

## INDEX

<b>S. No.</b>	<b>Topic</b>	<b>Page Number</b>
<b>1</b>	Introduction	<b>4</b>
<b>2</b>	Problem Statement	<b>9</b>
<b>3</b>	Implementation	<b>10</b>
	I. Software Tools Used	
	II. Classifiers	
	III. About The Implementation	
	IV. Program Architecture	
<b>4</b>	Results	<b>28</b>
<b>5</b>	Applications And Future Work	<b>34</b>
<b>6</b>	Refrences	<b>35</b>

## **I.) Introduction**

Social Media tools such as Facebook, Twitter and LinkedIn do not have automated means to capture, control and appropriately manage and workflow process complaints. SMCWATSI (Social Media Complaint workflow automation tool using sentiment intelligence) provides an end to end complaint management tool that automatically captures incoming complaints at their social media source and immediately translates the message intent and directly the complaints to workflow and call centre queues where they can be instantly assessed, prioritized and resolved.

“What other people think” has always been an important piece of information for most of us during the decision-making process.<sup>[1]</sup> Long before awareness of the World Wide Web became wide spread, many of us asked our friends to recommend an auto mechanic or to explain who they were planning to vote for in local elections, requested reference letters regarding job applicants from colleagues, or consulted Consumer Reports to decide what dishwasher to buy. But the Internet and the Web have now (among other things) made it possible to find out about the opinions and experiences of those in the vast pool of people that are neither our personal acquaintances nor well-known professional critics - that is, people we have never heard of. The body of work we review is that which deals with the computational treatment of (in alphabetical order) opinion, sentiment, and subjectivity in text. Such work has come to be known as opinion mining, sentiment analysis, and/or subjectivity analysis. The phrases review mining and appraisal extraction have been used, too, and there are some connections to affective computing, where the goals include enabling computers to recognize and express emotions.<sup>[2]</sup>

Most banks are not delivering such services today. The average response time to customer inquiries on social media from banks and financial services providers is 10 hours, and more than two out of every customer inquiries go unanswered. To sharpen their competitive advantage, banks must continuously improve their customer service capabilities by offering better responsive times and response rates through social media, as well as personalized services to customers by interpreting the data that is continuously generated on social media. By analysing the large volumes of data available on social media, banks can extract key insights that will enable them to improve product and service development, customer service, marketing, risk management and business performance. Since social media is all about the customer experience, banks need to build their social media strategies around the customer to drive loyalty, revenue and profitability.<sup>[3]</sup> With approximately two billion people using social

media around the world, banks must seriously consider how to engage with customers on social channels. Customers today expect the companies they do business with to listen, respond and offer services through social media; consider that inbound consumer engagement with brands in social channels is growing nearly nine times faster than social networks themselves.

### ***The Power of Social Media***

As customers increasingly use social media to share opinions on financial products and services, banks must listen, learn and respond, as well as incorporate their social activities into their overall corporate strategies. In most cases, this requires banks to rethink their core business strategies to make them more customer-centric. Traditionally, banks have employed a “push” strategy to communicate their offerings to customers, through advertising, direct mail, point-of-sale displays or face-to-face interactions. However, the industry focus has shifted from “customer service” to “customer engagement,” which requires a two-way mode of communication. To sharpen their engagement capabilities, banks need to enhance their understanding of customers by using social analytics to gain deep insights into customer behaviour, sentiments and needs.<sup>[3]</sup>

For instance, using social analytics, organizations can identify, analyse and interpret interactions and associations over social media, measure their impact and strengthen decision-based marketing. With the rapid development of text analytics and sentiment analytics, banks can uncover insights into customer behaviour that were otherwise unknown to them. In the meantime, banks also need to ensure that their traditional business intelligence systems, which primarily focus on historical reporting, dashboards and advanced visualization, are integrated with emerging social analytics tools and techniques. The input from social customer relationship management enables banks to develop profound knowledge about their customers, including sentiments, wishes and needs, resulting in improved engagement and intelligent decision-making. Banks can also use social media insights to create pricing models for loans and other banking products, and they can combine traditional scoring elements with data available in the public domain, such as Twitter, Facebook and other social networking sites, to ascertain creditworthiness and price loans.<sup>:[3]</sup>

Vantage Credit Union in St. Louis is an example of a bank that is using social media insights to its advantage. The organization is monitoring customer conversations on online communities and heeding their recommendations and complaints to improve customer

service. It has also developed a way for customers to securely conduct banking while on Twitter. Customers can text the credit union to retrieve information on their accounts — such as balances, holds and cleared checks — and even transfer money between accounts.

Clearly, to unleash the full potential of using social media channels, banks need to look beyond mere e-reputation and branding, to customer engagement and the ability to predict customer behaviour from insights gained from social data. Using these insights, banks can bolster their risk management capabilities by identifying potential defaulters and thwarting money laundering activities, and they can improve customer satisfaction by using social analytics to better understand the products and services that customer's desire and even personalize them, based on individual customer preferences. The effective use of social media tools can not only improve customer satisfaction, but it can also drive business expansion through acquisition of new customers and increased share of the customer's wallet.<sup>[3]</sup>

### ***Gaining customer insight***

Monitoring posts, likes and comments on social media can provide banks with a general idea of customer perception regarding their products and services. Using analytics to dissect that data can provide invaluable information regarding customer behaviour and sentiment, thereby allowing banks to design more personalized products and services.<sup>[3]</sup>

### ***Improving customer service***

Monitoring social channels will highlight challenges that customers are facing and help banks take steps to address issues before they degrade the customer relationship. Moreover, banks can use social analytics tools to anticipate how customers will respond to new strategies and take proactive steps to mitigate complaints.<sup>[3]</sup>

### ***Designing new internal processes***

Banks will require new processes to encourage collaboration across dispersed teams, business groups and divisions. Doing so will enable financial institutions to arm the right individual(s) with the right information, at the right time, in the right format needed to address customer preferences at every stage of their relationship?

The social media sphere is rapidly evolving. Banks and financial institutions that are quick to synergize their business operations with their social media strategies will be more responsive to customer needs and better able to offer customers the best experience. Nonetheless, there are also pitfalls to overcome, and a first mover advantage may not always translate into market leadership. As such, social media can be likened to a double edged sword — on the one hand, it can raise the spectre of security and privacy threats for banks and their customers, while on the other, and it most assuredly will generate enormous value. What is clear is that social strategies have become a mandate for the banking industry, and the question is how — not whether — banks will need to embark on their own social journey to secure their place in the future.<sup>[3]</sup>

➤ A note on Terminology: Opinion Mining, Sentiment Analysis, Subjectivity and all that.

*“The beginning of wisdom is the definition of terms”*

- Socrates.

