

**A Project Report**  
**on**  
**Sentiment analysis for social**  
**networks using machine learning**

*Submitted in partial fulfillment of the*  
*requirement for the award of the degree of*  
**Bachelor of Technology in Computer Science and**  
**Engineering**



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

**Under The Supervision of**

**Ms. Swati Sharma**

**Assistant Professor**

**Department of Computer Science and Engineering**

**Submitted By**

**SHIVANSH SRIVASTAVA – 18SCSE1010734**

**YASHSHRI SHARMA – 18SCSE1010149**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA DECEMBER -**  
**2021**



**SCHOOL OF COMPUTING SCIENCE AND  
ENGINEERING  
GALGOTIAS UNIVERSITY, GREATER NOIDA**

**CANDIDATE'S DECLARATION**

I/We hereby certify that the work which is being presented in the project, entitled “**Sentiment analysis for social networks using machine learning**” in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Ms. Swati Sharma, Assistant Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1010734 – SHIVANSH SRIVASTAVA

18SCSE1010149 – YASHSHRI SHARMA

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor  
(Ms. Swati Sharma, Assistant Professor)

**CERTIFICATE**

The Final Thesis/Project/ Dissertation Viva-Voce examination of **18SCSE1010734 – SHIVANSH SRIVASTAVA, 18SCSE1010149 – YASHSHRI SHARMA** has been held on \_\_\_\_\_ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

**Signature of Examiner(s)**

**Signature of Supervisor(s)**

**Signature of Project Coordinator**

**Signature of Dean**

Date:

Place:

## **ABSTRACT**

Social media like Facebook and Twitter is full of opinions, emotions and feelings of people all over the world. Through this project, we are detecting, classifying and quantifying emotions of any form. We are considering text collected from social media platforms for opinion mining. Analyzing and classifying text on the basis of emotions comes under Sentiment Analysis. This project is aiming at classifying text into six different Emotion-Categories, which are Happiness, Sadness, Fear, Anger, Surprise and Disgust. To extract these emotions from text we are using two different approaches and combining them. The first approach is based on NLP (Natural Language Processing) and using several textual features such as parts of speech, negations, degree words and other grammatical analysis. The second approach is based on ML (Machine Learning) classification algorithms. Through this project to remove the need of manual annotation of large datasets we are also, devising a method to automate the creation of training set itself. In addition to that, we are creating a large collection of emotional words along with their intensities.

## Table of Contents

No.	Title	Page
	<b>Candidates Declaration</b>	
	<b>Acknowledgement</b>	
	<b>Abstract</b>	
	<b>List of Table</b>	
	<b>List of Figures</b>	
	<b>Acronyms</b>	
	<b>Chapter 1 Introduction</b>	<b>9</b>
	<b>1.1 INTRODUCTION</b>	<b>9</b>
	<b>1.2 FORMULATION OF PROBLEM</b>	<b>11</b>
	<b>1.2.1 TOOLS AND TECHNOLOGY USED</b>	<b>27</b>
	<b>Chapter 2 Literature Survey/Project Design</b>	<b>35</b>
	<b>Chapter 3 Functionality/Working of Project</b>	<b>37</b>
	<b>3.1 Project Design Diagrams</b>	<b>57</b>
	<b>Chapter 4 Module Description</b>	<b>60</b>
	<b>Chapter 4 Results and Discussion</b>	<b>61</b>
	<b>Chapter 5 Conclusion and Future Scope</b>	
	<b>5.1 Conclusion</b>	<b>65</b>
	<b>5.2 Future Scope</b>	<b>65</b>
	<b>Reference</b>	<b>66</b>

## List of Figures

<b>S.No.</b>	<b>Caption</b>	<b>Page No.</b>
<b>1</b>	<b>Arrangement of Dart Files and Packages</b>	<b>8</b>
<b>2</b>	<b>Architectural Layers of Flutter</b>	<b>9</b>
<b>3</b>	<b>Class Diagram</b>	<b>11</b>
<b>4</b>	<b>Sequence Diagram</b>	<b>12</b>

## List of Tables

<b>S.No.</b>	<b>Title</b>	<b>Page No.</b>
<b>1</b>	<b>Data Table</b>	<b>6</b>
<b>2</b>	<b>Information Data</b>	<b>11</b>
<b>3</b>	<b>Accuracy of SMO classifier</b>	<b>62</b>
<b>4</b>	<b>Accuracy of J48 classifier</b>	<b>62</b>

## Acronyms

SVM	Support Vector Maching
ML	Machine Learning
DL	Deep Learning
CNN	Convolution Neural Networks



# CHAPTER-1

## 1.1 Introduction

The manner in which humans express their perspective has been modified by the age of internet. Online dialogue, weblog post, product review websites and many more are used to clarify the human perspective. Nowadays people dependency on such platforms has increased in tiffini quantity.

Now a typical buyer first tries to search for product and services online before making any final decision. The amount of data generated by user is increasing drastically and it's very difficult to research on those huge data. Machine learning and symbolic strategies are two important strategies used for sentiment analysis. Expertise based approach requires a large database of pre-defined feelings and a green-expertise illustration for figuring out the sentiments. Machine Learning (ML) approach uses an educational set to expand sentiment classifier that classifies sentiments. Given that for system learning method, pre-defined database of entire emotions is not needed, it's much easier than knowledge- based technique.

The forefront of this project covers the assessment of substance at web covering bunches of zones which are wrapping in exponential numbers despite in volumes, for example Amazon. Twitter wherein the tweet conveys surveys, but it would be very time consuming for looking to get general comprehension of such un-structured records (audits). Customers see unstructured measurement on a particular site and afterwards expanding picture regarding the items and ultimately delivering beyond any uncertain judgement, at long last. After that the surveys are added together for securing criticisms for unprecedented purpose to offer many important audits in which we make use of assessment examination. A procedure under which data set includes mentalities, emotions or evaluation which ponder over the way a human think is called Slant Assessment. It's totally hard endeavor, hoping to get the negative and positive segment. The capacities used for classifying the sentences have a completely sturdy modifier with the supposition to shorten the assessment.

Those are composed into unmistakable strategies which aren't without any issue summed up with the guidance of organization or clients making it troublesome to group them. Clients are influenced by Opinion assessment to classes whether the information regarding the item is fine or no longer needed before they gather. For catching items or administrations, organizations and advertisers utilize this investigation in a way that with regards to individual's needs it might be furnished. Super vised and

managed are two styles of machine information picking techniques which may be used for notion investigation. Unsupervised technique which does not consist of class and they don't give any destinations and hence conduct bunching, are becoming more familiar. Managed learning is a kind of learning which is constructed completely with respect to dataset and therefore all through the strategy names are getting outfitted to the model.

All those sorted dataset when experienced through fundamental leadership are made to supply yields. This project is based on supervised system learning as it helps to provide superior opinion examination.

Emotions are described as intense feelings that are directed at something or someone in response to internal or external events having a particular significance for the individual. And the internet, today, has become a key medium through which people express their emotions, feelings and opinions. Every event, news or activity around the world, is shared, discussed, posted and commented on social media, by millions of people. Eg. "The Syria chemical attacks break my heart!! :'" or "Delicious dinner at Copper Chimney! :D" or "OMG! That is so scary!". Capturing these emotions in text, especially those posted or circulated on social media, can be a source of precious information, which can be used to study how different people react to different situations and events. Business analysts can use this information to track feelings and opinions of people with respect to their products. The problem with most of the Sentiment Analysis that is done today is that the analysis only informs whether the public reaction is positive or negative but fails to describe the exact feelings of the customers and the intensity of their reaction. With our emotional analysis, they can have a more profound analysis of their markets than the naive 2-way Sentiment Analysis, which itself has turned their businesses more profitable. Business leaders can analyse the holistic view of people in response to their actions or events and work accordingly. Also, health-analysts can study the mood swings of individuals or masses at different times of the day or in response to certain events. It can also be used to formulate the mental or emotional state of an individual, studying his/her activity over a period of time, and possibly detect depression risks. There are plenty of research works that have focussed on Sentiment Analysis and provide a 2-way classification of text. But few have actually focussed on mining emotions from text. However, machine analysis of text to classify and score it on the basis of emotions poses the following challenges :

- Instead of the usual two categories in Sentiment Analysis, there are six Emotion-Categories in which we need to classify the tweets.
- Lack of manually annotated data to train classifiers to label data into six categories.

