bachelor Thesis Project

OPINION MINING AND SENTIMENT ANALYSIS

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT

FOR THE DEGREE OF B.E (COMPUTER ENGINEERING)

SUBMITTED BY:

Sagar Negi (346/CO/12)

Shrey Gupta (363/CO/12)

Vipul Khullar (393/CO/12)

GUIDED BY:

Dr. M.P.S. Bhatia

(Professor, Division of Computer Engineering)

DIVISION OF COMPUTER ENGINEERING

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF DELHI 2015-2016

Acknowledgement

We would like to express our deepest gratitude to our adviser Prof. MPS Bhatia for giving us the opportunity to work under his able supervision. He has been a pillar of strength throughout this research, showing us the way from the very beginning. The frequent brainstorming sessions have helped us bring out the best and have made us always feel excited about the work we are doing. His encouragement in times of distress and anxiety; patience in hearing our problems; rigorous reviewing to make sure that everything falls in place; and the immense knowledge that he possesses have helped us carry this project forward in the right spirits. Without his invaluable and continuous contributions, this project would not have been completed.

Netaji Subhas Institute of Technology
(Formerly, Delhi Institute of Technology)
Azad Hind Fauj Marg, Sector - 3, Dwarka, New Delhi – 110078 Telephone: 25099043, 25099020
Website: http://www.nsit.ac.in

CERTIFICATE

The project titled **Opinion Mining and Sentiment Analysis** by **Sagar Negi (346/CO/12), Shrey Gupta (363/CO/12) and Vipul Khullar (395/CO/12)** is a record of bona fide work carried out by us, in the Division of Computer Engineering, Netaji Subhas Institute of Technology, New Delhi, under the supervision and guidance of **Dr. M.P.S. Bhatia** in partial fulfilment of requirement for the award of the degree of Bachelor of Engineering in Computer Engineering, University of Delhi in the academic year 2015 - 2016.

Dr. M.P.S. Bhatia

Division of Computer Engineering

Netaji Subhas Institute of Technology

New Delhi

Date: _____

Candidates' Declaration

This is to certify that the work which is being hereby presented by us in this project titled **Opinion Mining and Sentiment Analysis** in partial fulfilment for the award of the Bachelor of Engineering submitted at the Department of Computer Engineering , Netaji Subhas Institute of Technology Delhi, is a genuine account of our work carried out during the period from January 2016 to May 2016 under the guidance of Dr. M.P.S. Bhatia, Department of Computer Engineering, Netaji Subhas Institute of Technology Delhi. The matter embodied in the project report to the best of our knowledge has not been submitted for the award of any other degree elsewhere.

Sagar Negi Shrey Gupta Vipul Khullar

Date: _____

This is to certify that the above declaration by the students is true to the best of my knowledge.

Dr. M.P.S. Bhatia

Date: _____

Contents

LIST OF FIGURES

Abstract

Opinion mining or Twitter sentiment analysis, the process of automatically extracting sentiment conveyed by Twitter data, is a field that has seen a dramatic increase in research in recent times. This **Bachelor'**s Thesis presents a study of the effects of linguistic negation on Twitter sentiment analysis.

Current state-of-the-art solutions in Twitter sentiment analysis and negation scope detection have been explored. Furthermore, a corpus of English Twitter data (*tweets*) annotated for linguistic negation has been created, and a system for negation scope detection in Twitter sentiment analysis has been developed and evaluated.

Our research represents the first work that explores sophisticated negation scope detection methods on tweets. The system produces better results than what has been reported in other domains. It has been incorporated into a state-of-the-art Twitter sentiment analysis classifier and the effects have been compared to other solutions commonly used within this field.

The study shows that the inclusion of the developed negation scope detection method in a Twitter sentiment analysis system improves the performance on tweets containing negation. However, the sparse distribution of linguistic negation in tweets results in a marginal performance gain on general data.

1. Introduction

With the recent growth of mobile information systems and the increased availability of smart phones, social media has become a large part of daily life in most societies. This development has entailed the creation of massive amounts of data: data which when analysed can be used to extract valuable information about a variety of subjects.

Sentiment analysis (SA), also known as *opinion mining* is the process of classifying the emotion conveyed by a text, for example as negative, pos-itive or neutral. The data made available by social media has contributed to a burst of research activity within SA in recent times and a shift in the focus of the field towards this type of data. Information gained from applying SA to social media data has many potential usages, for instance, to help marketers evaluate the success of an ad campaign, to identify how di erent demographics have received a product release, to predict user behaviour, or to forecast election results.

Sentiment Analysis refers to the use of text analysis and natural language processing to identify and extract subjective information in textual contents. There are two type of user-generated content available on the web – facts and opinions. Facts are statements about topics and in the current scenario, easily collectible from the Internet using search engines that index documents based on topic keywords. Opinions are user specific statement exhibiting positive or negative sentiments about a certain topic. Generally opinions are hard to categorize using keywords. Various text analysis and machine learning techniques are used to mine opinions from a document [1]. Sentiment Analysis finds its application in a variety of domains.

1.1. Twitter Sentiment Analysis

A popular social medium is Twitter,[1] a micro-blogging site that allows users to write textual entries of up to 140 characters, commonly referred to as *tweets*. As of June 2015, Twitter has over 302 million monthly active users according to their homepage, whereof approximately 88 % have their tweets freely readable. Additionally, over 84 % of the users also have their location specified in their profiles , enabling the possibility of performing drill-down on geographic locations. Data created by Twitter is made available through Twitter's API, and represents a real-time information stream

of opinionated data. Tweets can be filtered both by location and the time they were published. This has paved the way for a new sub-field of SA: Twitter sentiment analysis (TSA).

Length of a Tweet The maximum length of a Twitter message is 140 characters. This means that we can practically consider a tweet to be a single sentence, void of complex grammatical constructs. This is a vast difference from traditional subjects of Sentiment Analysis, such as movie reviews. **Language used** Twitter is used via a variety of media including SMS and mobile phone apps. Because of this and the 140-character limit, language used in Tweets tend be more colloquial, and filled with slang and misspellings. Use of hashtags also gained popularity on Twitter and is a primary feature in any given tweet. Our analysis shows that there are approximately 1-2 hashtags per tweet, as shown. **Data availability** another difference is the magnitude of data available. With the Twitter API, it is easy to collect millions of tweets for training. There also exist a few datasets that have automatically and manually labelled the tweets. **Domain of topics** People often post about their likes and dislikes on social media. These are not all concentrated around one topic. This makes twitter a unique place to model a generic classifier as opposed to domain specific classifiers that could be build datasets such as movie reviews.

Performing natural language processing on textual data from Twitter presents new challenges because of the informal nature of this data. Tweets often contain misspellings, and the constrictive limit of 140 characters en-courages slang and abbreviations. Unconventional linguistic means are also used, such as capitalization or elongation of words to show emphasis. Additionally, tweets contain special features like *emoticons* and *hashtags* that may have an analytical value. Hashtags are labels used for search and categorisation, and are included in the text prepended by a **"#"**. Emoticons are expressions of emotion, and can either be written as a string of char-acters e.g., **":-)"**, or as a unicode symbol. Finally, if a tweet is a reply or is directed to another Twitter user, *mentions* can be used by prepending a username with **"@"**.

1.2. Motivation and Research Focus Area

Business Businesses may use sentiment analysis on blogs, review websites etc. to judge the market response of a product. This information may also be used for intelligent placement of advertisements. For example, if product "A" and "B" are competitors and an online merchant business "M" sells both, then "M" may advertise for "A" if the user displays positive sentiments towards "A", its brand or related products, or "B" if they display negative sentiments towards "A". **Government** Governments and politicians can actively monitor public sentiments as a response to their current policies, speeches made during campaigns etc. This will help them make create better public awareness regarding policies and even drive campaigns intelligently. **Financial Markets** Public opinion regarding companies can be used to predict performance of their stocks in the financial markets. If people have a positive opinion about a product that a company A has launched, then the share prices of A are likely to go higher and vice versa. Public opinion can be used as an additional feature in existing models that try to predict market performances based on historical data.

In this project we explore the e ect of applying sophisticated *negation scope detection and use of bigrams and trigrams* to Twitter sentiment analysis or opinion mining through twitter.

The linguistic phenomenon of negation bigrams and trigrams are , described, have been shown to play a significant role in opinion mining . Councill et al. tested a sentiment classifier and found that including their negation classifier provided a 29.5 % improvement in F_1 score when classifying positive sentiment, and an 11.4 % improvement when classifying negative sentiment.

1.3. Project Goals

The goals for this project were five-fold, as follows:

G1: Research the State-of-the-Art in Twitter Sentiment Analysis

TSA has been the topic of several shared tasks hosted by the Association for Computational Linguistics.2 We will base our state-of-the-art research mainly on the results from these tasks, and naturally focus more on submissions that achieved the best results. The purpose of this goal is to form the knowledge basis required for developing a state-of-the-art TSA system.