The body of work we review is that which deals with the computational treatment of (in alphabetical order) *opinion*, *sentiment*, and *subjectivity* in text. Such work has come to be known as *opinion mining*, *sentiment analysis*, and/or *subjectivity analysis*. The phrases *review mining* and *appraisal extraction* have been used, too, and there are some connections to *affective computing*, where the goals include enabling computers to recognize and express emotions. This proliferation of terms reflects differences in the connotations that these terms carry, both in their original general-discourse usages and in the usages that have evolved in the technical literature of several communities. In 1994, Wiebe<sup>[4]</sup>, influenced by the writings of the literary theorist Banfield, centered the idea of *subjectivity* around that of *private states*, defined by Quirk et al. as states that are not open to objective observation or verification. Opinions, evaluations, emotions, and speculations all fall into this category; but a canonical example of research typically described as a type of subjectivity analysis is the recognition of opinion-oriented language in order to distinguish it from objective language. While there has been some research self-identified as subjectivity analysis on the particular application area of determining the value judgments (e.g., “four stars” or “C+”) expressed in the evaluative opinions that are found, this application has not tended to be a major focus of such work. The term *opinion mining* appears in a paper by Dave et al. that was published in the proceedings

of the 2003 WWW conference; the publication venue may explain the popularity of the term within communities strongly associated with Web search or information retrieval. According to Dave et al., the ideal opinion-mining tool would “process a set of search results for a given item, generating a list of product attributes (quality, features, etc.) and aggregating opinions about each of them (poor, mixed, good).” Much of the subsequent research self-identified as opinion mining fits this description in its emphasis on extracting and analyzing judgments on various aspects of given items. However, the term has recently also been interpreted more broadly to include many different types of analysis of evaluative text. The history of the phrase *sentiment analysis* parallels that of “opinion mining” in certain respects. The term “sentiment” used in reference to the automatic analysis of evaluative text and tracking of the predictive judgments therein appears in 2001 papers by Das and Chen and Tong, due to these authors’ interest in analyzing market sentiment. It subsequently occurred within 2002 papers by Turney and Pang et al., which were published in the proceedings of the annual meeting of the Association for Computational Linguistics (ACL) and the annual conference on Empirical Methods in Natural Language Processing (EMNLP). Moreover, Nasukawa and Yi entitled their 2003 paper, “Sentiment analysis: Capturing favorability using natural language processing”, and a paper in the same year by Yi et al. was named “Sentiment Analyzer: Extracting sentiments about a given topic using natural language processing techniques.” These events together may explain the popularity of “sentiment analysis” among communities self-identified as focused on NLP. A sizeable number of papers mentioning “sentiment analysis” focus on the specific application of classifying reviews as to their polarity (either positive or negative), a fact that appears to have caused some authors to suggest that the phrase refers specifically to this narrowly defined task. However, nowadays many construe the term more broadly to mean the computational treatment of opinion, sentiment, and subjectivity in text. **Thus, when broad interpretations are applied, “sentiment analysis” and “opinion mining” denote the same field of study (which itself can be considered a sub-area of subjectivity analysis).** We have attempted to use these terms more or less interchangeably in this survey. This is in no small part because we view the field as representing a unified body of work, and would thus like to encourage researchers in the area to share terminology regardless of the publication venues at which their papers might appear.

## II.) Problem Statement

The goal of the project is to do sentiment analysis of social media posts to extract complaints and define a workflow for their resolution. The main idea of the project is to create a model to predict the sentiment of the post using popular classification algorithms like Bayes classification, Support Vector Machine, etc. Moreover we also used the model from our previous minor project to calculate emotion values of the posts and then find out the overall feedback of the Customers on the particular Social Media Channel.

The goal of the project is to use social channels to deliver faster and more effective customer service and customized financial advice; share knowledge about regulations; and provide a feedback mechanism about banking products and services.

Our *Social Media Complaint workflow automation tool using sentiment intelligence* provides automated complaint capture, receipt and extraction of information required to process each individual complaint, whether it was originally received on Facebook, Twitter or any other Social Networking Websites.

### III.) Implementation

#### ➤ Software Tools Used

##### 1. Python

**Python** is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python is multipurpose, ie, it is not specialized to a specific target of users (like R for statistics, or PHP for web programming). It is extended through modules and libraries that hook very easily into the C programming language. For our purpose, we used a bunch of python libraries like tweepy, tkinter etc. A brief explanation for all of them has been given below. Python can be used for any programming task, from GUI programming to web programming with everything else in between. It's quite efficient, as much of its activity is done at the C level. For our purpose we have used the Python 3.4.3.

##### 2. Python csv module

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. There is no “CSV standard”, so the format is operationally defined by the many applications which read and write it. The lack of a standard means that subtle differences often exist in the data produced and consumed by different applications. These differences can make it annoying to process CSV files from multiple sources. Still, while the delimiters and quoting characters vary, the overall format is similar enough that it is possible to write a single module which can efficiently manipulate such data, hiding the details of reading and writing the data from the programmer. The **csv** module implements classes to read and write tabular data in CSV format. It allows programmers to say, “Write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats. The **csv** module’s **reader** and **writer** objects read and write sequences. Programmers can also read and write data in dictionary form using the DictReader and DictWriter classes.



### 3. PyCharm: Integrated Development Environment

**PyCharm** is an Integrated Development Environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django. PyCharm is developed by the Czech company Jet Brains. It is cross-platform working on Windows, Mac OS X and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License. PyCharm Community Edition is less extensive than the Professional Edition.

### 4. NLTK

**NLTK** is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, and an active discussion forum. NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project. NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.” In our project, we used NLTK in cleaning module as well as Part of Speech Tagging. NLTK helped in data cleaning by providing the list of stopwords. NLTK further helped in the tagging of Part of speech such as verbs, adjectives etc. NLTK provides several modules for these type of tasks, such as PunktTokenizer module for tokenizing the given string.

### 5. Tkinter

The **Tkinter** module (“Tk interface”) is the standard Python interface to the Tk GUI toolkit. Both Tk and Tkinter are available on most Unix platforms, as well as on Windows and Macintosh systems. Starting with the 8.0 release, Tk offers native look and feel on all platforms. Tkinter consists of a number of modules. The Tk interface is provided by a binary extension module named **\_tkinter**. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or

DLL), but might in some cases be statically linked with the Python interpreter. The public interface is provided through a number of Python modules. The most important interface module is the Tkinter module itself. To use Tkinter, all we need to do is to import the Tkinter module. We made extensive use of Tkinter to provide an interactive graphical user interface to the user.

## 6. Pickle

The pickle module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” or “flattening”. In our problem context, we have used pickling to save the classifiers as trained because to train these classifiers usually takes a lot of time, so time and again we don’t need to train these classifiers and we can directly unpickle them if any of them already exist in a file. If any of the classifier doesn’t exist already, we need to train it and then save it for future purpose.

## 7. Scikit-learn

**Scikit-learn** (formerly scikits.learn) is an open source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

## 8. Facebook Graph API

The Graph API is the core of Facebook Platform, enabling developers to read from and write data into Facebook. The Graph API presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph (e.g., people, photos, events, and pages) and the connections between them (e.g., friend relationships, shared content, and photo

tags).Facebook authentication enables developers' applications to interact with the Graph API on behalf of Facebook users, and it provides a single-sign on mechanism across web, mobile, and desktop apps.The Open Graph protocol enables developers to integrate their pages into the social graph. These pages gain the functionality of other graph objects including profile links and stream updates for connected users.

## **9. Tweepy**

The python tweepy library has been used for interacting with the twitter and to download the twitter data.It is basically a data streaming library for tweeter.Tweepy supports oauth authentication. Authentication is handled by the tweepy.AuthHandlerclass.Tweepy provides access to the well documented Twitter API. With tweepy, it's possible to get any object and use any method that the official Twitter API offers.Main Model classes in the Twitter API are Tweets, Users, Entities and Places. Access to each returns a JSON-formatted response and traversing through information is very easy in Python.<sup>[10]</sup>

## **10. TweepyStreamingAPI**

One of the main usage cases of tweepy is monitoring for tweets and doing actions when some event happens. Key component of that is the StreamListener object, which monitors tweets in real time and catches them.<sup>[9]</sup>

## **11. Tweepy REST API**

Twitter provides the REST search API for searching tweets from Twitter's search index. This is different than using thestreaming filter API, in that the later is real-time and starts giving you results from the point of query, while the former is retrospective and will give you results from past, up to as far back as the search index goes (usually last 7 days). While the streaming API seems like the thing to use when you want to track a certain query in real time, there are situations where you may want to use the regular REST search API. We may also want to combine the two approaches, i.e. start 2 searches, one using the streaming filter API to go forward in time and one using the REST search API to go backwards in time, in order to get some on-going and past context for your search term.<sup>[8]</sup>