• Unavailability of a comprehensive bag of Emotion-words labeled and scored according to Emotion-Categories (Happiness, Sadness, etc.) and their intensities, that can be used to detect Emotion-words in text. In order to address the aforementioned challenges, it was important to devise a system that could generate a good and reliable training-set for the classifier, a labeled bag of words, and an algorithm that could not only detect emotions, but also score and label the tweets according to those emotions. A lot of research has been done on classifying comments, opinions, movie/product reviews, ratings, recommendations and other forms of online expression into positive or negative sentiments. Emotions have also been studied, but in a limited extent, such as by asking specific questions and judging on the basis of replies, or an analysis done only on short one-lined headlines or a few others [1], [2], [3], all of which depended on the manual annotation of the training dataset of a small size and limited scope. In this paper, we propose a method to classify and quantify tweets according to six standard emotions suggested by Paul Ekman [4]. Here, we base our analysis on tweets posted on Twitter, but it can be easily extended to any kind of text whether it is one lined headlines, messages and posts on social media or larger chunks of writings, because of automatic development of our training set. Our main contributions are listed below:

- We have developed a system that could score and label any piece of text, especially tweets and posts on social media according to six Emotion-Categories: Happiness, Sadness, Fear, Surprise, Anger and Disgust along with their intensity scores, making use of its textual features, a variety of NLP tools and standard Machine Learning classifiers.
- Another significant contribution is that we have successfully devised a system that could automatically (without any manual effort) build an efficient training set for our ML Classifiers, consisting of a large enough set of labeled tweets from all Emotion-Categories.
- We have created a large bag of words in English, that consists of words expressing a particular emotion along with the intensity of that emotion.
- We were able to achieve an accuracy of about 91.7% and 85.4% using J48 and SMO classifiers respectively using the training set we built.

## 1.2 Formulation of problem

People like expressing sentiment. Happy or unhappy. Like or dislike. Praise or complain. Good or bad. That is, positive or negative.

*Sentiment analysis* in NLP is about deciphering such sentiment from text. Is it *positive, negative, both, or neither*? If there is sentiment, *which* objects in the text the sentiment is referring to and the actual sentiment phrase such as *poor, blurry, inexpensive, ...* (Not just *positive* or *negative*.) This is also called *aspect-based analysis* [1].

As a technique, sentiment analysis is both interesting and useful.

First, to the interesting part. It's not always easy to tell, at least not for a computer algorithm, whether a text's sentiment is positive, negative, both, or neither. The cues can be subtle. Overall sentiment aside, it's even harder to tell which objects in the text are the subject of which sentiment, especially when both positive and negative sentiments are involved.

Next, to the useful part. This is easy to explain. People who *sell* things want to know about how people *feel* about these things. It is called *customer feedback*. Ignoring it is bad for business.

There are other uses as well. Such as *opinion mining*, i.e. trying to figure out who holds (or held) what opinions. Such as, *according to John Smith, the coronavirus will simply go away within six months*. This task may be formalized as seeking *(source, target, opinion)* triples. In our example, *source = John Smith, target = coronavirus, opinion = will simply go away within six months*.

### (Many) Examples

Sentiment analysis is what you might call a long-tail problem. Lots of varying scenarios and subtleties. Such problems are often best described by examples.

First, let's see some easy positives.

Amazing customer service.

Love it.

Good price.

Next, some positives and negatives a bit harder to discriminate.

Positives:

What is not to like about this product.

Not bad.

Not an issue.

Not buggy.

Negatives:

Not happy.

Not user-friendly.

Not good.

Definitely not positive:

Is it any good?

The positives in the above list are not the strongest ones. That said, they are especially good for training ML algorithms to make key distinctions, as we definitely don't want these positives to be predicted as negatives.

**Positives:**

Low price.

**Negatives:**

Low quality.

These instances are especially good for training ML algorithms to make key distinctions.

**Positives:**

Quick turn-around.

**Negatives:**

Quick to fail.

The same point applies here.

**Positives:**

Inexpensive.

Negatives:

The same point applies here.

Finally, some negatives which are a bit harder to decipher.

**Positives:**

**What is not to like about this product.**

**Not bad.**

**Not an issue.**

**Not buggy.**

**Negatives:**

**Not happy.**

**Not user-friendly.**

**Not good.**

**Definitely not positive:**

**Is it any good?**

**1.3 Use Cases**

It's easy to imagine many. Here are some of the main specific ones.

1. Discover negative reviews of your product or service. On blog posts or eCommerce sites or social media. More broadly anywhere on the web.
2. Aggregate sentiment on financial instruments. Such as specific stocks. What is the recent market sentiment on stock xyz? Also, aspect-based variants. Such as *according to analysts at financial company xyz, stock abc is likely to grow 20% in the coming year*. Discerning who's opinion it is provides more information, which may be used to assess credibility or lack thereof.
3. Identify which components of your product or service are people complaining about? Especially strongly. For prioritizing tactical or long-term improvements.
4. Track changes to customer sentiment over time for a specific product or service (or a line of these). To check if things have been getting better ...
5. Track shifting opinions of politicians over time. Individuals or groups such as political parties. News media love to do this. To fuel nagging questions such as *you said that then but now this!*.

#### **1.4 Computational Problems**

What we've discussed thus far may be crystallized into two distinct computational problems.

1. What is the text's overall sentiment: *positive, negative, both, or neither?*
2. Which sentiment applies to which portions of the text. This is also called aspect-based sentiment analysis.

Let's start with the first problem, which we will call *sentiment classification*.



## 1.5 Sentiment Classification Problem

The input is text. The output we seek is whether the sentiment is *positive*, *negative*, *both* or *neither*. In a variant of this problem, which we will not address here, we are interested in additionally predicting the strengths of the positive and negative sentiments. You can imagine why. *xyz phone really sucks* is way more negative than *I'm a little disappointed with xyz phone*.

## 1.6 Dictionary-based Approach

The simplest approach is to create two dictionaries, of terms carrying positive and negative sentiment respectively. By term, we mean a word or a phrase. A text is classified as positive or negative based on hits of the terms in the text to these two dictionaries. A text is classified as neutral if it hits neither dictionary. A text is classified as both positive and negative if it hits in both dictionaries.

This approach is worth considering when one wishes to quickly get a somewhat effective sentiment classifier off-the-ground and one doesn't have a rich-enough data set of text labeled with the sentiment. Simplicity is one reason. The more important reason is that the machine learning alternative has its own obstacles to be overcome. We'll delve into these in detail when we discuss that topic.

Machine-learning obstacles notwithstanding, a dictionary-based approach will run into quality issues sooner or later. So if high precision and high recall of the various sentiment classes are important in your use case, you should consider biting the bullet upfront and investing in ML. Your task will become much easier if you can find a rich-enough labeled data set or come up with some creative ways to get one, possibly after some additional lightweight NLP (discussed in an upcoming section).

## Supervised Learning Challenges

The first challenge is the necessity of having a large and diverse data set of texts labeled with their sentiment classes: *positive*, *negative*, *both*, or *neither*.

The issue is this. Think of the text as being represented by a vector. For now in the usual vector space model, i.e. as a bag of words. That said, the challenge applies, albeit to a somewhat lesser extent, even to word embeddings.

The vector space is huge. Each word in the lexicon has a dimension.

The vast majority of the words in this space carry no sentiment. To train a machine learning classifier would require a huge training set. Much of what it would be doing is learning which words are “nuisance” words. That is, unlearning biases it collected along the way (see example below).

Let’s see an example from which the classifier can learn to wrongly associate neutral words with positive or negative sentiment.

xyz phone sucks → negative

It will learn to associate the word *phone* with the sentiment *negative*. Obviously we don’t want this. Unlearning this will require training set instances with the word *phone* in them that are labeled *neither* (i.e., *neutral*).

That being said, breaking up a large and diverse corpus (such as Wikipedia) into sentences and labeling each *neutral* might alleviate this problem. The intuition here is this. All words will initially learn to be neutral. Words such as *sucks* that repeatedly occur in text labeled *negative* will eventually ‘escape’ from their neutral label.

## **Beyond Bag-of-words Features**

From the labeled examples we saw in an earlier section, it seems that a ‘?’ is a predictor of sentiment. This makes sense intuitively. Skeptics ask questions. Not true believers.