G2: Create a Twitter Corpus Annotated for Opinion mining

To allow for the use of supervised machine learning techniques when developing a classification system, a corpus annotated for the presence of sentiment is required. Several sentiment -annotated corpora have been created and are freely available. As tweets contain unique characteristics, a classifier trained on out-of-domain data may struggle to classify Twitter data. Therefore, we aim to create a Twitter corpus.

G3: Develop a gettweets script using twitter API

We aim to create a script of collecting tweets real-time using the Twitter API. The API has a rate limit of 100 tweets per user window of 15 minutes. Tweets collected are then fed into the classifier for their sentiment analysis.

G4: Develop a Twitter Sentiment Classifier

The final goal is to create a state-of-the-art TSA system. This classifier will then be implemented in the pipeline of a TSA system. The performance of the classifier will be tested against some popular models available.

G5: Displaying the analytics using a pie chart

Main focus is to use any analytical api to display the results of the sentiment analysis of tweets and thus giving the user a pictorial and analytical view of the situation.

1.4. Thesis Structure

Relevant background theory for the project in addition to the tools and external data sets used are described in Chapter 2. Chapter 3 presents the current state-of-the-art in TSA. Additionally, it contains a description of recent developments within the field of identifying linguistic negation, and a detailed look at some solutions, as well as an evaluation of their suitability for application in TSA. The resulting corpus and its characteristics are also presented. Chapter 5 contains the implementation details of two classification systems that have been created: TSA system. Experiments conducted on these systems and their results are presented in Chapter 6. Finally, Chapter 7 contains an evaluation of the degree to which the project goals have been achieved, as well as the conclusions drawn and suggested future work.

2. Tools and Methods

This chapter contains a presentation of the background theory and technological tools relevant to this project, as well as the external data used in the conducted experiments.

2.1. Background Theory

Sentiment analysis (SA) comprises many concepts common to the whole field of natural language processing in addition to many concepts from machine learning: the most relevant of these are described in this section.

2.1.1. Machine Learning

Machine learning has become a cornerstone in the field of SA. Most well performing systems incorporate some form of supervised machine learning; this is discussed further in Section 3.4. Here, we give a description of several machine learning algorithms relevant to the current state-of-the art in sentiment analysis.

Support Vector Machines

The Support Vector Machine (SVM) classification algorithm was formally described by Cortes and Vapnik [1995]. The algorithm considers data points based on their spatial location, and attempts to split the feature space into optimal class segments. This division of the feature space is referred to as training the machine. A trained SVM can then be used for classification of new examples by assigning them a class based on which segment of the feature space they are located in. In its basic form, it is an algorithm for linear classification of binary problems.

When dealing with two linearly separable classes, the feature space is divided into class segments by creating a hyperplane with the largest possible margin between the two classes. This margin maximization is the essential concept of SVMs. The closest data points of both classes, parallel to the vector defining the hyperplane, constitute the support vectors. These give the algorithm its name. Figure 2.1 shows the hyperplane and support vectors in a two dimensional linear classification problem.

The algorithm can also be applied to problems where the examples are not linearly separable by allowing misclassified points. If no hyperplane exists to split all examples, the soft margin method introduces a slack variable that gives a penalty for each misclassified

example [Cortes and Vapnik, 1995]. This penalty increases with the **distance from the example's support vector. The slack variable** governs the trade-off between classification errors and margin size. To allow for non-linear classification, one can map the data into a higher dimensionality space by using the kernel trick. This is done by applying a kernel function to the data. The process is well explained in Fletcher [2009]. Popular kernel functions include radial basis function (RBF), Sigmoidal Kernel, and Polynomial Kernel. Figure 2.2 shows a data remapping done with an RBF kernel. Using the RBF kernel function, there are two parameters that require adjustment for good performance: the regularization parameter C and the influence **radius parameter γ.**

C is a penalty for misclassification, and controls the complexity of the decision surface. A low C makes a complex hyperplane that attempts to correctly classify most of the training samples (increasing the chance of overfitting), while increasing the C regularizes the hyperplane, resulting in a more generalized classifier. Th**e γ controls** the radius of influence for each training example, in an inverse **manner. This means that a low γ makes each training example have an influence over a larger area, and a higher γ will result in the training** examples only influencing themselves. These two parameters are highly dependent on each other.

Additionally, the SVM algorithm can be applied to classification problems featuring more than two classes. Commonly used methods include one-against-one and one-against-the-rest. Hsu and Lin [2002] provide a thorough comparison of these methods

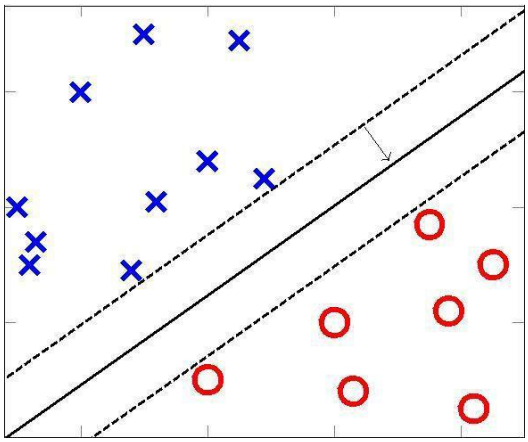


Figure .1.: An SVM linearly separating two classes

Naïve Bayes Classifier

Naïve Bayes (NB) classifiers, also known as Naïve Bayes Learners, are a relatively simple group of probabilistic classifiers based on **Bayes' Theorem. For classification in certain domains, their** performance has been shown to be comparable to much more complex machine learning algorithms, like neural networks or decision-tree learners [Mitchell, 1997].

Given a document d, the task of assigning a class c to the document is done by looking at the probabilities of each class given the document, and finding the

maximum a posteriori (MAP) probability∈ estimate: **cMAP = argmax c C**
P(c|d)

The formula for each class can be rewrittene∈ using Bayes' Theorem:

$$\mathbf{C(MAP) = argmax\ c\ C\ P(d|c) \times P(c)\ P(d)}$$

Because the probability of the document is the same across all classes, the class that maximises the numerator is the same as the class that maximises the whole expression.

$$\mathbf{C(MAP) = argmax\ c \in C\ P(d|c) \times P(c)}$$

The denominator can then be dropped:

The document is represented as a vector of feature attributes: d = a1, a2, a3, ..., an

This gives the approach used by the NB classifier:∈

$$\mathbf{C(NB) = argmax\ c\ C\ P(c)\ Y\ a\ P(a|c)}$$

The NB classifier assumes that all attribute values are conditionally independent. Despite this assumption, the classifier performs well on text classification tasks in practice.

2.1.2. Natural Language Processing

Natural language processing (NLP) is a field in the Human-Machine Interaction area concerned with the use of human natural languages for communication with computers. Among the many topics of NLP, the following are particularly relevant in this project.

Bag-of-Words Model

A common way to represent text documents in a simplified manner is by using a bag-of-words model. The technique lists term occurrence and optionally the frequency of term occurrence, disregarding grammar and term order. Machine learning classifiers can use the resulting model directly as feature vectors.

Linguistic Negation

Linguistic negation is a grammatical concept that encompasses devices used to reverse the truth value of propositions in language. Givón [1993] defines two forms of grammatical negation: morphological negation, where individual words are negated with an affix, and syntactic negation, where a set of words is negated by a word or phrase. Negators in syntactical negation, known as negation cues or negation signals, function as operators, with an associated affected scope of words. Syntactic negation is what is most relevant within NLP, as well as textual data mining in general, and is what we mean throughout this report when referring to negation.

Part-of-Speech Tagging

Part-of-speech (POS) tagging is the process of categorizing the tokens of a sentence into the different parts of speech (such as nouns, verbs, adjectives and adverbs) based on their definitions as well as the

contexts. This way, POS tagging attempts to solve the problem of word ambiguity. There are many POS taggers for regular languages trained on treebanks — particularly for the newswire domain — such as the Penn Treebank .However, the conversational language of Twitter causes an out-of-domain problem for these traditional POS taggers, degrading their performance.

n-grams(bigrams and trigrams)

In the fields of computational linguistics and probability, an ***N*-gram** is a contiguous sequence of *n* items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The *n*-grams typically are collected from a text or speech corpus. When the items are words, *n*-grams may also be called **shingles**. *N*-gram models are widely used in statistical natural language processing.

Percentage of Correctly Classified Scopes

For several classification tasks where the output is a sequence, classification metrics that only consider individual units regardless of their order are insufficient. Percentage of correctly classified scopes (PCS) is a metric for measuring the performance of a scope classifier,. A scope is considered correctly classified if, for a given tweet cue, every token in its associated scope has been correctly marked as in a negation scope. The evaluation metric accuracy is defined as the number of correctly classified samples divided by the total number of samples. The PCS metric can thus be considered an accuracy measure.

2.2. Tools

Many external tools and libraries were necessary for the realization of this project. The following lists the most important of these.