## ➤ Classifiers

The automated categorization (or classification) of texts into predefined categories has witnessed a booming interest in the last 10 years, due to the increased availability of documents in digital form and the ensuing need to organize them. In the research community the dominant approach to this problem is based on machine learning techniques: a general inductive process automatically builds a classifier by learning, from a set of preclassified documents, the characteristics of the categories. The advantages of this approach over the knowledge engineering approach (consisting in the manual definition of a classifier by domain experts) are a very good effectiveness, considerable savings in terms of expert labour power, and straightforward portability to different domains. This survey discusses the main approaches to text categorization that fall within the machine learning paradigm. We will discuss in detail issues pertaining to three different problems, namely, document representation, classifier construction, and classifier evaluation.<sup>[11]</sup>

Content-based classification is classification in which the weight given to particular subjects in a document determines the class to which the document is assigned. It is, for example, a rule in much library classification that at least 20% of the content of a book should be about the class to which the book is assigned. In automatic classification it could be the number of times given words appears in a document.

Request-oriented classification (or -indexing) is classification in which the anticipated request from users is influencing how documents are being classified. The classifier asks himself: “Under which descriptors should this entity be found?” and “think of all the possible queries and decide for which ones the entity at hand is relevant”.<sup>[11]</sup>

Request-oriented classification may be classification that is targeted towards a particular audience or user group. For example, a library or a database for feminist studies may classify/index documents differently when compared to a historical library. It is probably better, however, to understand request-oriented classification as *policy-based classification*: The classification is done according to some ideals and reflects the purpose of the library or database doing the classification. In this way it is not necessarily a kind of classification or indexing based on user studies. Only if empirical data about use or users are applied should request-oriented classification be regarded as a user-based approach.

Automatic document classification tasks can be divided into three sorts: supervised document classification where some external mechanism (such as human feedback) provides

information on the correct classification for documents, unsupervised document classification (also known as document clustering), where the classification must be done entirely without reference to external information, and semi-supervised document classification, where parts of the documents are labelled by the external mechanism. There are several software products under various license models available.<sup>[11]</sup>

## **1. Naïve Bayes Classifier**

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s, and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.<sup>[12]</sup>

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, Naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method; Russell and Norvig note that "[naive Bayes] is sometimes called a Bayesian classifier, a somewhat careless usage that has prompted some Bayesians to call it the idiot Bayes model."<sup>[13]</sup>

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class

variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.<sup>[13,14]</sup>

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.

An advantage of naive Bayes is that it only requires a small amount of training data to estimate the parameters necessary for classification.

### *Probabilistic model*

---

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector  $\mathbf{x} = (x_1, \dots, x_n)$  representing some  $n$  features (independent variables), it assigns to this instance probabilities

$$p(C_k | x_1, \dots, x_n)$$

for each of  $K$  possible outcomes or *classes*.

The problem with the above formulation is that if the number of features  $n$  is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}.$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on  $C$  and the values of the features  $F_i$  are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(C_k) p(x_1, \dots, x_n | C_k) \\ &= p(C_k) p(x_1 | C_k) p(x_2, \dots, x_n | C_k, x_1) \\ &= p(C_k) p(x_1 | C_k) p(x_2 | C_k, x_1) p(x_3, \dots, x_n | C_k, x_1, x_2) \\ &= p(C_k) p(x_1 | C_k) p(x_2 | C_k, x_1) \dots p(x_n | C_k, x_1, x_2, x_3, \dots, x_{n-1}) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature  $F_i$  is conditionally independent of every other feature  $F_j$  for  $j \neq i$ , given the category  $C$ . This means that

$$\begin{aligned} p(x_i | C_k, x_j) &= p(x_i | C_k), \\ p(x_i | C_k, x_j, x_q) &= p(x_i | C_k), \\ p(x_i | C_k, x_j, x_q, x_l) &= p(x_i | C_k), \end{aligned}$$

and so on, for  $i \neq j, q, l$ . Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

This means that under the above independence assumptions, the conditional distribution over the class variable  $C$  is:

$$p(C_k|x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^n p(x_i|C_k)$$

Where the evidence  $Z = p(\mathbf{x})$  is a scaling factor dependent only on  $x_1, \dots, x_n$ , that is, a constant if the values of the feature variables are known.

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision rule.<sup>[15]</sup> The corresponding classifier, a Bayes classifier, is the function that assigns a class label  $\hat{y} = C_k$  for some  $k$  as follows:

$$\hat{y} = \operatorname{argmax}_{k \in \{1, \dots, K\}} p(C_k) \prod_{i=1}^n p(x_i|C_k).$$

## 2. Multinomial Naïve Bayes Classifier

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial  $(p_1, \dots, p_n)$  where  $p_i$  is the probability that event  $i$  occurs (or  $K$  such multinomials in the multiclass case). A feature vector  $\mathbf{x} = (x_1, \dots, x_n)$  is then a histogram, with  $x_i$  counting the number of times event  $i$  was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document (see bag of words assumption). The likelihood of observing a histogram  $\mathbf{x}$  is given by

$$p(\mathbf{x}|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i}$$

The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space:



$$\begin{aligned}
\log p(C_k|\mathbf{x}) &\propto \log \left( p(C_k) \prod_{i=1}^n p_{ki}^{x_i} \right) \\
&= \log p(C_k) + \sum_{i=1}^n x_i \cdot \log p_{ki} \\
&= b + \mathbf{w}_k^\top \mathbf{x}
\end{aligned}$$

Where,

$$b = \log p(C_k) \text{ and } w_{ki} = \log p_{ki}.$$

If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero. This way of regularizing naive Bayes is called Laplace smoothing when the pseudocount is one, and Lidstone smoothing in the general case.<sup>[16]</sup>

Rennie et al. discuss problems with the multinomial assumption in the context of document classification and possible ways to alleviate those problems, including the use of tf-idf weights instead of raw term frequencies and document length normalization, to produce a naive Bayes classifier that is competitive with support vector machines.<sup>[17]</sup>

### 3. Gaussian Naïve Bayes Classifier

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. For example, suppose the training data contain a continuous attribute,  $\mathbf{x}$ . We first segment the data by the class, and then compute the mean and variance of  $\mathbf{x}$  in each class. Let  $\mu_c$  be the mean of the values in  $\mathbf{x}$  associated with class  $c$ , and let  $\sigma_c^2$  be the variance of the values in  $\mathbf{x}$  associated with class  $c$ . Then, the probability distribution of some value given a class,  $p(\mathbf{x} = v|c)$ , can be computed by plugging  $v$  into the equation for a Normal distribution parameterized by  $\mu_c$  and  $\sigma_c^2$ . That is,

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

Another common technique for handling continuous values is to use binning to discretize the feature values, to obtain a new set of Bernoulli-distributed features; some literature in fact suggests that this is necessary to apply naive Bayes, but it is not, and the discretization may throw away discriminative information.<sup>[18]</sup>

#### 4. Bernoulli Naïve Bayes Classifier

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence features are used rather than term frequencies. If  $x_i$  is a boolean expressing the occurrence or absence of the  $i$ 'th term from the vocabulary, then the likelihood of a document given a class  $C_k$  is given by

$$p(\mathbf{x}|C_k) = \prod_{i=1}^n p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

where  $p_{ki}$  is the probability of class  $C_k$  generating the term  $w_i$ . This event model is especially popular for classifying short texts. It has the benefit of explicitly modelling the absence of terms. Note that a naive Bayes classifier with a Bernoulli event model is not the same as a multinomial NB classifier with frequency counts truncated to one.<sup>[18]</sup>