## Leveraging Dictionaries as Features

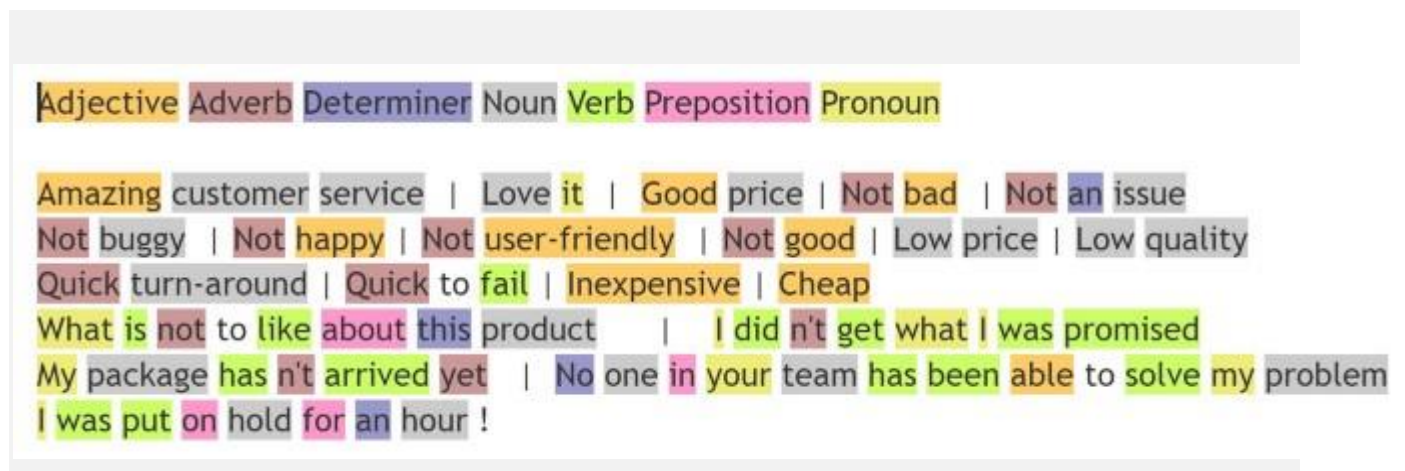
If we already have dictionaries of phrases correlated with positive or negative sentiment (or find them easy to construct), why not use them as additional features. They don't have to be complete. Just curated. So we can take advantage of their quality.

In more detail, here's how. Say *not good* is in the dictionary of negatives. We would create a boolean feature for this entry. This feature's value is 1 if *not good* appears in text and 0 if not. We might also add the entry (*not good, negative*) to our training set. Note that here we are thinking of *not good* as the full text.

## Use Part-of-speech?

Sentiment-rich words are often adjectives. This makes one wonder whether using information about the part-of-speech of each word in the text might be useful? As additional features or for pruning features.

Let's start by looking at the parts-of-speech of the words in our various examples. This analysis was done using the online pos-tagger at [2].



The screenshot shows a POS-tagger interface with a legend at the top and several example sentences below. The legend includes: Adjective (orange), Adverb (grey), Determiner (purple), Noun (light blue), Verb (green), Preposition (pink), and Pronoun (yellow). The example sentences are: 'Amazing customer service | Love it | Good price | Not bad | Not an issue', 'Not buggy | Not happy | Not user-friendly | Not good | Low price | Low quality', 'Quick turn-around | Quick to fail | Inexpensive | Cheap', 'What is not to like about this product | I did n't get what I was promised', 'My package has n't arrived yet | No one in your team has been able to solve my problem', and 'I was put on hold for an hour !'. Each word in these sentences is highlighted with a colored box corresponding to its part-of-speech.

Parts-of-speech of various words in sentiment-carry sentences

What thoughts does this trigger? The POS-tag *adjective* seems significantly correlated with sentiment

polarity (positive or negative). The POS-tag *adverb* also. *Determiners*, *prepositions*, and *pronouns* seem to predict the neutral class.

How might we take advantage of this? We could gate bag-of-words features on their parts-of-speech. For example, filter out all words whose POS-tag is determiner, preposition, or pronoun. This may be viewed as an elaborate form of stop-words removal.

### **Feature Engineering: Some Observations**

Whereas these observations are general, they especially apply to our problem (sentiment classification).

First, we don't need strong evidence before we add a new feature. Merely a weak belief that it might help. The machine learning algorithm will figure out how predictive this feature is, possibly in conjunction with other features. As the training set gets richer over time, the ML will automatically learn to use this feature more effectively if this is possible. Weak features can add up.

The only downside to this is that if we go overboard, i.e. add too many features, the feature space explosion may come back to haunt us. More on that later.

Let's expand on "weak belief that it might help". Here, 'help' just means that the feature is predictive of some sentiment class. We don't need to know which. The ML will figure this out. That is, which feature value predicts which sentiment class. By contrast, when setting up a rule-based system (of which dictionaries are a special case) one has to specify which combinations of feature values predict which sentiment class.

### **1.7 Does This Risk Feature Space Explosion?**

We have already accepted that using bag-of-words features will explode our feature space. For reasons discussed earlier, we have decided to bite the bullet on this front. The question is, will the additional

features mentioned in this section make the matter worse?

Actually they will make it better. Let's reason through this. First the *question-mark* feature. It is boolean-valued. No explosion here. Next, the dictionary-based features. These in fact reduce the noise in the space of word vectors as they surface sentiment-rich words and phrases. Finally, the part-of-speech features. Using them as suggested, for filtering (i.e. removing words), prunes the feature space.

### **Word k-gram Features?**

We deliberately put this after the previous section because this does run a greater risk of exploding the feature space if not done right. The space of word  $k$ -grams even with  $k = 2$  is huge. That said, pruning this space sensibly can potentially increase the benefit-to-cost ratio from these features.

Below are some plausible ideas to consider. In the discussion, we limit ourselves to  $k=2$ , i.e. to bigrams, although it applies more generally.

1. Prune away bigrams from the model that don't have sufficient support in the training set. (By the support of a bigram we mean the number of times it occurs in the training set.)
2. For additional pruning, consider parts-of-speech as well.

### **Taking Stock ...**

We'll close this section by taking stock of what we have discussed here and its implications. First, we see that the ML approach can be empowered with a variety of features. We simply throw features into the mix. So long as there is a plausible case for each inclusion. We don't worry about correlations among features. Too complicated to analyze. Let the ML sort it out. The end justifies the means. So long as we have a rich enough labeled data set which we can partition to train-and-test splits and reliably measure the quality of what we are referring to as 'end'.

We do need to think about the feature space explosion. We already did.

Now a few words about the learning algorithm. We have lots of choices. Naive Bayes. Logistic Regression. Decision Tree. Random Forest. Gradient Boosting. Maybe even Deep Learning. The key point to bring to the surface is that these choices span varying levels of sophistication. Some can automatically discover multivariate features that are especially predictive of sentiment. The risk here is that many of the multivariate features they discover are also noisy.

Okay so now we have lots of feature choices and lots of learning algorithm choices. Potentially very powerful. But also risky. As mentioned earlier, we can mitigate the risk by keeping in mind the feature-space explosion.

Ultimately though we should focus on building as rich of a labeled data set, even if only incrementally. Longer-term this has more value than tactically optimizing features to compensate for not having a great training set. This is the single most important aspect of this problem. Invest in this.

## 1.8 Target Variants

To this point, we've been thinking of sentiment classification as a 4-class problem: *positive*, *negative*, *both*, *neither*. In some settings, the class *both* can be ignored. In such settings, we interpret *neither* as *neutral*.

In most use cases, we only care about the first two. So *neutral* is a nuisance class. 'Nuisance' means it needs to be accounted for, even though it's not what we seek. Why does it need to be accounted for? Well, we don't want text that is neutral to get classified as positive or negative. Said another way, including the neutral class (backed by a sufficiently rich training set for it), improves the precision of the positives and negatives.

This is easy to illustrate with an example. Remember the instance

```
xyz phone sucks → negative
```

We wouldn't want the inference *phone* → *sucks*. Meaning that every phone sucks. By adding the neutral class, along with a suitably rich training set for it, the risk of this type of unwarranted inference

reduces greatly.

## 1.9 Probabilistic Classification

Regardless of which learning algorithm we end up choosing — Naive Bayes, Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, ... — we should consider leveraging the predicted probabilities of the various classes. For example, if the predicted probabilities on an input are roughly 50% (positive), 50% (negative), 0% (0) then we can interpret the text as having both positive and negative sentiments.

### How to build a training set efficiently

Okay, so it's clear that the ML approach is powerful. Let's now look to “feeding the beast”, i.e. building a rich training set.

Here's an idea of how to quickly assemble a large set of texts that can be manually labeled efficiently.

1. Pick a suitable source of unstructured text. Such as product reviews at an e-commerce site.
2. Create two columns in a spreadsheet, one for *text*, one for *label*.
3. Put each document (e.g. each product review) in its own cell in the column labeled *text*.
4. Manually add the labels.

Let's elaborate on step 4. Consider crowd-sourcing it. Or at least dividing up the work among team members. Plus adopt a convention that an empty cell in the label column denotes a specific label. A good choice is *neither*, i.e. *neutral*.

You might be surprised at how quickly you can build up a rich training set using this process.

If your product reviews data set comes with a star-rating attached to each review, you can use this rating to auto-label the positive and negative instances. This can speed up the labeling process. That said, you should make a manual pass after the auto-labeling to review it and correct those labels that are wrong.

The assumption underlying this auto-labeling is that its quality is reasonably good. So that only a small proportion of the labels need fixing. You do have to look at them all. Still, visually scanning all labels has a much higher throughput than editing individual ones.