Natural Language Toolkit

The **Natural Language Toolkit**, or more commonly **NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for the Python programming language. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems.

Flask

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. Flask is called a micro framework because it does not presume or force a developer to use a particular tool or library. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

oAuth

OAuth is an open standard for authorization, commonly used as a way for Internet users to log in to third party websites using their Microsoft, Google, Facebook, Twitter, One Network etc. accounts without exposing their password.

Re(library)

The module **re** provides full support for Perl-like regular expressions in Python. The re module raises the exception re.error if an error occurs while compiling or using a regular expression. Here the use of this library occurs in preprocessing of tweets.

High Charts

is a charting library written in pure javascript, offering an easy way of adding interactive charts to your web site or web application. Highcharts currently supports many chart types, including line, spline, area, areaspline, column, bar, pie, scatter, bubble, gauge and polar chart types.

Twitter API

The REST APIs provides programmatic access to read and write Twitter data. Author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

CSV

In computing, a **comma-separated values** (**CSV**) file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

ArgPrse

the recommended command-line parsing module in the Python standard library. This was written for argparse in Python 3. A few details are different in 2.x, especially some exception messages, which were improved in 3.x.

2.3. External Data Sets

One of the major challenges in Sentiment Analysis of Twitter is to collect a labelled dataset. Researchers have made public the following datasets for training and testing classifiers.

Twitter Sentiment Corpus

This is a collection of 5513 tweets collected for four different topics, namely, Apple, Google, Microsoft, Twitter It is collected and hand-classified by Sanders Analytics LLC [3]. Each entry in the corpus contains, Tweet id, Topic and a Sentiment label. We use Twitter-Python library to enrich this data by downloading data like Tweet text, Creation Date, Creator etc. for every Tweet id. Each Tweet is hand classified by an American male into the following four categories. For the purpose of our experiments, we consider Irrelevant and Neutral to be the same class.

Positive For showing positive sentiment towards the topic **Positive** For showing no or mixed or weak sentiments towards the topic **Negative** For showing negative sentiment towards the topic **Irrelevant** For non-English text or off-topic comments

Class	Count	Example
neg	529	#Skype often crashing: #microsoft, what are you doing?
neu	3770	How #Google Ventures Chooses Which Startups Get Its $200 Million http://t.co/FCWXoUd8 via @mashbusiness @mashable
pos	483	Now all @Apple has to do is get swype on the iphone and it will be crack. Iphone that is

Stanford Twitter

This corpus of tweets, developed by Sanford's Natural Language processing research group, is publically available . The training set is collected by querying Twitter API for happy emoticons like ":)" and sad emoticons like ":(" and labelling them positive or negative. The emoticons were then stripped and Re-Tweets and duplicates removed.

It also contains around 500 tweets manually collected and labelled for testing purposes. We randomly sample and use 5000 tweets from this dataset. An example of Tweets in this corpus are shown in Table 2

Class	Count	Example
neg	2501	Playing after the others thanks to TV scheduling may well allow us to know what's go on, but it makes things look bad on Saturday nights
pos	2499	@francescazurlo HAHA!!! how long have you been singing that song now? It has to be at least a day. i think you're wildly entertaining!

3. Related Work

This chapter first looks at the recent development within the field of sentiment analysis (SA) and discusses a shift of the field towards Twitter data. The state-of-the-art in Twitter sentiment analysis (TSA).

3.1. Recent Developments within Sentiment Analysis

Exploring popular opinion on various subjects has always been an **important part of humans' information gathering behaviour. Where** one in the past needed to conduct surveys to learn about opinion trends, for instance to conduct political polls, the availability of online data expressing sentiment has allowed for non-intrusive data mining to extract this information.

Over the last decade, there has been a substantial increase in the amount of work done in the field of SA. Surveys conducted by Pang and Lee [2008] and Liu and L. Zhang [2012] give a good overview of the state-of-the-art at the respective points in time. The work in the field of SA has largely followed the available data, both in terms of the amount of work done and the focus area. Figure 3.1 shows the amount of hits for queries (3.1) and (3.2) on Google Scholar,1 displaying a shift of the field towards Twitter data in recent years.

"opinion mining" OR "sentiment analysis" OR "opinion classification"

"opinion mining" OR "sentiment analysis" OR "opinion classification"

AND "micro-blogging" OR "microblogging" OR "twitter"

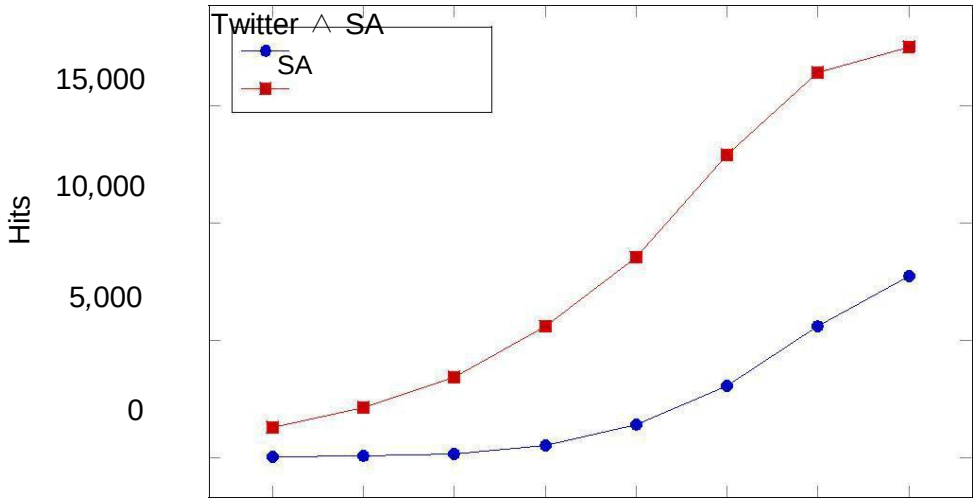


Figure 2.: Google Scholar hits for queries SA: (3.1) and Twitter ∧ SA: (3.2), for articles published from 2007 up to and including 2014

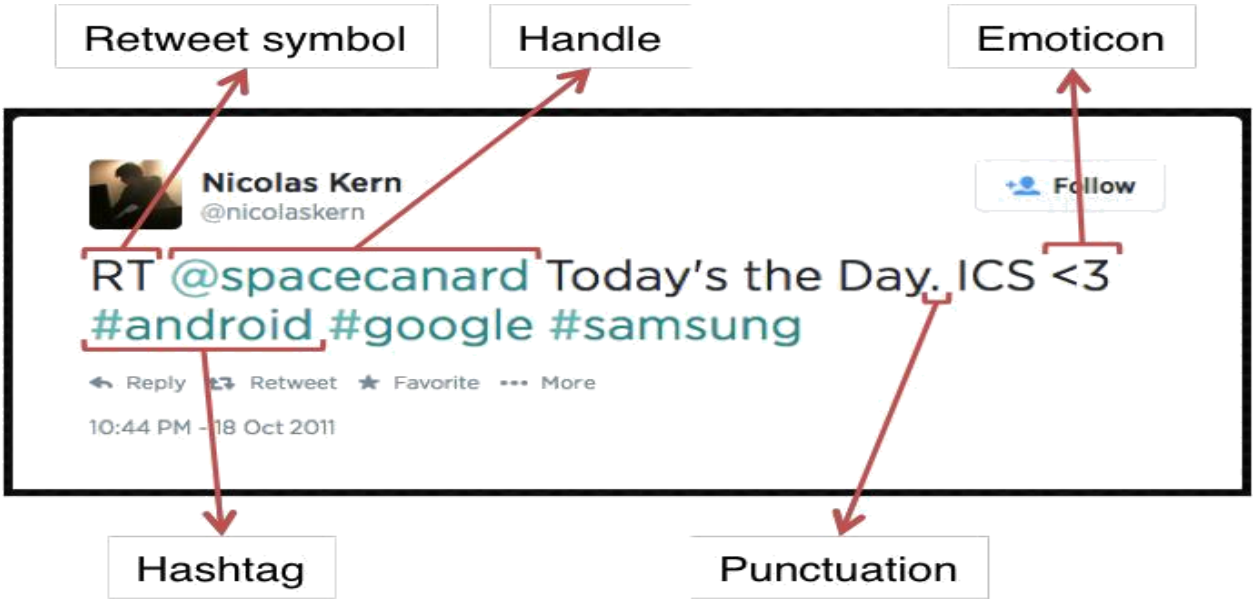
3.2. State-of-the-Art in Twitter Sentiment Analysis

In this section we present the current state-of-the-art in TSA by breaking the field down into several areas. The typical approach to TSA uses a supervised machine learning system including three main steps: pre-processing, feature extraction, and training the classifier. To reduce noise and remove unnecessary information, the pre-processing step consists of a variety of filters for, e.g., normalizing URLs and elongated words. Features for the classifier are extracted using sentiment scores from polarity lexica, statistics from metacommunicative expressions specific to conversational language such as emoticons and hashtags, as well as natural language processing information including bag-of-words, part-of-speech tags and word clusters. Finally, training the classifier is usually a matter of performing a grid search over the parameter space for selecting the most suitable parameters for a supervised machine learning model.