#### 5. Support Vector Machines

The Support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.<sup>[19]</sup> An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

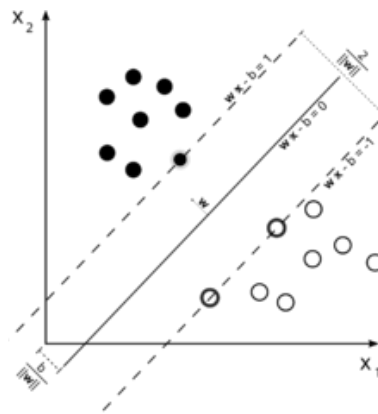
In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data is not labeled, a supervised learning is not possible, and an unsupervised learning is required, that would find natural clustering of the data to groups, and map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called support vector clustering is highly used in industrial applications either when data is not labeled or when only some data is labeled as a preprocessing for a classification pass; the clustering method was published.<sup>[20]</sup>

Given some training data  $\mathcal{D}$ , a set of  $n$  points of the form

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^P, y_i \in \{-1, 1\}\}_{i=1}^n$$

where the  $y_i$  is either 1 or  $-1$ , indicating the class to which the point  $\mathbf{x}_i$  belongs. Each  $\mathbf{x}_i$  is a  $P$ -dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having  $y_i = 1$  from those having  $y_i = -1$ . Any hyperplane can be written as the set of points  $\mathbf{x}$  satisfying



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

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

where  $\cdot$  denotes the dot product and  $\mathbf{w}$  the (not necessarily normalized) normal vector to the hyperplane. The parameter  $\frac{b}{\|\mathbf{w}\|}$  determines the offset of the hyperplane from the origin along the normal vector  $\mathbf{w}$ . If the training data are linearly separable, we can select two hyperplanes

in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called "the margin". These hyperplanes can be described by the equations

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

and

$$\mathbf{w} \cdot \mathbf{x} - b = -1.$$

Geometrically, the distance between these two hyperplanes is  $\frac{2}{\|\mathbf{w}\|}$ , so to maximize the distance between the planes we want to minimize  $\|\mathbf{w}\|$ . As we also have to prevent data points from falling into the margin, we add the following constraint: for each  $i$  either

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \quad \text{for } \mathbf{x}_i \text{ of the first class}$$

or

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \quad \text{for } \mathbf{x}_i \text{ of the second.}$$

This can be rewritten as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n. \quad (1)$$

We can put this together to get the optimization problem:

Minimize (in  $\mathbf{w}, b$ )

$$\|\mathbf{w}\|$$

subject to (for any  $i = 1, \dots, n$ )

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1.$$

## ➤ About the Implementation

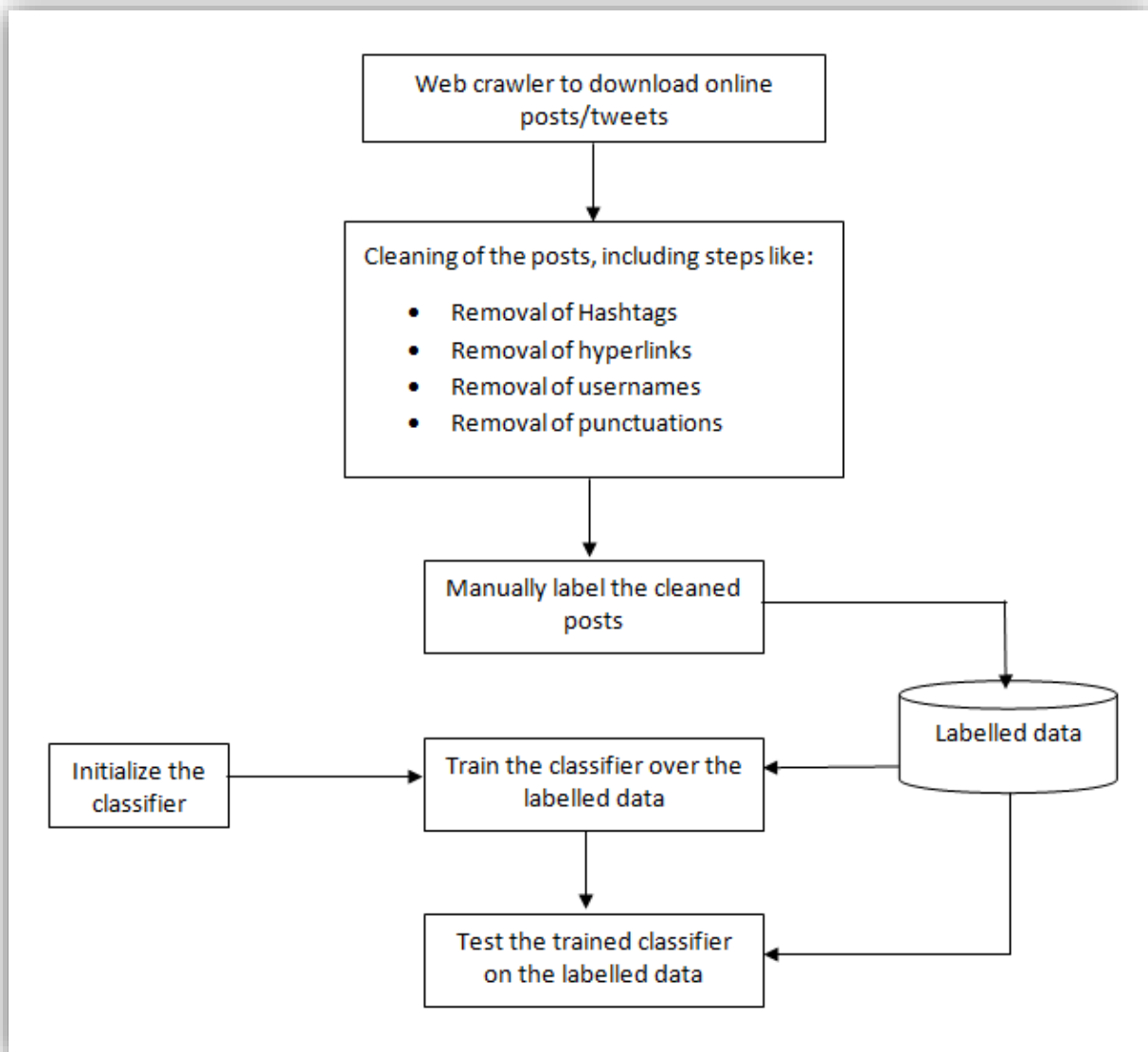


FIGURE: WORKFLOW

Twitter is a social networking and microblogging service that lets its users post real time messages, called tweets. Tweets have many unique characteristics, which implicates new challenges and shape up the means of carrying sentiment analysis on it as compared to other domains.

To get large publically available Twitter datasets, we use Twitter API.

In order to make authorized calls to Twitter's APIs, your application must first obtain an OAuth access token on behalf of a Twitter user or you could issue Application-only

authenticated requests when user context is not required. The way you will obtain such tokens will depend on your use case.<sup>[8,9]</sup>

Facebook is an online social networking service.

After registering to use the site, users can create a user profile, add other users as "friends", exchange messages, post status updates and photos, share videos and receive notifications when others update their profiles. Additionally, users may join common-interest user groups, organized by workplace, school or college, or other characteristics, and categorize their friends into lists such as "People From Work" or "Close Friends". Additionally, users may join the groups or subscribe to the pages corresponding to the services like banking , telephone etc. Users may post complaints, suggestions or seek help on these social media communities. Facebook had over 1.18 billion monthly active users as of August 2015. Because of the large volume of data users submit to the service, it is a very golden resource for text classification purpose.