### **1.10 Training Instance Granularity**

Generally speaking, to the extent possible, input instances should be more granular than coarser. Customer product reviews are generally granular enough. Especially if they are already tagged with the ratings, from which we might auto-derive the sentiment target. In this case, breaking longer reviews down to individual sentences and manually tagging them with an appropriate sentiment label might be too much work, whereas its benefit unclear.

Next, consider starting points being longer documents. Such as full-length review articles of product classes. For example, *The Best 10 Phones for 2020* or *The Best 10 Stocks for 2020*.

The case for breaking these down into finer granularity such as paragraphs or even sentences is stronger. Clearly, if we can restrict the text to the region to which a specific sentiment is applicable, it can help improve the learning algorithm's accuracy.

As an extreme example, say you have a 20-page document, all of it neutral, except one sentence which has a strong sentiment. It makes sense to label this sentence with the sentiment and the rest of the text as neutral. That'll likely work better than labeling the 20-page document with the sentiment in that one sentence.



## 1.11 Granularity Of Instances In The Field

As discussed above, for the training set, finer-grained instances in the training set are generally better than coarser-grained ones. This preference does not apply to classification time, i.e. the use of the classifier in the field. We should go ahead and predict the sentiment of whatever text we are given, be it a sentence or a chapter.

Unlike during training, there is no downside to predicting the sentiment of a long document. It's just a question of expectations. If a user seeks a sentiment of a document longer than a paragraph, what she really means is she wants the overall general sentiment across the text. Is it positive overall, negative overall, both, or neither (neutral)?

This is fine, sometimes that is what you want. And once you have discovered documents that carry some sentiment, you can always drill down to run the sentiment classifier on their individual sentences or paragraphs.

In view of this, we should keep in mind that evaluation on a test set held-out from the labeled data set will not yield an accurate assessment of how well the classifier works in the field. The held-out test set is derived from the labeled data set, which is composed of granular instances for reasons discussed earlier. The field's inputs are not necessarily always that granular.

## 1.12 Aspect-based Sentiment Analysis

Here, in addition to deciphering the various sentiments in the text we also seek to figure out which of them applies to what.

Clearly such analysis can be very useful, as illustrated by the example below.

xyz phone sucks → negative

You clearly want to know what is being complained about and what is being liked.

Often, we also care to extract the actual sentiment phrases. Consider the example below from a made-up holistic review of a new TV.

Good price. Sharp image. Vivid colors. Static in Audio. Motion lags a bit.

Ideally, we'd like to extract (aspect, sentiment-phrase, polarity) triples from it.

Good price. Sharp image. Vivid colors. Static in Audio. Motion lags a bit.

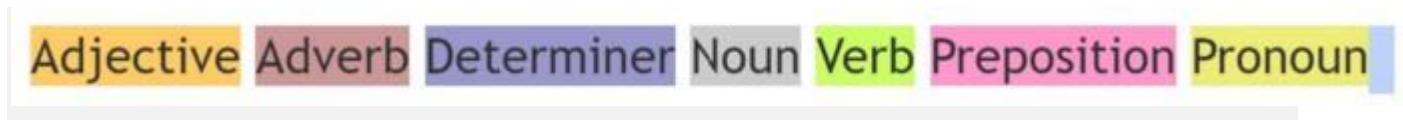
The polarities may help derive an overall quality score (e.g., here 3 out of 5). May have other uses as well.

### **Extracting Aspects And Sentiment Phrases**

Let's run this text through the POS-tagger at [2].

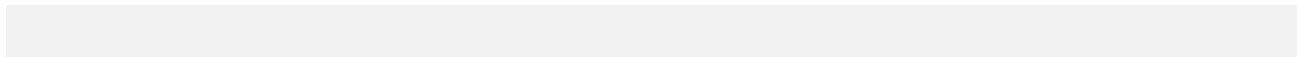
Good price. Sharp image. Vivid colors. Static in Audio. Motion lags a bit.

Recall that the POS-tag legend is



Legend of POS tags

What jumps out at you? As a first attempt, splitting the text into sentences, running a POS-tagger on each sentence, and if the tag sequence is



Adjective Noun

Deem adjective as sentiment-phrase, noun as aspect

deeming adjective to be the sentiment-phrase and noun to be the aspect works surprisingly well. In precision terms, that is. Not recall because this pattern is too-specific. For example, it doesn't detect the aspect-sentiment phrase in *Motion lags a bit*.

So how can we try to extend the idea of the previous paragraph to try to improve recall? Formulate this as a sequence labeling problem. See [3] for a detailed sequence-labeling formulation of a similar problem, *named entity recognition*.

The text is tokenized as a sequence of words. Associated with this sequence is a label sequence, which indicates what is the *aspect* and what the *sentiment-phrase*.

Below is our earlier example, reformulated in this convention, with A denoting aspect, S denoting sentiment-phrase, and N denoting neither. We've split the pair into two as it won't fit in a horizontal line.

words Good price. Sharp image. Vivid colors. Static in Audio.

labels S A S A S A S N A words Motion lags a bit.

labels A S S S

In [3] we focused on Hidden Markov models for sequence labeling. Here, it is more natural to work with conditional Markov models [4], for reasons we explain below.

First, what is a conditional Markov model? Recall that our inference problem is to input a sequence of

words and find the most likely sequence of labels for it. For the token sequence [*Motion, lags, a, bit*] we would expect the best label sequence to be [A, S, S, S].

A conditional Markov model (CMM) models this inference problem as one of finding the label sequence  $L$  that maximizes the conditional probability  $P(L|T)$  for the given token sequence  $T$ . The Markov model makes certain assumptions which make this inference problem tractable. Specifically,  $P(L|T)$  is assumed to be factorable as

$$P(L|T) = P(L_1|L_0, T_1) * P(L_2|L_1, T_2) * \dots * P(L_n|L_{n-1}, T_n)$$

Rather than explain it, let's illustrate it with our example. We have added a label  $B$  denoting begin.

$$P([B, A, S, S, S] | [B, \textit{Motion}, \textit{lags}, a, \textit{bit}]) = P(A|B, \textit{Motion}) * P(S|A, \textit{lags}) * P(S|S, a) * P(S|S, \textit{bit})$$

We won't describe the inference algorithm. It is too complex for this post. Besides, this is not our focus. However, we will explain the individual probabilities in the above example qualitatively. Equipped with such an explanation, we can imagine trying out all possible label sequences, computing the probability of each, and finding the one that has the highest probability.

Let's start with  $P(A|B, \textit{Motion})$ . This is influenced by two factors and their interaction. First, the likelihood that the first word is part of the aspect. Second, the likelihood that *Motion* is an aspect word.

The first factor's likelihood is significantly greater than 0. We can imagine many real examples in which the first word is an aspect word. Such as *camera is low-resolution*.

It is the second factor's likelihood that we'd like to dwell more on. Consider  $P(A|\textit{Motion})$ , ignoring the influence of the previous state  $B$ . The CMM allows us to model this probability as being influenced by any features of our choice derived from the combination of  $A$  and *Motion*. Possibly overlapping. The HMM, by contrast, would work in terms of  $P(\textit{Motion}|A)$  instead. It would treat *Motion* and  $A$  as symbols, not letting us exploit any features we may deem useful.

In effect, we can think of  $P(A|Motion)$  as a supervised learning problem in which  $(A, Motion)$  is the input and  $P(A|Motion)$  the output. The power of this approach lies in its ability to learn complex mappings  $P(L_i|T_i)$  in which we can use whatever features from the pair  $(L_i, T_i)$  that we deem fit.

Two features especially come to mind. The word's part-of-speech and whether the word is labeled as being in a recognized named entity. (See [3] which covers named entity recognition in NLP with many real-world use cases and methods.)

The part-of-speech feature has already been suggested by the examples we saw, in which the POS-tag *noun* seemed a predictor of the label *aspect* and *adjective* a predictor of *sentiment-phrase*. The named entity feature is motivated by the intuition that aspects are often objects of specific types. For instance, retail products. A NER that can recognize retail products and associated product features can be very useful to pick these out as *aspects* from sentiment-laden reviews.

Typically we set up NER to recognize fine-grained entities. Such as *product names*. Not *noun phrases*. While in principle we could, noun phrases are too varied to model as NER. POS-tag is coarser-grained. In view of this, we can think of the benefit of combining the two features as follows. NER gives us precision. The POS feature helps with recall.

### **1.2.1 Tools and technology used**

Natural language processing (NLP) is the branch of artificial intelligence (AI) that deals with training a computer to understand, process, and generate language. Search engines, machine translation services, and voice assistants are all powered by the technology.