3.2.1. Tweet Pre-processing

Common preprocessing tasks in TSA include filtering out or normalizing URLs and user mentions, because these items have minimal information value in the context of sentiment classification. Agarwal et al. [2011] perform this normalization by substituting user mentions with the tag ||T|| and URLs with the tag ||U||. Another Twitter-specific syntactic feature is prefixing tweets with **"**RT**"** to indicate that the following part of the tweet is a retweet — a repost of previous content. A simple way of handling this is to remove the **"**RT**"** string from the tweet. It is also common to normalize elongated words, e.g., cooooooll, soooooooo, or happyyyyyy by substituting letters that occur many times sequentially with one or two occurrences of the letter. It was previously quite common to filter out hashtags [Selmer & Brevik 2013]. The assumption behind this is that hashtags when used as intended — i.e., to categorize posts by topic — offer little information of value. Mohammad [2012] show through experiments that hashtags add sentiment-semantic information to tweets by indicating the tone of the **message or the writer's emotions.** The following tweet in relation to the Occupy Wall Street (OWS) movement is an example of this:

> **"***We are fighting for the 99 % that have been left behind.*
> *#OWS #anger***"**



Hashtags

A hashtag is a word or an un-spaced phrase prefixed with the hash symbol (#). These are used to both naming subjects and phrases that are currently in trending topics. For example, #iPad, #news

Regular Expression: #(\w+)
Replace Expression: HASH_\1

Handles

Every Twitter user has a unique username. Any thing directed towards that user can be indicated be writing their username **pre eded y @ . Thus, these are** like proper nouns. For example, @Apple

Regular Expression: @(\w+)
Replace Expression: HNDL_\1

URLs

Users often share hyperlinks in their tweets. Twitter shortens them using its in-house URL shortening service,

like http://t.co/FCWXoUd8 - such links also enables Twitter to alert users if the link leads out of its domain. From the point of view of text classification, a particular URL is not important. However, presence of a URL can be an important feature. Regular expression for detecting a URL is fairly complex because of different types of URLs **that can be there, but because of Twitter's shortening service, we can** use a relatively simple regular expression.

Regular Expression: (http|https|ftp)://[a-zA-Z0-9\\./]+
Replace Expression: URL

Emoticons

Use of emoticons is very prevalent throughout the web, more so on micro- blogging sites. We identify the following emoticons and replace them with a single word. Table lists the emoticons we are currently detecting. All other emoticons would be ignored.

Emoticons	Examples					
EMOT_SMILEY	:-)	:)	(:	(-:		
EMOT_LAUGH	:-D	:D	X-D	XD	xD	
EMOT_LOVE	<3	:*				
EMOT_WINK	;-)	;)	;-D	;D	(;	(-;
EMOT_FROWN	:-(	:(	(:	(-:		
EMOT_CRY	:,(	:'(	:"(	:((		

Punctuations

Although not all Punctuations are important from the point of view of classification but some of these, like question mark, exclamation mark can also provide information about the sentiments of the text. We replace every word boundary by a list of relevant punctuations present at that point. Table lists the punctuations currently identified. We also remove any single quotes that might exist in the text.

Punctuations	Examples	
PUNC_DOT	.	
PUNC_EXCL	!	¡
PUNC_QUES	?	¿
PUNC_ELLP		

Repeating Characters

People often use repeating characters while using colloquial language, **like "I'm in a hurrryyyyy", "We won, yaaayyyyy!" As our final pre-**processing step, we replace characters repeating more than twice as two characters.

Regular Expression: (.)\1{1,}
Replace Expression: \1\1

3.2.2 Features

A wide variety of features can be used to build a classifier for tweets. The most widely used and basic feature set is word n-grams. However, there's a lot of domain specific information present in tweets that can also be used for classifying them. We have experimented with two sets of features:

Unigrams

Unigrams are the simplest features that can be used for text classification. A Tweet can be represented by a multiset of words present in it. We, however, have used the presence of unigrams in a tweet as a feature set. Presence of a word is more important than how many times it is repeated. Pang found that presence of unigrams yields better results than repetition. This also helps us to avoid having to scale the data, which can considerably decrease training time.

We also observe that the unigrams nicely follow Zipf's law. It states that in a corpus of natural language, the frequency of any word is inversely proportional to its rank in the frequency table.

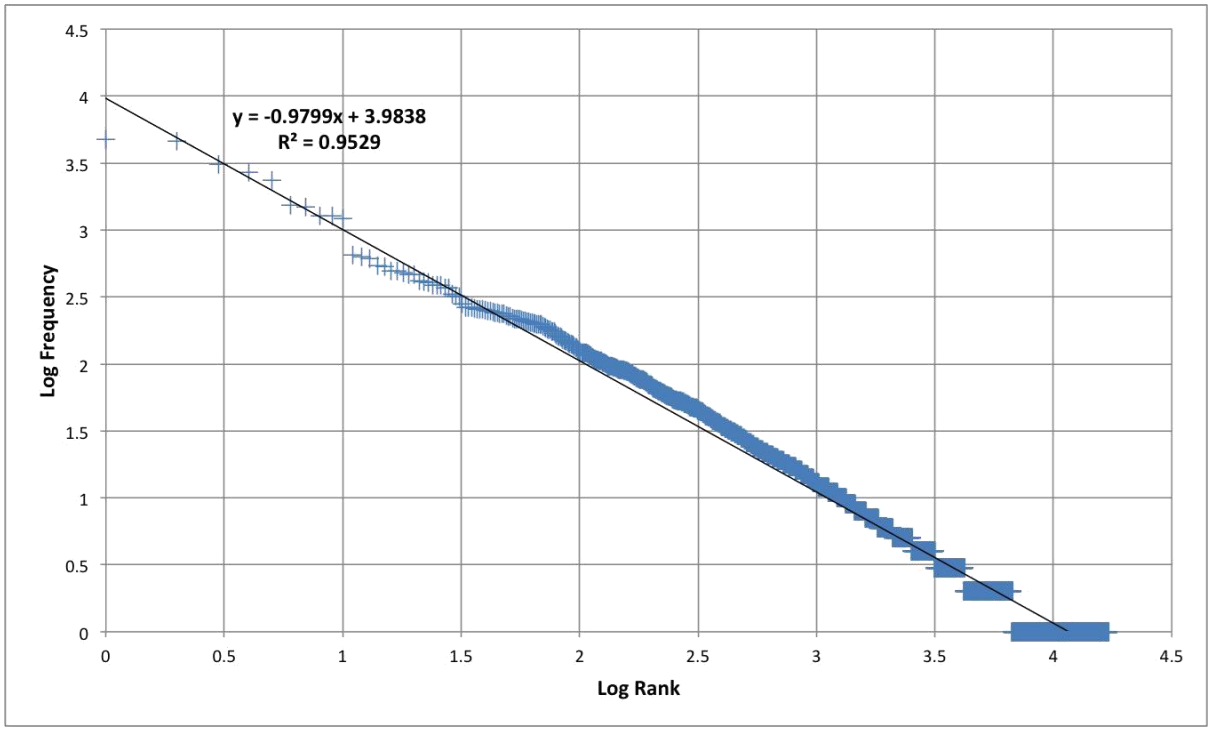


Figure3: Zipf's Law - Log Frequency versus Log Rank plot for unigrams

N-grams

N-gram refers to an n-long sequence of words. Probabilistic Language Models based on Unigrams, Bigrams and Trigrams can be successfully used to predict the next word given a current context of words. In the domain of sentiment analysis, the performance of N-grams is unclear. According to Pang et al., some researchers report that unigrams alone are better than bigrams for classification movie reviews, while some others report that bigrams and trigrams yield better product-review polarity classification.

As the order of the n-grams increases, they tend to be more and more sparse. Based on our experiments, we find that number of bigrams and trigrams increase much more rapidly than the number of unigrams with the number of Tweets. Figure shows the number of n-grams versus number of Tweets. We can observe that bigrams and trigrams increase almost linearly whereas unigrams are increasing logarithmically.