### 1) **Crawling of Facebook Posts**

- First, in order to download facebook posts, we need to become Facebook application developer. For that we need to sign up our Facebook account as an App Developer.
- Once you have become a Facebook developer, we then create a new application and successful creation of the application , gives an APP ID and APP SECRET Key , which are then used by the crawler to access the facebook data.
- We establish our graph URL that we manipulate and open to receive JSON object responses containing data about Likes, talking about etc. To collect JSON objects of posts we will have to change our URL that we were manipulating so that it is secure, and able to pull the Post data. To make our graph url secure , we append our APP\_ID and APP\_Secret obtained earlier.
- To open the required url , we have used urllib package of python.
- After requesting that url , we get the corresponding json dump consisting of the posts on that page.
- Facebook uses the concept of paging , thus maximum 25 posts are in the one page.
- Code then extracts the url of the next page of posts from that page and keeps on doing that , till the all the posts have not been downloaded .

## 2) Cleaning of Downloaded Data

Raw data is the original data and is often hard to use for analysis directly. Data processed after applying the data cleaning step is ready for analysis. For this reason data cleaning process plays a very crucial in our Project. The data we extracted from twitter and facebook contains many undesirable symbols, words, and punctuations etc. that may slow down our applied algorithms or even make them to perform incorrectly. Hence, it becomes very necessary to write data cleaning modules or in some cases use the language library for the same purpose.

We present some of the differences between the rawdata and the tidy data obtained after the cleaning module is applied on it:

- Raw data may only be need to process only once (during cleaning).
- Raw data is the original data and is often hard to use for analysis directly.
- Processed Data is ready for analysis.
- Processing of raw data includes removal of hashtags, links, usernames, punctuations, non-English words .

We pre-process all the postss as follows:

- i. Remove all URLs (e.g. `www.example.com`), hash tags (e.g. `#topic`), targets (`@username`)
- ii. Correct spellings: A sequence of repeated characters is tagged by a weight. We do this to differentiate between the regular usage and emphasized usage of a word.
- iii. Using a POS tagger, the NL Processor linguistic Parser, we tag the adjectives, verbs and adverbs.

The following strategy has been followed to make the downloaded data tidy




- i. For Removal of hashtags, a substring removal python function is used.
- ii. Similar to the removal of hashtags, substring removal function is used to remove the links in the tweets.
- iii. Each punctuation is replaced by an empty substring
- iv. Removal of strings generally present in posts such as `@someUser` etc. through the substring removal function.

- v. The substring removal function removes the part of the string that contains a substring e.g. if substring = 'http', then http://www.google.com is removed, that means, remove until a space is found.
- vi. NLTK is a leading platform for building Python programs to work with human language data.
- vii. It provides easy-to-use suite of text processing libraries for tokenization and tagging.
- viii. NLTK has been used to remove the stopping words from all the tweets.
- ix. Words are fed to the Part of Speech tagger after Sentence Tokenizer and Word Tokenizer is applied.

### 3) Scoring Module

- 1) *If an adverb or verb is encountered after another adverb or verb then we add it. We repeat this process till another adjective isn't encountered.*
- 2) *Now if the added value of adverbs or verbs is less than 0 i.e., negative, then for the upcoming adjective, we subtract its value from 5 instead of multiplying.*
- 3) *And if added value of verbs or adverbs is positive and  $\geq 0.5$  then multiply it with the upcoming adjective else multiply 0.5 with upcoming adjective.*
- 4) *Later these multiplication results are added and the sum is divided by 5 times the number of adjectives encountered.*

### 4) Labelling of Posts

-  Since we couldn't obtain labelled data , so we need to label the data manually for our purpose.
-  Labelling is assigning classes to all the posts , these labelled posts are being used for training and testing .
-  For labeling , we prepare a script that shows every posts and asks user about the label of each post.



## 5) **Training of classifiers**

- ✚ We use a module to generate training and test data in a structured format that can be directly used by our training and test modules.
- ✚ We use 70% of our data as training data and rest of the dataset is being used for testing.
- ✚ We train several classifiers using our training data and then find out the accuracy of the respective classifiers by feeding the test data to these classifiers. `classify_module` incorporates all these functionality.