While the term originally referred to a system's ability to read, it's since become a colloquialism for all computational linguistics. Subcategories include natural language generation (NLG) — a computer's ability to create communication of its own — and natural language understanding (NLU) — the ability to understand slang, mispronunciations, misspellings, and other variants in language.

### **1.2.2 How natural language processing works**

NLP works through machine learning (ML). Machine learning systems store words and the ways they come together just like any other form of data. Phrases, sentences, and sometimes entire books

are fed into ML engines where they're processed using grammatical rules, people's real-life linguistic habits, or both. The computer then uses this data to find patterns and extrapolate what comes next. Take translation software, for example: In French, "I'm going to the park" is "*Je vais au parc*," so machine learning predicts that "I'm going to the store" will also begin with "*Je vais au*." All the computer needs after that is the word for "store."

### 1.2.3 NLP applications

Machine translation is a powerful NLP application, but search is the most used. Every time you look something up in Google or Bing, you're feeding data into the system. When you click on a search result, the system interprets it as confirmation that the results it has found are correct and uses this information to better search in the future.

0 seconds of 26 minutes, 50 seconds Volume 0%

Chatbots work the same way: They integrate with Slack, Microsoft Messenger, and other chat programs where they read the language you use, then turn on when you type in a trigger phrase. Voice assistants such as Siri and Alexa also kick into gear when they hear phrases like "Hey, Alexa." That's why critics say these programs are always listening: If they weren't, they'd never know when you need them. Unless you turn an app on manually, NLP programs must operate in the background, waiting for that phrase.

### 1.2.4 Sequential Minimal Optimization

Most machine learning libraries use the SMO algorithm or some variation.

The SMO algorithm will solve the following optimization problem:

$$\begin{aligned} \text{minimize}_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^m \alpha_i \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \text{ for any } i = 1, \dots, m \\ & \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned}$$

It is a kernelized version of the soft-margin formulation we saw in Chapter 5. The objective function we are trying to minimize can be written in Python (Code Listing 37):

This is the same problem we solved using CVXOPT. Why do we need another method? Because we would like to be able to use SVMs with big datasets, and using convex optimization packages usually involves matrix operations that take a lot of time as the size of the matrix increases or become impossible because of memory limitations. The SMO algorithm has been created with the goal of being faster than other methods.

The idea behind SMO

---

When we try to solve the SVM optimization problem, we are free to change the values of  $\alpha$  as long as we respect the constraints. Our goal is to modify  $\alpha$  so that in the end, the objective function returns the smallest possible value. In this context, given a vector  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_m)$  of Lagrange multipliers, we can change the value of any  $\alpha_i$  until we reach our goal.

The idea behind SMO is quite easy: we will solve a simpler problem. That is, given a vector  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_m)$ , we will allow ourselves to change only two values of  $\alpha$ , for instance,  $\alpha_3$  and  $\alpha_7$ . We will change them until the objective function reaches its minimum given this set of alphas. Then we will pick two other alphas and change them until the function returns its smallest value, and so on. If we continue doing that, we will eventually reach the minimum of the objective function of the original problem.

SMO solves a sequence of several simpler optimization problems.

How did we get to SMO?

---

This idea of solving several simpler optimization problems is not new. In 1982, Vapnik proposed a method known as “chunking,” which breaks the original problem down into a series of smaller problems (Vapnik V. , 1982). What made things change is that in 1997, Osuna, et al., proved that solving a sequence of sub-problems will be guaranteed to converge as long as we add at least one example violating the KKT conditions (Osuna, Freund, & Girosi, 1997).

Using this result, one year later, in 1998, Platt proposed the SMO algorithm.

Why is SMO faster?

---

The great advantage of the SMO approach is that we do not need a QP solver to solve the problem for two Lagrange multipliers—it can be solved analytically. As a consequence, it does not need to store a huge matrix, which can cause problems with machine memory. Moreover, SMO uses several heuristics to speed up the computation.

The SMO algorithm

---

The SMO algorithm is composed of three parts:

- One heuristic to choose the first Lagrange multiplier
- One heuristic to choose the second Lagrange multiplier
- The code to solve the optimization problem analytically for the two chosen multipliers

Tip: A Python implementation of the algorithm is available in Appendix B: The SMO Algorithm. All code listings in this section are taken from this appendix and do not work alone.

### The analytical solution

At the beginning of the algorithm, we start with a vector  $\alpha(\alpha_1, \alpha_2, \dots, \alpha_m)$  in which  $\alpha_i = 0$  for all  $i = 1, \dots, m$ . The idea is to pick two elements of this vector, which we will name  $\alpha_1$  and  $\alpha_2$ , and to change their values so that the constraints are still respected.

The first constraint  $0 \leq \alpha_i \leq C$ , for any  $i = 1, \dots, m$  means that  $0 \leq \alpha_1 \leq C$  and  $0 \leq \alpha_2 \leq C$ . That is why we are forced to select a value lying in the blue box of Figure 50 (which displays an example where  $C = 5$ ).

$$\sum_{i=1}^m \alpha_i y_i = 0$$

The second constraint is a linear constraint  $\sum_{i=1}^m \alpha_i y_i = 0$ . It forces the values to lie on the red diagonal, and the first couple of selected  $\alpha_1$  and  $\alpha_2$  should have different labels ( $y_1 \neq y_2$ ).



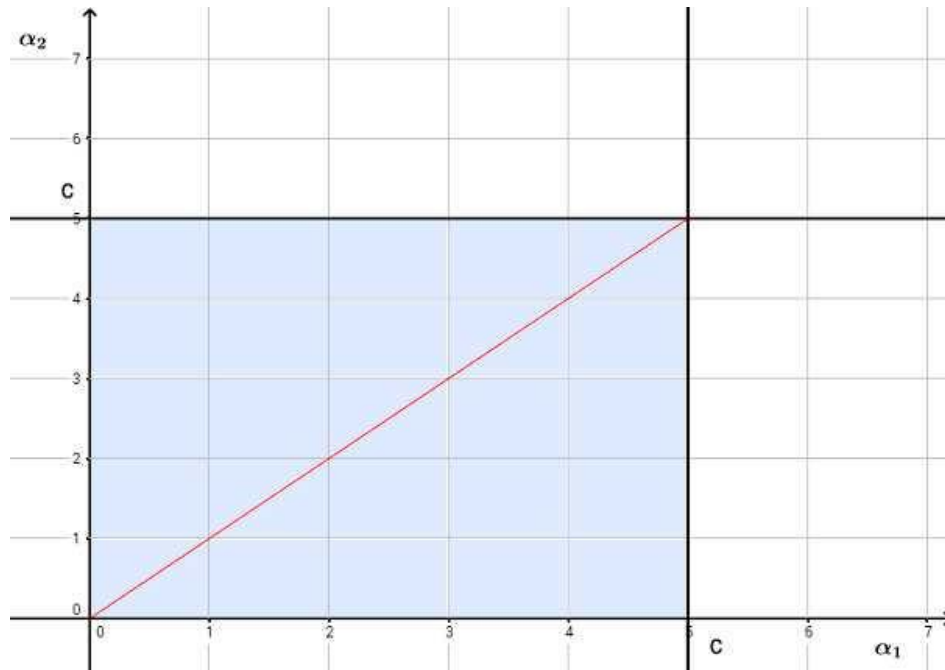


Figure 50: The feasible set is the diagonal of the box

In general, to avoid breaking the linear constraint, we must change the multipliers so that:

$$\alpha_1 y_1 + \alpha_2 y_2 = \text{constant} = \alpha_1^{\text{old}} y_1 + \alpha_2^{\text{old}} y_2$$

We will not go into the details of how the problem is solved analytically, as it is done very well in (Cristianini & Shawe-Taylor, 2000) and in (Platt J. C., 1998).

Remember that there is a formula to compute the new  $\alpha_2$ :

$$\alpha_2^{\text{new}} = \alpha_2 + \frac{y_2(E_1 - E_2)}{K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)}$$

with  $E_i = f(\mathbf{x}_i) - y_i$  being the difference between the output of the hypothesis function and the example label.  $K$  is the kernel function. We also compute bounds, which applies to  $\alpha_2^{\text{new}}$ ; it cannot be smaller than the lower bound, or larger than the upper bound, or constraints will be violated. So  $\alpha_2^{\text{new}}$  is clipped if this is the case.

Once we have this new value, we use it to compute the new  $\alpha_1$  using this formula:

$$\alpha_1^{new} = \alpha_1^{old} + y_1 y_2 (\alpha_2^{old} - \alpha_2^{new})$$

### Understanding the first heuristic

The idea behind the first heuristic is pretty simple: each time SMO examines an example, it checks whether or not the KKT conditions are violated. Recall that at least one KKT condition must be violated. If the conditions are met, then it tries another example. So if there are millions of examples, and only a few of them violate the KKT conditions, it will spend a lot of time examining useless examples. In order to avoid that, the algorithm concentrates its time on examples in which the Lagrange multiplier is not equal to 0 or  $C$ , because they are the most likely to violate the conditions (Code Listing 38).