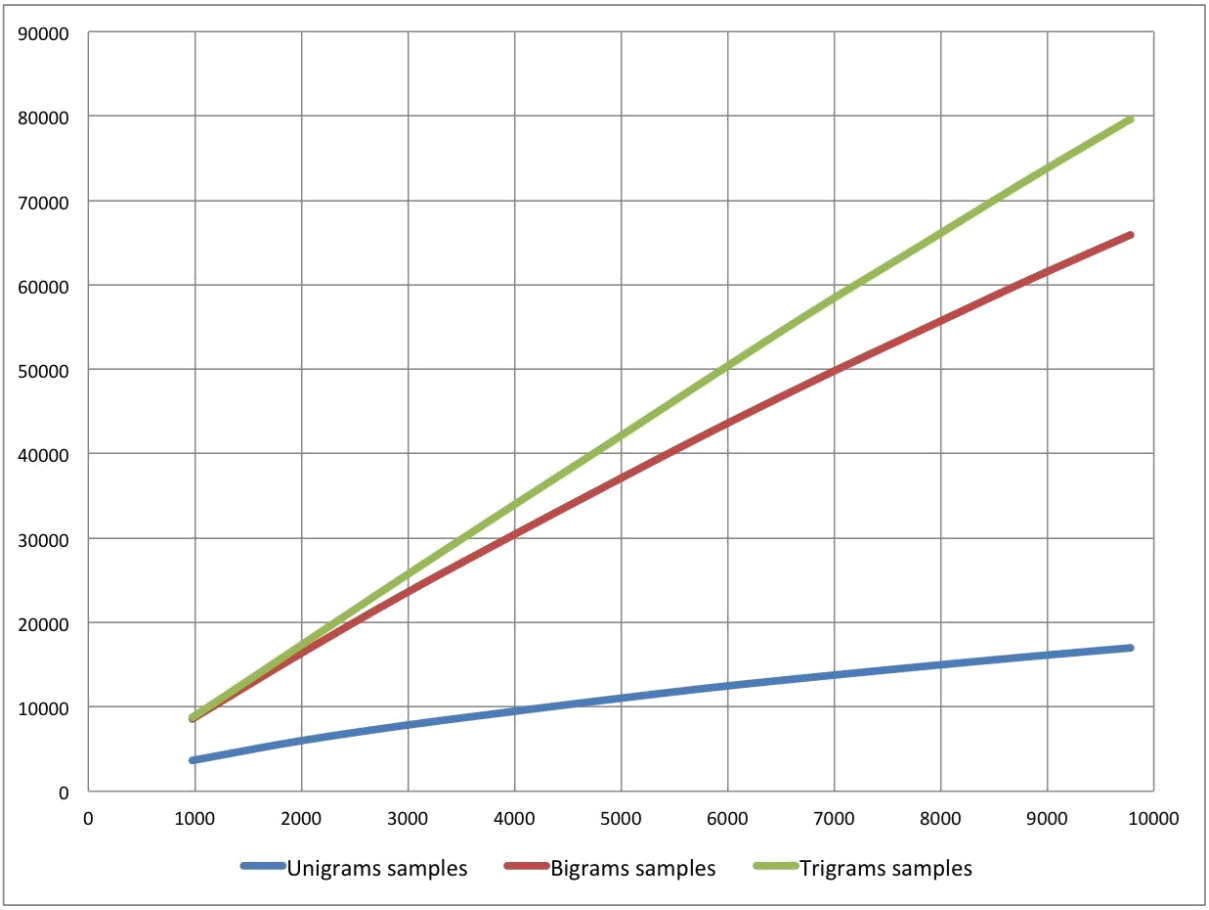


Figure4 : Number of n-grams vs. Number of Tweets

Negation Handling

The need negation detection in sentiment analysis can be illustrated by the difference in the meaning of the phrases, "This is good" vs. "This is not good" However, the negations occurring in natural language are seldom so simple. Handling the negation consists of two tasks – Detection of explicit negation cues and the scope of negation of these words.

Councill et al. look at whether negation detection is useful for sentiment analysis and also to what extent is it possible to determine the exact scope of a negation in the text .They describe a method for negation detection based on Left and Right Distances of a token to the nearest explicit negation cue.

Detection of Explicit Negation Cues

To detect explicit negation cues, we are looking for the following words in Table . The search is done using regular expressions.

S.No.	Negation Cues
1.	never
2.	no
3.	nothing
4.	nowhere
5.	noone
6.	none
7.	not
8.	havent
9.	hasnt
10.	hadnt
11.	cant
12.	couldnt
13.	shouldnt

14.	wont
15.	wouldnt
16.	dont
17.	doesnt
18.	didnt
19.	isnt
20.	arent
21.	aint
22.	Anything ending with "n't"

Table : Explicit Negation Cues

Scope of Negation

Words immediately preceding and following the negation cues are the most negative and the words that come farther away do not lie in the scope of negation of such cues. We define left and right negativity of a word as the chances that meaning of that word is actually the opposite. Left negativity depends on the closest negation cue on the left and similarly for Right negativity. Figure illustrates the left and right negativity of words in a tweet.

```
Words: ['HASH_Skype', 'crash', 'too', 'much', 'PUNC_EXCL',
'not', 'expect', 'this', 'from', 'HASH_MICROSOFT']
Neg_l: [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.9, 0.8, 0.7, 0.6]
Neg_r: [0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 0.0, 0.0, 0.0, 0.0]
```

Figure 9: Scope of Negation

3.2.3 Feature Extraction

Tang et al. [2014] define this feature set as the state-of-the-art feature set, labelling it STATE. We will adopt this definition in our project. The STATE feature set includes most typically used features, and is built up as follows for each tweet:

- N-grams: The presence of word n-**grams (where $1 \leq n \leq 4$) and** character n-**grams (where $3 \leq n \leq 5$).**
- Emoticons: The presence of positive and negative emoticons.
- Negation: The number of individual negated terms within the tweet. Negated terms are also marked as being in a negated context for sentiment lexicon look-ups.

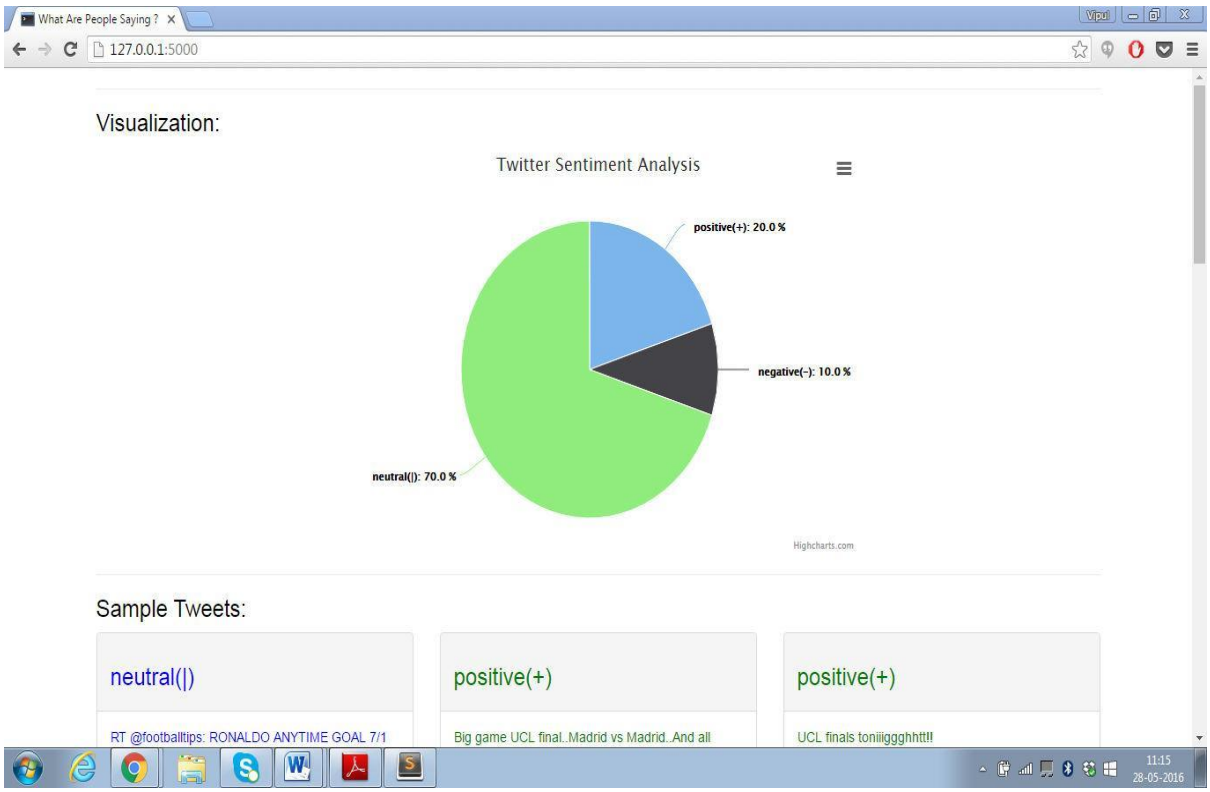
3.2.4 Building the classifier

The classifier step declares which classifier to use, along with its default parameters. We use a Support Vector Machine (SVM) classifier because it is a state-of-the-art learning algorithm proven effective on text categorization tasks, and robust on large feature spaces. We also use the Naïve Bayes algorithm of the nltk library of python since it is effective for tasks having natural language processing and textual analysis.