#### IV.) Screenshots and Some Analysis

##### ➤ Screenshots of GUI

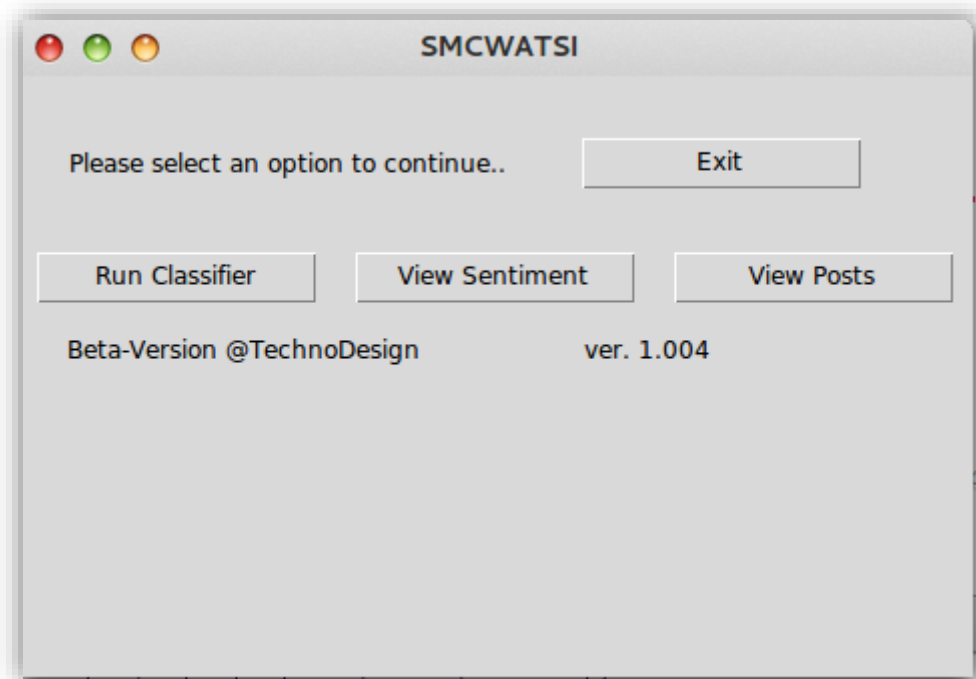


FIGURE: MAIN SCREEN

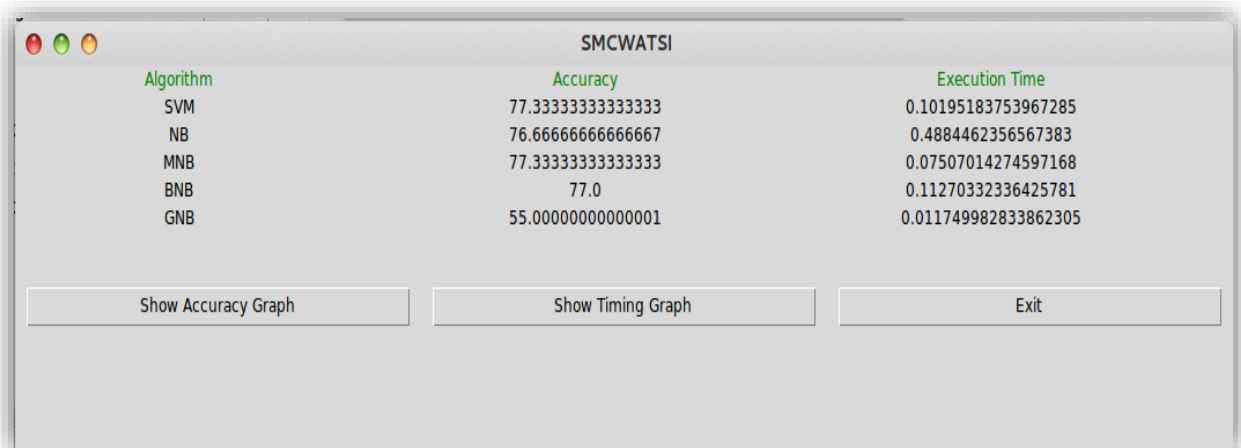


FIGURE: CLASSIFIER WINDOW

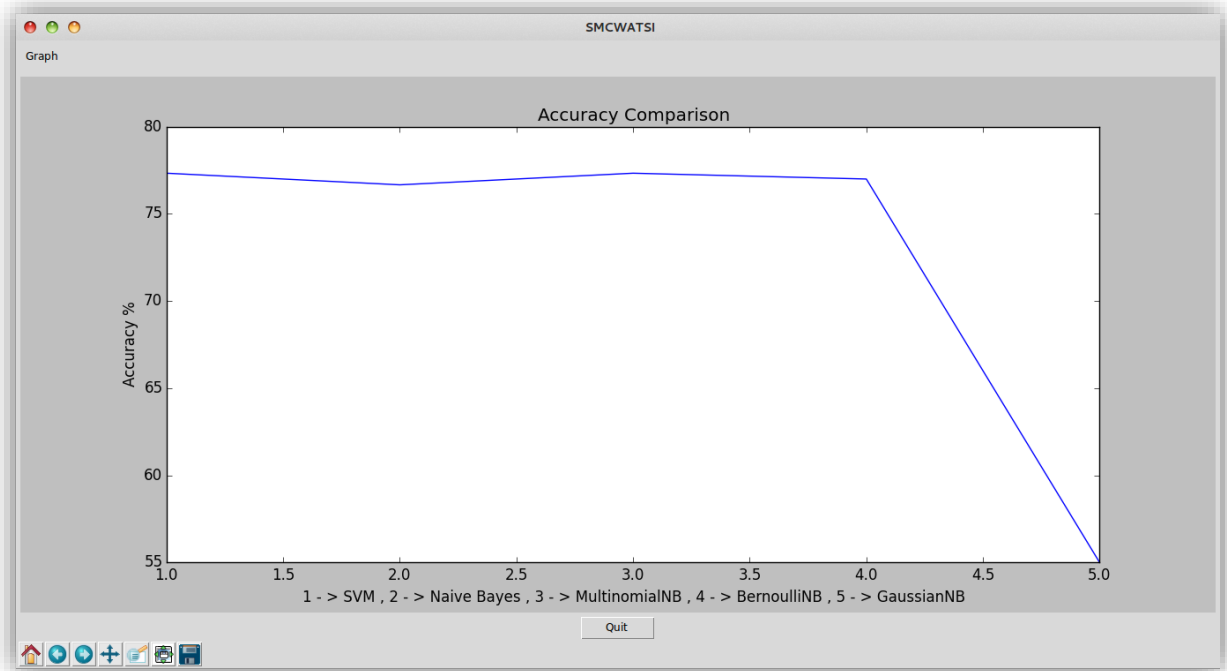


FIGURE: ACCURACY COMPARISON GRAPH WINDOW

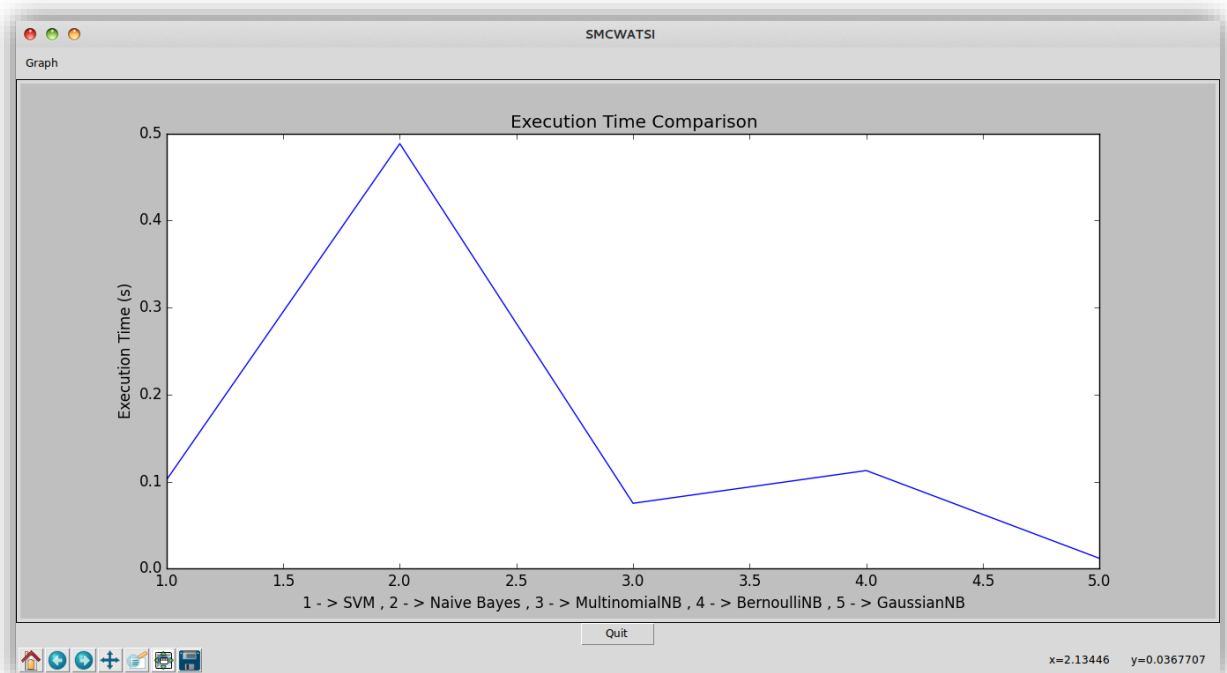
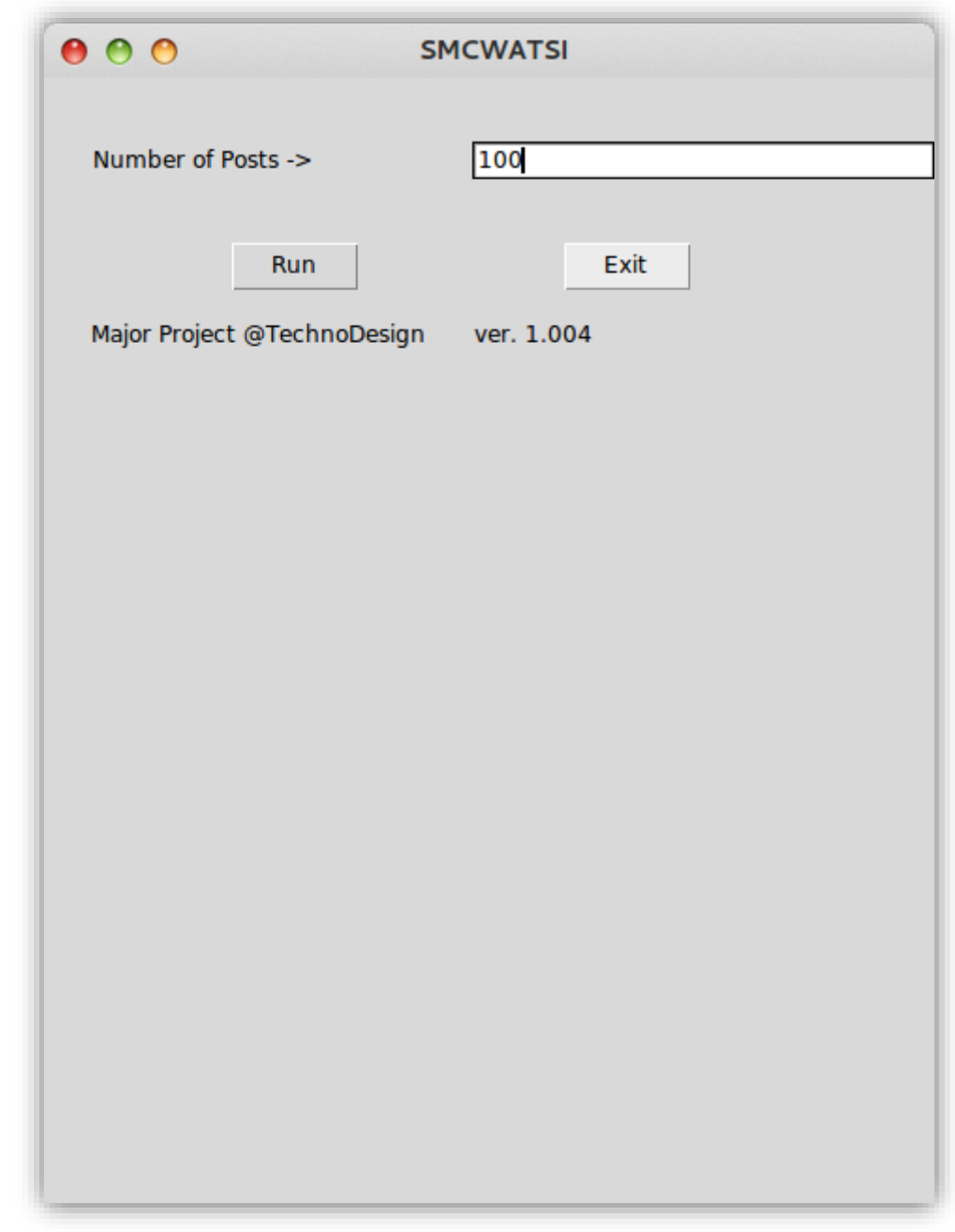