Because solving the problem analytically involves two Lagrange multipliers, it is possible that a bound multiplier (whose value is between 0 and  $C$ ) has become KKT-violated. That is why the main routine alternates between all examples and the non-bound subset (Code Listing 39). Note that the algorithm finishes when progress is no longer made.

### Understanding the second heuristic

The goal of this second heuristic is to select the Lagrange multiplier for which the step taken will be maximal.

How do we update  $\alpha_2$ ? We use the following formula:

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)}$$

Remember that in this case that we have already chosen the value  $\alpha_1$ . Our goal is to pick the  $\alpha_2$  whose  $\alpha_2^{new}$  will have the biggest change. This formula can be rewritten as follows:

$$\alpha_2^{new} = \alpha_2 + \text{step}$$

with:

$$\text{step} = \frac{y_2(E_1 - E_2)}{K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)}$$

So, to pick the best  $\alpha_2$  amongst several  $\alpha_i$ , we need to compute the value of step for each  $\alpha_i$  and select the one with the biggest step. The problem here is that we need to call the kernel function  $K$  three times for each step, and this is costly. Instead of doing that, Platt came with the following approximation:

$$\text{step} \approx |E_1 - E_2|$$

As a result, selecting the biggest step is done by taking the  $\alpha_i$  with the smallest error if  $E_1$  is positive, and the  $\alpha_i$  with the biggest error if  $E_1$  is negative.

## CHAPTER-2 Literature Survey

There are fundamental recipes to exhume novelties from content. They represent methodologies and gimmicks growing acquainted with systems (1). The accompanying two areas edit to these methodologies. A. Representative Tactics Tectonic of the examination in unsupervised conclusion characterization using symbolic procedures makes maltreatment of accessible wordy worth. Turney (2) utilized pack-of-words approach for assessment examination. In that approach, connections between the individual words aren't considered and a report is spoken to as a minor gathering of words. To decide those grades are joined with some total capacities.

He introduces the extremity of a review in view of the normal semantic prelude of tuples separated from the look-see where tuples are idioms having descriptors or verb modifiers. He introduces the semantic prelude of tuples harnessing the internet quest Altavista. Kampeska. (3) harnessed the vocabular database WordNet (4) to decide the passionate substance of a word along polychrome measures. They pieced up a separation metric on WordNet and decided the semantic proem of descriptors. WordNet database comprises of words associated by first word relations. Baronial. (5) raised up a shell operating word space show formalism that beats the trouble in wordy bargaining assignment. It speaks to the In setting of a word alongside its general disbandment. Balashur. (6) presented Emote Net, an applied delineation of content. that stores the structure and the semantics of genuine occasions for a particular space. Emogine applied the idea of Finite State Automata to distinguish the passionate replies cranked by exercise. One of the members of Sem Eval 2007 Task No. 14 (7) exercised coarse grained and fine grainy ways to deal with distinguish assessments in news features. In coarse grainy approach, they performed double characterization of passions and in fine granular approach they grouped passions into polychromatic reaches. Knowledge base approach is institute to be sensitive due to the necessary of a huge vocabular database. Since social network generates huge quantum of data every second, sometimes larger than the size of available lexical database, sentiment analysis got tedious and incorrect.

B. Machine learning techniques widget acing strategies use an instruction set and a test set for order. Tutelage set contains input work vectors and their comparing class names. using this prep total set, a type display is developed which whacks to order the. information highlight vectors into comparing class names. at that point a test set is operated to ratify the adjustment by foreknowing

the class names of unobtrusive element vectors. rainbow jigger picking up literacy of methodologies like credulous bayes (n b). Last extreme entropy (me), and bolster vector machines (s v m) are . applied to arrange heartstrings (8). Dominos etal. (9) introduce that real bayes works legitimately for specific issues with really settled capacities. that's shocking as the central supposition of innocent bayes is that the capacities are fair. zhenniu etal. (10) presented a fresh eluding of the box new model wherein effective methodologies are exercised for work determination, weight Calculus and class. the new form is construct absolutely with respect to Bayesian arrangement of tenets. ideal presently weights of the classifier are balanced by form for employing consigliere trademark and specific capacity.' consigliere highlight's the records that

speaks to a class and' specific element's the information that encourages in feting directions. the application of the bones weights, they figured the liability of every class and henceforth . ventured forward the Bayesian arrangement of rules. Barbosa teal. (11) outlined a 2- step programmed opinion assessment form for arranging tweets. they employed a knockabout instruction set to lessen

the marking crack in developing classifiers. above all else, hey sorted tweets into peculiar and objective tweets. from that point forward, peculiar tweets are named as gigantic and negative tweets. Celik Yilmaz teal. (12) propelled an oratory principally hung expression grouping approach for homogenizing boisterous tweets. in oratory rested word grouping, idioms having. suchlike articulation is bunched and doled out typical souvenirs. They likewise used content handling systems like doling out comparable souvenirs for reckoning, html joins, punter identifiers, and objective business undertaking names for standardization. later to doing standardization, they harnessed probabilistic models to distinguish extremity vocabularies. they performed class the perversion of the boos texter classifier with these extremity wordbooks as capacities and bagged a de-escalated bungles bring. Wu etal. (13) proposed an affect break show for twitter assessment. in the event that username is deposited inside the capsule of a tweet, it's. far impacting development and it adds to affecting Shot. any tweet that begins with username is a rewet that speaks to a motivated movement and it adds to empowered probability. they establish that there is a solid relationship among these chances. Pak

etal.(14) made a twitter corpus through mechanically storing up tweets the misemployment of twitter programming interface and routinely noting on the bones the use of feelings. exercising that corpus, . they erected an estimation classifier constitutionally hung at the multinomial innocent bayes classifier that utilizations n-gram and Pos Tickets as faculties. in that Approach, there's a hazard of mistake for the reason that heartstrings of tweets in preparing set are arranged simply largely grounded at the extremity of chords. the instruction set is in like manner less ace since it contains most straightforward tweets having Chords.

There is various type of information uploaded and shared on social media in the form of text, videos, photos and audio [15]. Social media is rich with raw and unprocessed data and the improvement in technology, especially in machine learning and artificial intelligence, allow the data to be processed and converted it into a useful data that they can benefit most business organization [16]. This paper focuses to provide a better understanding of the application of sentiment analysis in social media platform by examining related literature published between 2014 to 2019. Sentiment analysis is an approach that uses Natural Language Processing (NLP) to extract, convert and interpret opinion from a text and classify them into positive, negative or natural sentiment [17]. Most of the previous study applied sentiment analysis into a product or movie review to better understand their customer and make the necessary decision to improve their product or services [18]. Scholars have been conducting a study on sentiment analysis since the last decade which most papers started to appear and rapidly growing after the year 2004 [19]. Sentiment analysis is divided into three different levels which are sentence level, document level and feature level. The purpose is to classify the opinion either from sentence, document or features into positive and negative sentiment [20]. There are 2 main methods of sentiment analysis have been identified which is a machine learning approach and lexicon-based approach. Machine learning approach utilized algorithms to extract and detect sentiment from a data while lexicon-based approach works by counting the positive and negative words that related to the data. Scholars have been developing a new effective and accurate model in sentiment analysis. But there is a challenge arise in developing a model where most of it is design for the English language. But a recent study shows that there is sentiment analysis model design in other languages such as Korean [21], Thailand [22], Arabic [23], Malay [24], Portuguese [25] and Chinese [26].

## CHAPTER-3

### Functionality of Project

#### 3.1 Data

To gather the data many options are possible. In some previous paper researches, they built a program to collect automatically a corpus of tweets based on two classes, “positive” and “negative”, by querying Twitter with two type of emoticons:

- Happy emoticons, such as “:)””, “:P””, “:-)”” etc.
- Sad emoticons, such as “:(“”, “:’(“”, “=(““.

Others make their own dataset of tweets by collecting and annotating them manually which very long and fastidious.

Additionally to find a way of getting a corpus of tweets, we need to take of having a balanced data set, meaning we should have an equal number of positive and negative tweets, but it needs also to be large enough. Indeed, more the data we have, more we can train our classifier and more the accuracy will be.

After many researches, I found a dataset of 1578612 tweets in english coming from two sources: Kaggle and Sentiment140. It is composed of four columns that are *ItemID*, *Sentiment*, *SentimentSource* and *SentimentText*. We are only interested by the *Sentiment* column corresponding to our label class taking a binary value, 0 if the tweet is negative, 1 if the tweet is positive and the *SentimentText* columns containing the tweets in a raw format.