3.2.5 live tweets classification

In order to apply our classifier to live Twitter data, we wrap the classifier in a web application, depicted in Figure . The application uses the flask web framework2 to allow a user to query Twitter for a search phrase. The resulting tweet hits are classified using a pre-trained classifier, and presented to the user indicating their sentiment polarities. The total distribution of polarity is displayed as a graph to

give the user an impression of the overall opinion of the tweets matching the specified query.



3.2.6 Making of pie chart

We use high charts api for making of pie chart to show the percentage of positive, negative and neutral tweets classified using machine learning algorithms. The tweets are classified live and take minimal time to display the results.

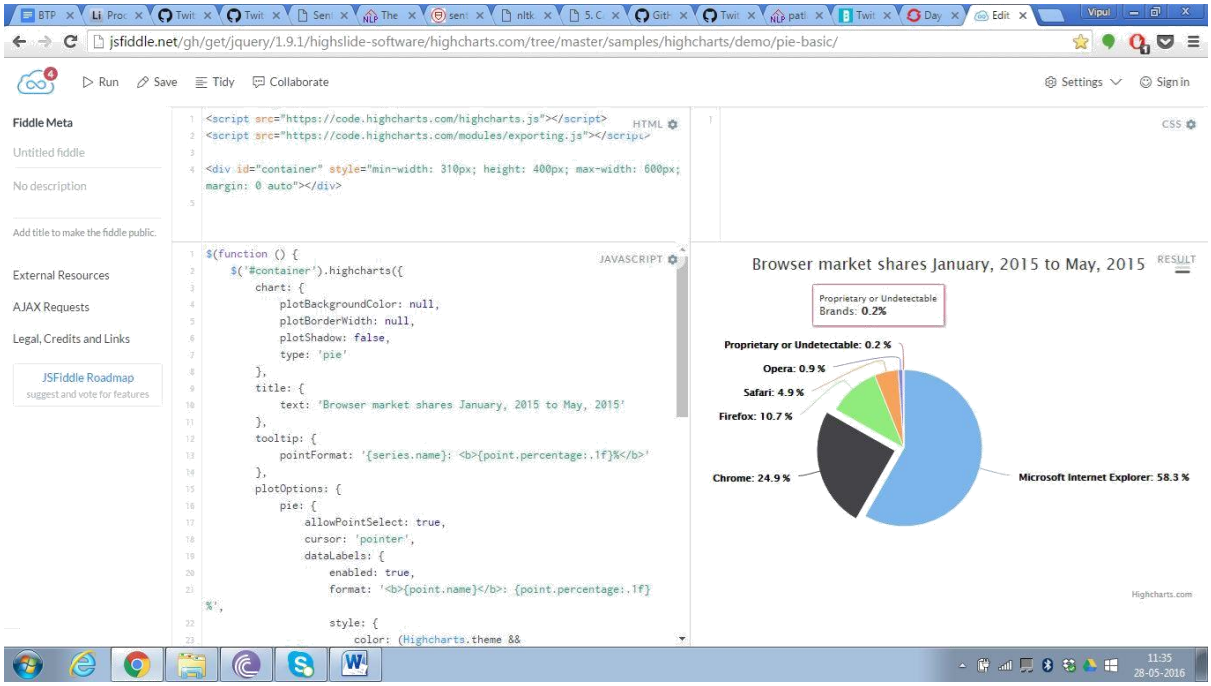
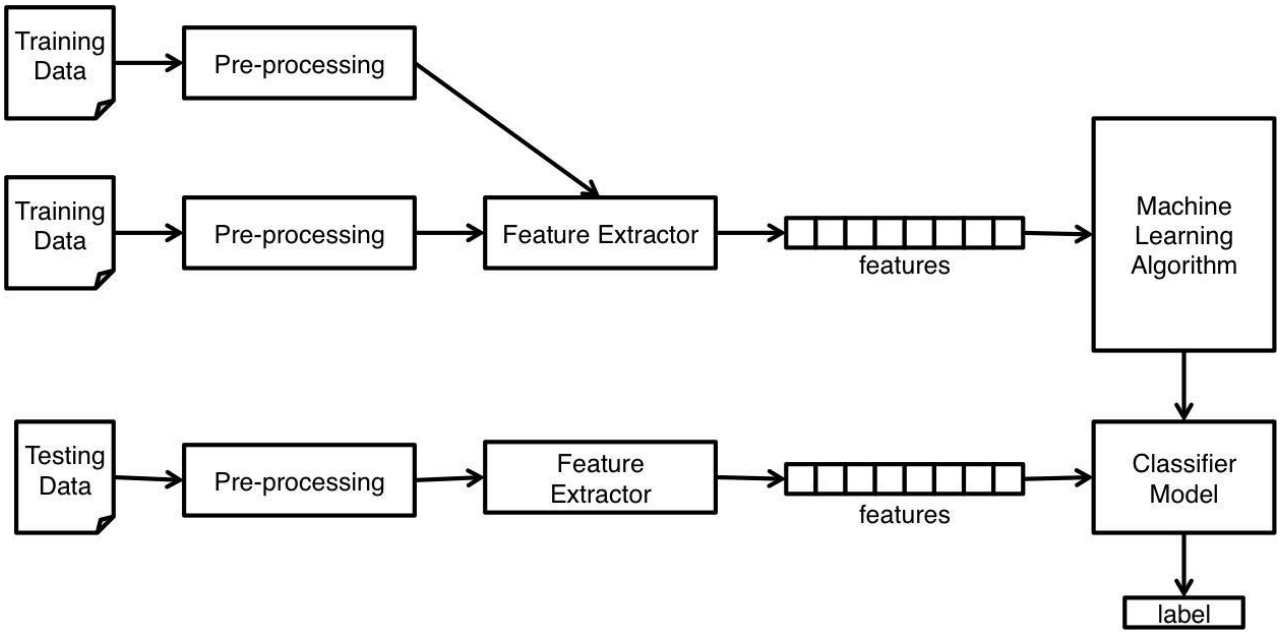


Fig: sample pie chart

4. Architecture



4.1 training data

Training data consists of manually annotated 21, 656 tweets. The format of the data set is .csv. The tweets are labelled as positive, negative, neutral. The data is stored as:

Label, tweet

For example:

Positive, 'this is the best film I have seen!!!! <3'

Negative, 'ahhhh I donnnt like it ☹'

Neutral, 'spiderman is the hero of the city'

Training data is read by csv reader. Csv is the library in python for dealing with data present in csv format.

4.2 Pre-Processing of the tweets

The tweets read by csv reader contain various anomalies, in order to classify them we first convert them into a basic phrase level form by removing repeating characters, RTs, handles etc. The details of the pre-processing step have been described in the above sections.

4.3 Feature extractor

This step describes the features to be used for classification of tweets. Feature extraction is the most important step in any machine learning problem since the quantity and quality of features always have a direct impact on accuracy of the classifier. We in this project have used the following features:

- Unigrams
- Bi-grams
- Tri-grams
- Negation
- Emoticons

The feature extraction is a complicated process and requires a good understanding of the nltk module. The features are stored in a dictionary which is based on the bag-of words- model.

The **bag-of-words model** is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. Recently, the bag-of-words model has also been used for computer vision.

The following models a text document using bag-of-words.
Here are two simple text documents:

```
(1) John likes to watch movies. Mary likes movies too.
(2) John also likes to watch football games.
```

Based on these two text documents, a list is constructed as follows:

```
[
    "John",
    "likes",
    "to",
    "watch",
    "movies",
    "also",
    "football",
    "games",
    "Mary",
    "too"
]
```

This list has 10 distinct words. Using the indexes of the list, each document is represented by a 10-entry vector:

```
(1) [1, 2, 1, 1, 2, 0, 0, 0, 1, 1]
(2) [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]
```

Hence all the words whether single words or a combination of two or more words are stored in a bag where:

Bag[f]=1 represents that this feature is present in the tweet.

Bag[f]=0 represents that this feature is not present .

Based on this features for 21,656 tweets are computed stored in the form of dictionary and serve as a basis for classification.

4.4 machine learning algorithm

The classifier step declares which classifier to use, along with its default parameters. We use a naïve Bayes (nltk) classifier because it is a state-of-the-art learning algorithm proven effective on text categorization tasks, and robust on large feature spaces.

More details about the naïve Bayes algorithms and other algorithms used are described in above sections.

4.5 storing the classifier as a model

Training classifiers and machine learning algorithms can take a very long time, especially if you're training against a larger data set. Instead, what we can do is use the Pickle module to go ahead and serialize our classifier object, so that all we need to do is load that file in real quick.

So, how do we do this? The first step is to save the object. To do this, first you need to import pickle at the top of your script, then, after you have trained with .train() the classifier, you can then call the following lines:

```
save_classifier = open("naivebayes.pickle","wb")
pickle.dump(classifier, save_classifier)
save_classifier.close()
```

This opens up a pickle file, preparing to write in bytes some data. Then, we use pickle.dump() to dump the data. The first parameter to pickle.dump() is what are you dumping, the second parameter is where are you dumping it.

After that, we close the file as we're supposed to, and that is that, we now have a pickled, or serialized, object saved in our script's directory!