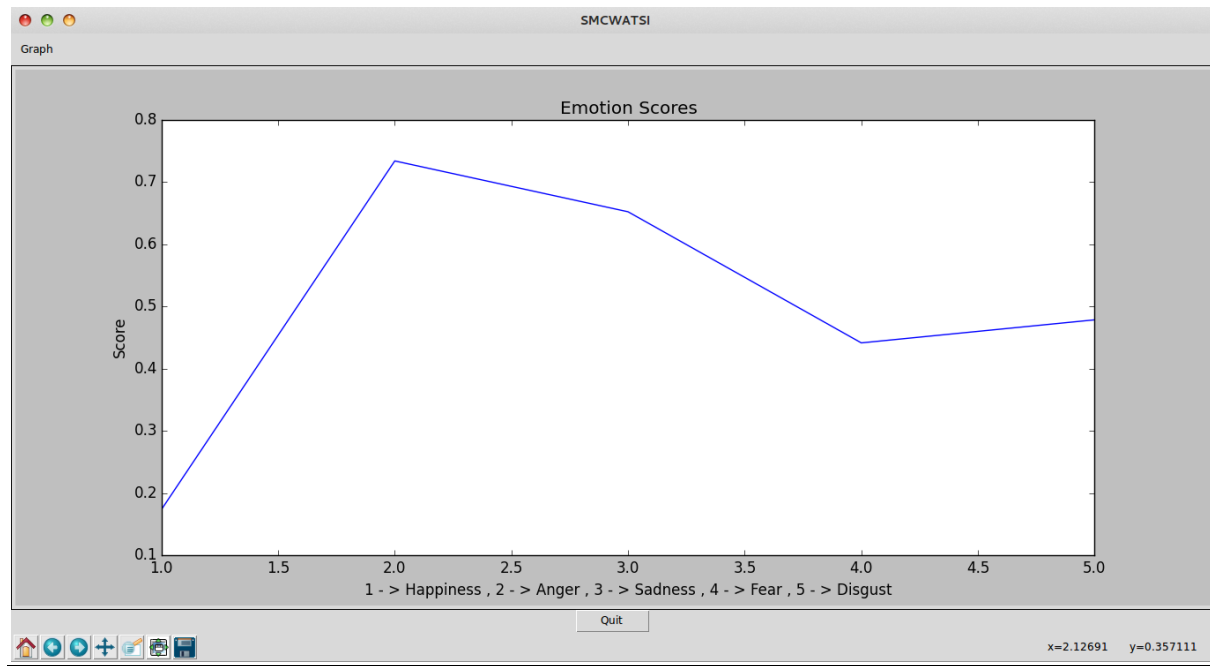
FIGURE: EXECUTION TIME COMPARISON GRAPH WINDOW



**FIGURE: SENTIMENT ANALYSIS WINDOW**



FIGURE: SENTIMENT ANALYSIS WINDOW SHOWING RESULTS



**FIGURE: EMOTION SCORES GRAPH WINDOW**

The figures given before show the graphical user interface. A comparison of Accuracy and Execution Times of the five algorithms on a constricted data set is given on the next page.