	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my APL friend.....
1	2	0	Sentiment140	I missed the New Moon trailer...
2	3	1	Sentiment140	omg its already 7:30 :O
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRY. I've been at this dentist since 11.. I was supposed to just get a crown put on (30mins)...
4	5	0	Sentiment140	i think mi bf is cheating on me!! T_T
5	6	0	Sentiment140	or i just worry too much?
6	7	1	Sentiment140	Juuuuuuuuuuuuuuuuuuuuuuuussssst Chillin!!
7	8	0	Sentiment140	Sunny Again Work Tomorrow :-  TV Tonight
8	9	1	Sentiment140	handed in my uniform today . i miss you already
9	10	1	Sentiment140	hmmmm.... i wonder how she my number @-)

Table 3.1.1: Example of twitter posts annotated with their corresponding sentiment, 0 if it is negative, 1 if it is positive.

In the Table 3.1.1 showing the first ten twitter posts we can already notice some particularities and difficulties that we are going to encounter during the preprocessing steps.

- The presence of **acronyms** "bf" or more complicated "APL". Does it mean apple ? Apple (the company) ? In this context we have "friend" after so we could think that he refers to his smartphone and so Apple, but what about if the word "friend" was not here ?
- The presence of **sequences of repeated characters** such as "Juuuuuuuuuuuuuuuuuuuussssst", "hmmmm". In general when we repeat several characters in a word, it is to emphasize it, to increase its impact.
- The presence of **emoticons**, ":O", "T\_T", ":-|" and much more, give insights about user's moods.
- **Spelling mistakes** and “urban grammar” like "im gunna" or "mi".
- The presence of **nouns** such as "TV", "New Moon".

Furthermore, we can also add,

- People also indicate their moods, emotions, states, between two *such as*, \*|cries\*,

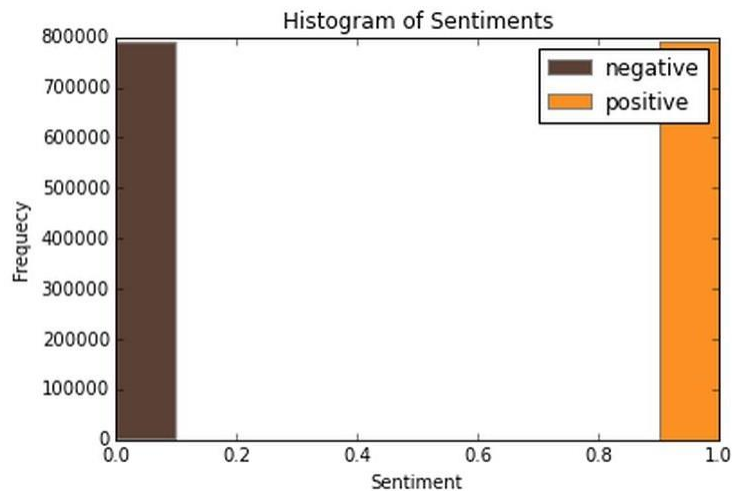


\*hummin\*, \*sigh\*.

- The negation, “can't”, “cannot”, “don't”, “haven't” that we need to handle like: “I don't like chocolate”, “like” in this case is negative.

We could also be interested by the grammar structure of the tweets, or if a tweet is subjective/objective and so on. As you can see, it is **extremely complex** to deal with languages and even more when we want to analyse text typed by users on the Internet because people don't take care of making sentences that are grammatically correct and use a ton of acronyms and words that are more or less english in our case.

We can visualize a bit more the dataset by making a chart of how many positive and negative tweets does it contains,



*Figure 3.1.1: Histogram of the tweets according to their sentiment*

We have exactly 790177 positive tweets and 788435 negative tweets which signify that the dataset is well-balanced. There is also no duplicates.

Finally, let's recall the Twitter terminology since we are going to have to deal with in the tweets:

- **Hashtag:** A hashtag is any word or phrase immediately preceded by the # symbol. When you click on a hashtag, you'll see other Tweets containing the same keyword or topic.
- **@username:** A username is how you're identified on Twitter, and is always preceded immediately by the @ symbol. For instance, Katy Perry is @katyperry.
- **MT:** Similar to RT (Retweet), an abbreviation for "Modified Tweet." Placed before the Retweeted text when users manually retweet a message with modifications, for example shortening a Tweet.
- **Retweet:** RT, A Tweet that you forward to your followers is known as a Retweet. Often used to pass along news or other valuable discoveries on Twitter, Retweets always retain original attribution.
- **Emoticons:** Composed using punctuation and letters, they are used to express emotions concisely, ";) :)" ...". Now we have the corpus of tweets, we need to use other resources to make easier the pre-processing step.

### 3.2 Resources

In order to facilitate the pre-processing part of the data, we introduce five resources which are,

- An **emoticon dictionary** regrouping 132 of the most used emoticons in western with their sentiment, negative or positive.
- An **acronym dictionary** of 5465 acronyms with their translation.
- A **stop word dictionary** corresponding to words which are filtered out before or after processing of natural language data because they are not useful in our case.
- A **positive and negative word dictionaries** given the polarity (sentiment out-of-context) of words.

- A **negative contractions and auxiliaries dictionary** which will be used to detect negation in a given tweet such as “don’t”, “can’t”, “cannot”, etc.

The introduction of these resources will allow to uniform tweets and remove some of their complexities with the acronym dictionary for instance because a lot of acronyms are used in tweets. The positive and negative word dictionaries could be useful to increase (or not) the accuracy score of the classifier. The emoticon dictionary has been built from wikipedia with each emoticon annotated manually. The stop word dictionary contains 635 words such as “the”, “of”, “without”. Normally they should not be useful for classifying tweets according to their sentiment but it is possible that they are.

Also we use Python 2.7 (<https://www.python.org/>) which is a programming language widely used in data science and scikit-learn (<http://scikit-learn.org/>) a very complete and useful library for machine learning containing every techniques, methods we need and the website is also full of tutorials well-explained. With Python, the libraries, Numpy (<http://www.numpy.org/>) and Panda (<http://pandas.pydata.org/>) for manipulating data easily and intuitively are just essential.

### 3.3 Pre-processing

Now that we have the corpus of tweets and all the resources that could be useful, we can pre-process the tweets. It is a very important since all the modifications that we are going to during this process will directly impact the classifier's performance. The pre-processing includes cleaning, normalization, transformation, feature extraction and selection, etc. The result of pre-processing will be consistent and uniform data that are workable to maximize the classifier's performance.

All of the tweets are pre-processed by passing through the following steps in the same order.

#### 3.3.1 Emoticons

We replace all emoticons by their sentiment polarity **||pos||** and **||neg||** using the emoticon dictionary. To do the replacement, we pass through each tweet and by using a regex we find out if it contains emoticons, if yes they are replaced by their corresponding polarity.

	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my APL friend.....
1	2	0	Sentiment140	I missed the New Moon trailer...
2	3	1	Sentiment140	omg its already 7:30 :O
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I've been at this dentist since 11.. I was suposed 2 just get a crown put on (30mins)...
4	5	0	Sentiment140	i think mi bf is cheating on me!! T_T
5	6	0	Sentiment140	or i just worry too much?
6	7	1	Sentiment140	Juuuuuuuuuuuuuuuuuuu Chillin!!
7	8	0	Sentiment140	Sunny Again Work Tomorrow :-  TV Tonight
8	9	1	Sentiment140	handed in my uniform today . i miss you already
9	10	1	Sentiment140	hmmm.... i wonder how she my number @-

*Table 3.3.1.1: Before processing emoticons, list of tweets where some of them contain emoticons.*

	<b>ItemID</b>	<b>Sentiment</b>	<b>SentimentSource</b>	<b>SentimentText</b>
<b>0</b>	1	0	Sentiment140	is so sad for my APL friend.....
<b>1</b>	2	0	Sentiment140	I missed the New Moon trailer...
<b>2</b>	3	1	Sentiment140	omg its already 7:30   pos
<b>3</b>	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRY. I've been at this dentist since 11.. I was suposed 2 just get a crown put on (30mins)...
<b>4</b>	5	0	Sentiment140	i think mi bf is cheating on me!!!   neg
<b>5</b>	6	0	Sentiment140	or i just worry too much?
<b>6</b>	7	1	Sentiment140	Juuuuuuuuuuuuuuuuussssst Chillin!!
<b>7</b>	8	0	Sentiment140	Sunny Again Work Tomorrow   neg   TV Tonight
<b>8</b>	9	1	Sentiment140	handed in my uniform today . i miss you already
<b>9</b>	10	1	Sentiment140	hmmmm.... i wonder how she my number   pos

*Table 3.3.1.2: After processing emoticons, they have been replaced by their corresponding tag*

The data set contains 19469 positive emoticons and 11025 negative emoticons.