Next, how would we go about opening and using this classifier? The .pickle file is a serialized object, all we need to do now is read it into

memory, which will be about as quick as reading any other ordinary file. To do this:

```
classifier_f = open("naivebayes.pickle", "rb")
classifier = pickle.load(classifier_f)
classifier_f.close()
```

Here, we do a very similar process. We open the file to read as bytes. Then, we use pickle.load() to load the file, and we save the data to the classifier variable. Then we close the file, and that is that. We now have the same classifier object as before!

4.6 testing of model

The model developed for opinion mining needs to be tested for evaluating its performance. The data set used for the project is divided into training_ set and testing_set respectively. Various metrics available like accuracy, f value, precision recall etc. are computed .Various algorithms like naïve Bayes, SVM ,Max entropy are compared on the basis of these metrics. Further details are in the subsequent section.

4.7 Live Tweet Classification

We use twitter rest API v1.1 for collecting live tweets.

The REST APIs provide programmatic access to read and write Twitter data. Author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

GET search/tweets

Returns a collection of relevant Tweets matching a specified query.

Resource Information

Response formats	JSON
Requires authentication?	Yes

Rate limited?	Yes

Parameters:

q:

A UTF-8, URL-encoded search query of 500 characters maximum, including operators. Queries may additionally be limited by complexity.

result_type:

Specifies what type of search results you would prefer to receive. The current default is "mixed." Valid values include:
* mixed: Include both popular and real time results in the response.
* recent: return only the most recent results in the response
* popular: return only the most popular results in the response.

Count:

The number of tweets to return per page, up to a maximum of 100. Defaults to 15. This was formerly the "rpp" parameter in the old Search API.

API keys are needed for authentication and can be obtained from the site https://dev.twitter.com/apps/new

4.8 displaying the results

The application uses the flask web framework2 to allow a user to query Twitter for a search phrase. The resulting tweet hits are classified using a pre-trained classifier, and presented to the user indicating their sentiment polarities. The total distribution of polarity is displayed as a graph to give the user an impression of the overall opinion of the tweets matching the specified query.

Features of flask:

- Contains development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja2 templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired

The **Web Server Gateway Interface** (**WSGI**) is a specification for simple and universal interface between web servers and web applications or frameworks for the Python programming language. It was originally specified in PEP 333 authored by Phillip J. Eby, and published on 7 December 2003. It has since been adopted as a standard for Python web application development. The latest version of the specification is v1.0.1, also known as PEP 3333, published on 26 September 2010.

The WSGI has two sides: the "server" or "gateway" side (often a web server like Apache or Nginx), and the "application" or "framework" side (the python script itself). To process a WSGI request, the server side executes the application and provides environment information and a call-back function to the application side. The application processes the request, and returns the response to the server side using the call-back function it was provided.

Between the server and the application, there may be a *WSGI middleware*, which implements both sides of the API. The server receives a request from a client and forwards it to the middleware. After processing it sends a request to another WSGI application whose response is forwarded to the client via the middleware and ultimately via the server. There may be multiple middle wares, thus forming a stack of WSGI-compliant applications.

Flask apps run on localhost and provide debugging facilities as well. For analytics we use highcharts API which is based on JavaScript and gives a plethora of options.

5. Experiments

We train 90% of our data using different combinations of features and test them on the remaining 10%. We take the features in the following combinations.

- Only unigrams.
- Only unigrams + bi-grams +tri-grams.
- Uni-grams+negation.
- Uni-grams + bi-grams + tri-grams +negation.
- Inclusion and exclusion of stop words.

The task of classification of a tweet can be done with respect to 2 algorithms **– naïve Bayes and SVM**.

Table 2: Training and Testing data

Total size of dataset	21,656
Size of training set	18000 tweets
Size of testing dataset	3240 tweets

Table 3: Training dataset

Positive tweets	8000
Negative tweets	8000
Neutral tweets	2000

Table 4 : Testing dataset

Positive tweets	1200
Negative tweets	1200
Neutral tweets	800

5.1 Naïve Bayes

Naive Bayes classifier is the simplest and the fastest classifier. Many researchers claim to have gotten best results using this classifier.

For a given tweet, if we need to find the label for it, we find the probabilities of all the labels, given that feature and then select the label with maximum probability.

labelNB : = argmaxlabel P(label|features)

In order to find the probability for a label, this algorithm first uses the Bayes rule to express P(label - features) in terms of P(label) and P(featureslabel) as,

$$P(label|features) = P(label) * P(features|label)$$

Making the `naive' assumption that all the features are independent,

$$P(label|features) = P(label) * P(f1|label) * ... * P(fn|label)$$

Rather than computing P(featues) explicitly, we can just calculate the denominator for each label, and normalize them so they sum to one:

$$\mathbf{P(label|features) = P(label) * P(f1|label) * ... * P(fn|label)}$$

The results from training the Naive Bayes classifier are shown below:

Feature used	Accuracy in%
unigrams	**74.86**
unigrams + bi-grams +tri-grams	**75.77**
Uni-grams+negation	**76.29**
Uni-grams + bi-grams + tri-grams +negation	**77.59**
Inclusion and exclusion of stop words	**75.46(accuracy decreases)**

Table 5 : accuracy of naïve Bayes

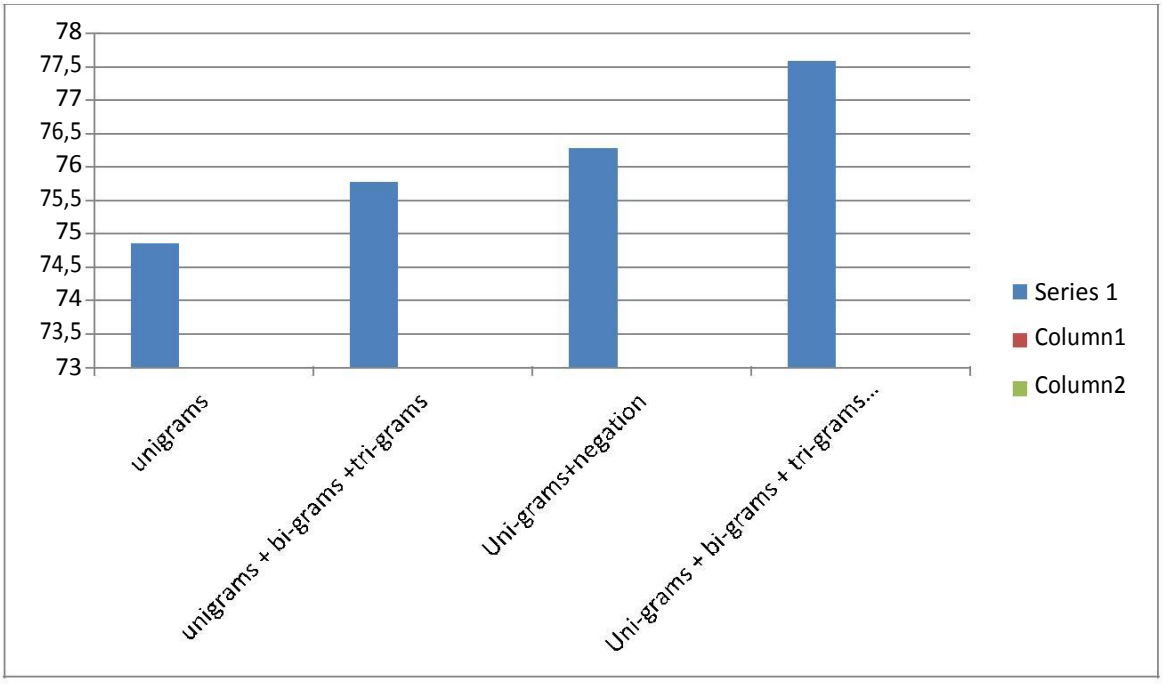


Figure 5: results of naïve bayes

5.2 SVM(SUPPORT VECTOR MACHINES)

The Support Vector Machine (SVM) classification algorithm was formally described by Cortes and Vapnik [1995]. The algorithm considers data points based on their spatial location, and attempts to

split the feature space into optimal class segments. This division of the feature space is referred to as training the machine. A trained SVM can then be used for classification of new examples by assigning them a class based on which segment of the feature space they are located in. In its basic form, it is an algorithm for linear classification of binary problems.

We are given a training dataset of n points of the form

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$$

where the y_i **are either 1 or −1, each indicating the class to which** the point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\vec{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized.