## ➤ Time and Accuracy Comparison of Algorithms Applied

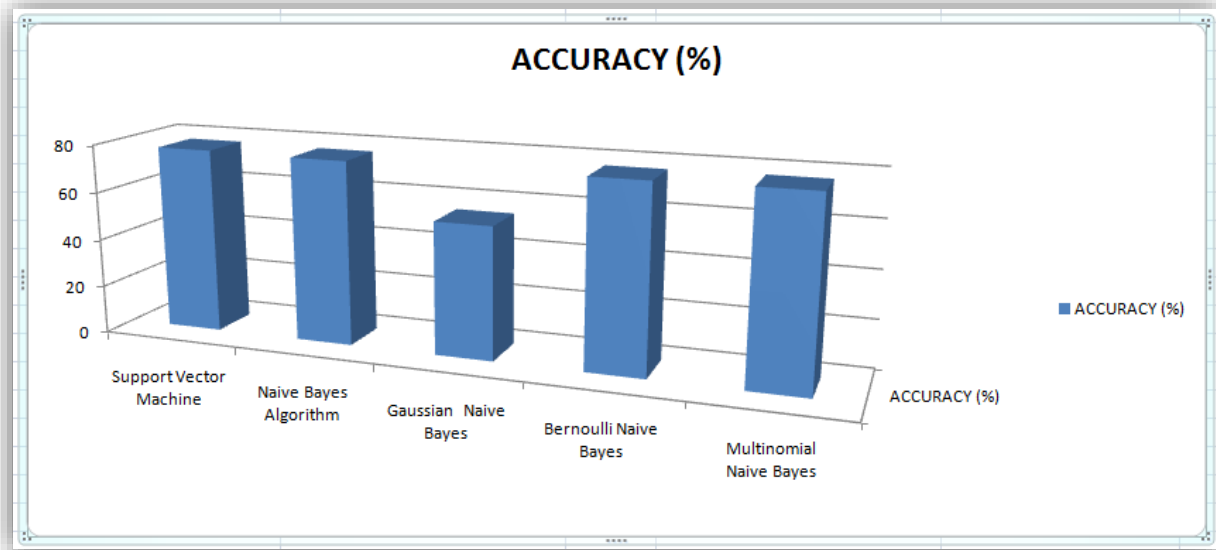


FIGURE: ACCURACY COMPARISON OF THE FIVE ALGORITHMS APPLIED FOR A CONSTRICTED DATA SET

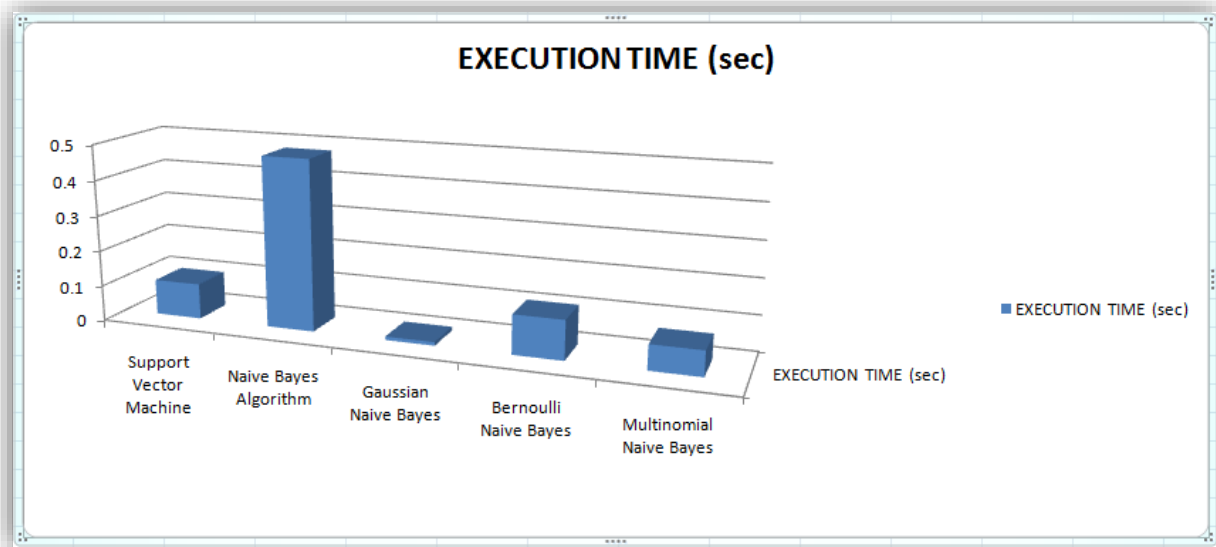


FIGURE: EXECUTION TIME COMPARISON OF THE FIVE ALGORITHMS APPLIED FOR A CONSTRICTED DATA SET

## V.) Applications and Future Work

Social Media tools such as Facebook, Twitter and LinkedIn do not have automated means to capture, control and appropriately manage and workflow process complaints. SMCWATSI (Social Media Complaint workflow automation tool using sentiment intelligence) provides an end to end complaint management tool that automatically captures incoming complaints at their social media source and immediately translates the message intent and directly the complaints to workflow and call centre queues where they can be instantly assessed, prioritized and resolved.

- Our Work Flow Model can be used by several commercial banks to automate their complaint/positive feedback collection process.
- This architecture can be slightly modified in order for it to work for any type of agency that might be wishing to automate their complaint/positive feedback collection process.
- Companies need to provide customers real time social media, email and SMS responses. SMC4 automatically captures and interprets incoming communications so that real-time actions and decisions can be taken. SMC4's sophisticated content analytics controls and pre-processes social media, email and SMS conversations, automatically moving conversations to the right person or department for action.

Following is the list of functionalities that can be added to our present workflow model:

- Department Classification: We have envisioned to create a user interface that will ask the user to enter a date to extract posts from that date to the present date. It will then analyse the posts and provide 2 options. First option will be able to check the majority emotion of the downloaded set of posts. This can be done in order to see the opinion of the majority of people posting on Facebook pages. And the second option will be able to select the posts and then classify them according to the department they should be sent.
- Sarcasm Detection: This module is envisioned to improve the current classification process. to detect sarcasm properly a computer would have to figure out that you meant the opposite of what you just said. It is sometimes hard for Humans to detect sarcasm, and Humans have a much better grasp at the English language than computers do, so this was not going to be an easy task.
- Apriori Association: This module is also envisioned to improve the current classification process. We haven't used any association rule and this method will provide a method that shows how much the different associations affect the probabilities of occurrence of words. It will provide stronger lexical affinity between the words and hence better contextual understanding by the machine.



## VI.) References

- 1) ComScore/the Kelsey group, “Online consumer-generated reviews have significant impact on offline purchase behavior,” Press Release, <http://www.comscore.com/press/release.asp?press=1928>, November 2007 [March 2015]
- 2) M. Bansal, C. Cardie, and L. Lee, “The power of negative thinking: Exploiting label disagreement in the min-cut classification framework,” in *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2008. (Poster paper) [March 2015]
- 3) “How Banks Can Use Social Media Analytics To Drive”, Internet: [www.cognizant.com/InsightsWhitepapers/How-Banks-Can-Use-Social-Media-Analytics-To-Drive-Business-Advantage.pdf](http://www.cognizant.com/InsightsWhitepapers/How-Banks-Can-Use-Social-Media-Analytics-To-Drive-Business-Advantage.pdf)
- 4) R. Picard, *Affective Computing*. MIT Press, 1997 [March 2015]
- 5) B. Liu, “Web data mining: Exploring hyperlinks, contents, and usage data,” *Opinion Mining*. Springer, 2006 [March 2015]
- 6) T. Nasukawa and J. Yi, “Sentiment analysis: Capturing favorability using natural language processing,” in *Proceedings of the Conference on Knowledge Capture (K-CAP)*, 2003 [March 2015]
- 7) J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack, “Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques,” in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2003. [March 2015]
- 8) Twitter. “REST API”. Internet: <https://dev.twitter.com/rest/public>, February 2015 [March 2015]
- 9) Twitter. “Streaming API”. Internet: <https://dev.twitter.com/streaming/overview>, February 2015 [March 2015]
- 10) ReadTheDocs. “Tweepy”. Internet: <https://media.readthedocs.org/pdf/tweepy/latest/tweepy.pdf>, February 2015 [March 2015]
- 11) Russell, Stuart; Norvig, Peter (2003) [1995]. *Artificial Intelligence: A Modern Approach* (2nd ed.). Prentice Hall. ISBN 978-0137903955.
- 12) Rennie, J.; Shih, L.; Teevan, J.; Karger, D. (2003). Tackling the poor assumptions of Naive Bayes classifiers (PDF). ICML.

- 13) Rish, Irina (2001). An empirical study of the naive Bayes classifier (PDF). IJCAI Workshop on Empirical Methods in AI.
- 14) Hand, D. J.; Yu, K. (2001). "Idiot's Bayes — not so stupid after all?". International Statistical Review 69 (3): 385–399. doi:10.2307/1403452. ISSN 0306-7734.
- 15) Zhang, Harry. The Optimality of Naive Bayes (PDF). FLAIRS2004 conference.
- 16) Caruana, R.; Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. Proc. 23rd International Conference on Machine Learning. CiteSeerX: 10.1.1.122.5901.
- 17) NarasimhaMurty, M.; Susheela Devi, V. (2011). Pattern Recognition: An Algorithmic Approach. ISBN 0857294946.
- 18) John, George H.; Langley, Pat (1995). Estimating Continuous Distributions in Bayesian Classifiers. Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence. Morgan Kaufmann. pp. 338–345.
- 19) McCallum, Andrew; Nigam, Kamal (1998). A comparison of event models for Naive Bayes text classification (PDF). AAAI-98 workshop on learning for text categorization Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning 20(3): 273. doi:10.1007/BF00994018.
- 20) Ben-Hur, Asa, Horn, David, Siegelmann, Hava, and Vapnik, Vladimir; "Support vector clustering" (2001) Journal of Machine Learning Research, 2: 125-137.