Any hyperplane can be written as the set of points $\vec{x}$ satisfying

$$\vec{w} \cdot \vec{x} - b = 0,$$

The results from training the LINEAR SVC classifier are shown below:

Feature used	**Accuracy in%**
unigrams	**70.66**
unigrams + bi-grams +tri-grams	**72.71**
Uni-grams+negation	**71.57**
Uni-grams + bi-grams + tri-grams +negation	**73.27**
Inclusion and exclusion of stop words	**71.69**

Table 6: SVM classifier

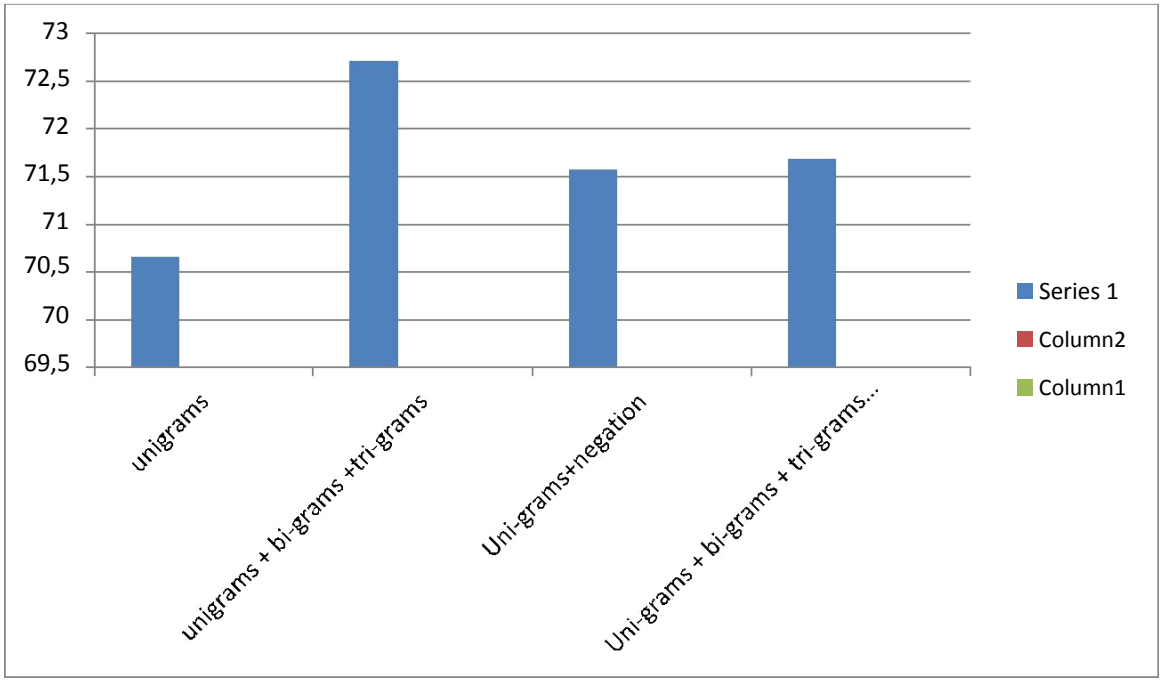


Figure 6: comparison of various features

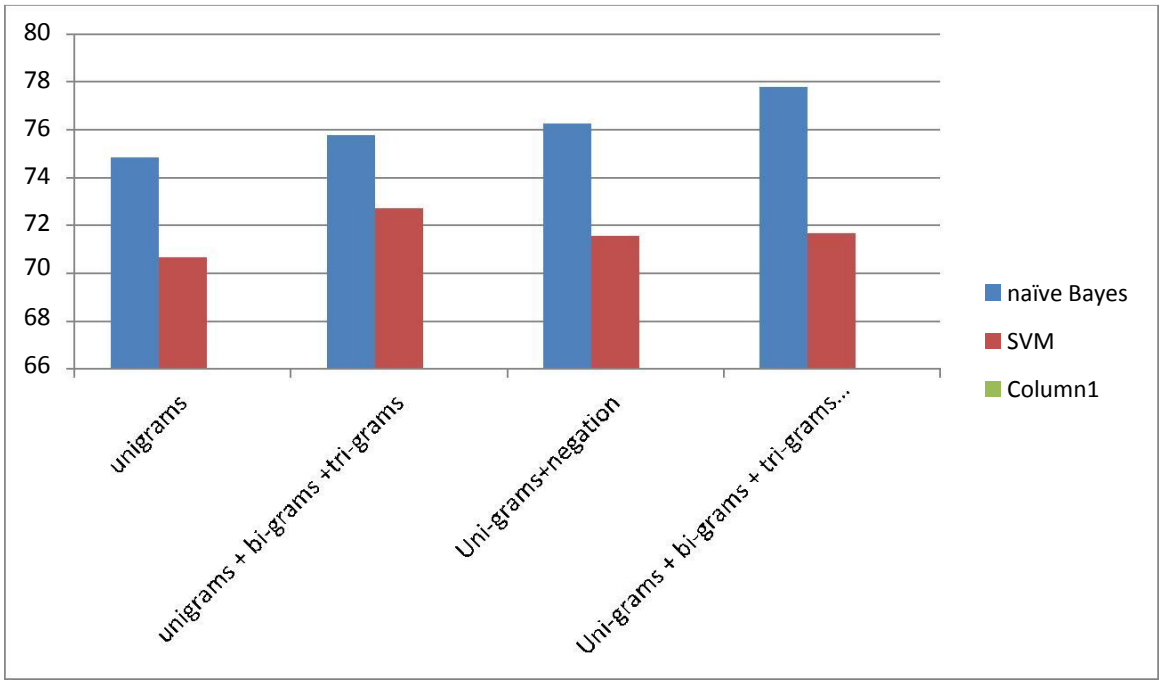


Figure 7: svm vs Naïve Bayes

6. Results

In this paper, we create a sentiment classifier for twitter using labelled data sets. We also investigate the relevance of using stopwords step classifier and negation detection for the purpose of sentiment analysis.

Accuracy of the classifier increases if we use negation detection or introduce bigrams and trigrams. Thus we can conclude that both Negation Detection and higher order n-grams are useful for the purpose of text classification. However, if we use both n-grams and negation detection, the accuracy falls marginally.. In general, Naive Bayes Classifier performs better than SVM Classifier.

Also inclusion of stop words have led to increase of accuracy.

We achieve the best accuracy of **77.59%** in the case of Unigrams + Bigrams + Trigrams, trained on Naive Bayes Classifier.

7. Conclusion

This chapter contains an evaluation of the project and the conclusions we draw from the results.

7.1. Evaluation

At the beginning of this project, we formulated five goals, described in Section 1.3. They form a natural chain of dependency, as each goal builds on all the previous. In this section we evaluate the degree to which each goal has been accomplished

G1: Research the State-of-the-Art in Twitter Sentiment Analysis

The area of Twitter sentiment analysis (TSA) was explored, with special attention to the several shared tasks that have been hosted for this field. The related works discovered formed the basis for most of the work performed throughout the project.

G2: Create a Twitter Corpus Annotated for Opinion mining

A Twitter corpus annotated for the presence of sentiment, the, was successfully created .The corpus contained 21k tweets with three sentiments namely positive, negative and neutral.

G3: Develop a gettweets script using twitter API

The script was successfully created. Tweets obtained were stored in json format. The working of script required API keys and secret **tokens which were taken from developer's account of twitter.**

G4: Develop a Twitter Sentiment Classifier

A sentiment classifier for Twitter data was developed, incorporating several features that benefit from the NSD system. The results confirm that taking negation into account in general improves the sentiment classification performance significantly, and that using a sophisticated NSD system slightly improves the performance further. Accuracy of the classifier was close to 78%.

G5: Displaying the analytics using a pie chart

The analytics were successfully displayed using the highcharts API. Front end was made in javascript, HTML. The user interface of the web application was easy to use. Tweets were classified real time and displayed.

8. Future Work

A Twitter corpus annotated for both sentiment and negation would be a valuable resource to measure the effects of linguistic negation in TSA. This would allow for evaluating the performance of a sentiment classifier, and the impact of different features, with gold standard negation scope detection, thus displaying the maximum possible performance gain with perfect negation handling. This could, for instance, be done by applying the negation annotation system developed in this project.

Building a classifier for Hindi tweets There are many users on Twitter that use primarily Hindi language. The approach discussed here can be used to create a Hindi language sentiment classifier. **Improving Results using Semantics**

Analysis Understanding the role of the nouns being talked about can help us better classify a given tweet. For example, "Skype often crashing: microsoft, what are you doing?" Here Skype is a product and Microsoft is a company. We can use semantic labellers to achieve this.

9. References

9.1 Research Papers

- Twitter Sentiment Analysis :Exploring the Effects of Linguistic Negation by Jørgen Faret Johan Reitan.
- Sentiment analysis of twitter feeds by Prof Yogesh Garg.
- Isaac G Councill, Ryan McDonald, and Leonid Velikovich. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the workshop on negation and speculation in natural language processing*, pages 51-59. Association for Computational Linguistics, 2010.
- Twitter as a Corpus for Sentiment Analysis and Opinion Mining Alexander Pak, Patrick Paroubek

9.2 Websites

- http://www.laurentluce.com/posts/twitter-sentiment-analysis- using-python-and-nltk/
- www.stackoverflow.com
- http://sentiment.christopherpotts.net/index.html
- www.udacity.com