### 3.3.1 URLs

We replace all URLs with the tag `||url||`. There is about 73824 urls in the data set and we proceed as the same way we did for the emoticons.

	ItemID	Sentiment	SentimentSource	SentimentText
50	51	0	Sentiment140	baddest day eveer.
51	52	1	Sentiment140	bathroom is clean..... now on to more enjoyable tasks.....
52	53	1	Sentiment140	boom boom pow
53	54	0	Sentiment140	but i'm proud.
54	55	0	Sentiment140	congrats to helio though
55	56	0	Sentiment140	David must be hospitalized for five days end of July (palatine tonsils). I will probably never see Katie in concert.
56	57	0	Sentiment140	friends are leaving me 'cause of this stupid love <a href="http://bit.ly/ZoxZC">http://bit.ly/ZoxZC</a>
57	58	1	Sentiment140	go give ur mom a hug right now. <a href="http://bit.ly/azFwv">http://bit.ly/azFwv</a>
58	59	1	Sentiment140	Going To See Harry Sunday Happiness
59	60	0	Sentiment140	Hand quilting it is then...

*Table 3.3.2.1: Tweets before processing URLs.*

	ItemID	Sentiment	SentimentSource	SentimentText
50	51	0	Sentiment140	baddest day eveer.
51	52	1	Sentiment140	bathroom is clean..... now on to more enjoyable tasks.....
52	53	1	Sentiment140	boom boom pow
53	54	0	Sentiment140	but i'm proud.
54	55	0	Sentiment140	congrats to helio though
55	56	0	Sentiment140	David must be hospitalized for five days end of July (palatine tonsils). I will probably never see Katie in concert.
56	57	0	Sentiment140	friends are leaving me 'cause of this stupid love   url
57	58	1	Sentiment140	go give ur mom a hug right now.   url
58	59	1	Sentiment140	Going To See Harry Sunday Happiness
59	60	0	Sentiment140	Hand quilting it is then...

*Table 3.3.2.2: Tweets after processing URLs.*

### 3.3.2 Unicode

For simplicity and because the ASCII table should be sufficient, we choose to remove any unicode character that could be misleading for the classifier.

	ItemID	Sentiment	SentimentSource	SentimentText
1578592	1578608	1	Sentiment140	'Zu SpÃœt' by Die Ã„rzte. One of the best bands ever
1578593	1578609	1	Sentiment140	Zuma bitch tomorrow. Have a wonderful night everyone goodnight.
1578594	1578610	0	Sentiment140	zummie's couch tour was amazing....to bad i had to leave early
1578595	1578611	0	Sentiment140	ZuneHD looks great! OLED screen @720p, HDMI, only issue is that I have an iPhone and 2 iPods . MAKE IT A PHONE and ill buy it @micro...
1578596	1578612	1	Sentiment140	zup there ! learning a new magic trick
1578597	1578613	1	Sentiment140	zyklonic showers *evil*
1578598	1578614	1	Sentiment140	ZZ Top â€œ I Thank You ...@hawaiibuzz .....Thanks for your music and for your ear(s) ...ALL !!!! Have a fab... â™«   url
1578599	1578615	0	Sentiment140	zzz time. Just wish my love could B nxt 2 me
1578600	1578616	1	Sentiment140	zzz twitter. good day today. got a lot accomplished. imstorm. got into it w yet another girl. dress shopping tmrw
1578601	1578617	1	Sentiment140	zzz's time, goodnight.   url

Table 3.3.3.1: Tweets before processing Unicode.

	ItemID	Sentiment	SentimentSource	SentimentText
1578592	1578608	1	Sentiment140	'Zu Spt' by Die rzte. One of the best bands ever
1578593	1578609	1	Sentiment140	Zuma bitch tomorrow. Have a wonderful night everyone goodnight.
1578594	1578610	0	Sentiment140	zummie's couch tour was amazing....to bad i had to leave early
1578595	1578611	0	Sentiment140	ZuneHD looks great! OLED screen @720p, HDMI, only issue is that I have an iPhone and 2 iPods . MAKE IT A PHONE and ill buy it @micro...
1578596	1578612	1	Sentiment140	zup there ! learning a new magic trick
1578597	1578613	1	Sentiment140	zyklonic showers *evil*
1578598	1578614	1	Sentiment140	ZZ Top I Thank You ...@hawaiibuzz .....Thanks for your music and for your ear(s) ...ALL !!!! Have a fab...   url
1578599	1578615	0	Sentiment140	zzz time. Just wish my love could B nxt 2 me
1578600	1578616	1	Sentiment140	zzz twitter. good day today. got a lot accomplished. imstorm. got into it w yet another girl. dress shopping tmrw
1578601	1578617	1	Sentiment140	zzz's time, goodnight.   url

Table 3.3.3.1: Tweets after processing Unicode.

### 3.4.1 HTML entities

HTML entities are characters reserved in HTML. We need to decode them in order to have characters entities to make them understandable.

```
' Cannot get chatroom feature to work. Updated Java to 10, checked ports, etc. I can see video, but in the &quot;chat,&quot; only a spinning circle. '
```

*Figure 3.3.4.1: A tweet before processing HTML entities.*

```
u' Cannot get chatroom feature to work. Updated Java to 10, checked ports, etc. I can see video, but in the "chat," only a spinning circle. '
```

*Figure 3.3.4.2: A tweet after processing HTML entities.*

### 2.3.1 Case

The case is something that can appears useless but in fact it is really important for distinguish proper noun and other kind of words. Indeed: “General Motor” is the same thing that “general motor”, or “MSc” and “msc”. So reduce all letters to lowercase should be normally done wisely. In this project, for simplicity we will not take care of that since we assume that it should not impact too much the classifier’s performance.

	ItemID	Sentiment	SentimentSource	SentimentText
1578592	1578608	1	Sentiment140	'Zu Spt' by Die rzte. One of the best bands ever
1578593	1578609	1	Sentiment140	Zuma bitch tomorrow. Have a wonderful night everyone goodnight.
1578594	1578610	0	Sentiment140	zummie's couch tour was amazing....to bad i had to leave early
1578595	1578611	0	Sentiment140	ZuneHD looks great! OLED screen @720p, HDMI, only issue is that I have an iPhone and 2 iPods . MAKE IT A PHONE and ill buy it @micro...
1578596	1578612	1	Sentiment140	zup there ! learning a new magic trick
1578597	1578613	1	Sentiment140	zyklonic showers *evil*
1578598	1578614	1	Sentiment140	ZZ Top I Thank You ...@hawaiibuzz .....Thanks for your music and for your ear(s) ...ALL !!!! Have a fab...   url
1578599	1578615	0	Sentiment140	zzz time. Just wish my love could B nxt 2 me
1578600	1578616	1	Sentiment140	zzz twitter. good day today. got a lot accomplished. imstorm. got into it w yet another girl. dress shopping tmrw
1578601	1578617	1	Sentiment140	zzz's time, goodnight.   url

*Table 3.3.5.1: Tweets before processing lowercase.*



ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140 is so sad for my apl friend.....
1	2	0	Sentiment140 i missed the new moon trailer...
2	3	1	Sentiment140 omg its already 7:30   pos
3	4	0	Sentiment140 .. omgaga. im sooo im gunna cry. i've been at this dentist since 11.. i was suposed 2 just get a crown put on (30mins)...
4	5	0	Sentiment140 i think mi bf is cheating on me!!!   neg
5	6	0	Sentiment140 or i just worry too much?
6	7	1	Sentiment140 juuuuuuuuuuuuuuuuusssst chillin!!
7	8	0	Sentiment140 sunny again work tomorrow   neg   tv tonight
8	9	1	Sentiment140 handed in my uniform today . i miss you already
9	10	1	Sentiment140 hmhhh.... i wonder how she my number   pos

Table 3.3.5.2: Tweets after processing lowercase

### 3.3.1 Targets

The target correspond to usernames in twitter preceded by “@” symbol. It is used to address a tweet to someone or just grab the attention. We replace all usernames/targets by the tag ||target||. Notice that in the data set we have 735757 targets.

ItemID	Sentiment	SentimentSource	SentimentText
45	46	1	Sentiment140 @ginaaa <3 go to the show tonight
46	47	0	Sentiment140 @spiral_galaxy @ymptweet it really makes me sad when i look at muslims reality now
47	48	0	Sentiment140 - all time low shall be my motivation for the rest of the week.
48	49	0	Sentiment140 and the entertainment is over, someone complained properly.. @rupturereapture experimental you say? he should experiment with a me...
49	50	0	Sentiment140 another year of lakers .. that's neither magic nor fun ...
50	51	0	Sentiment140 baddest day ever.
51	52	1	Sentiment140 bathroom is clean..... now on to more enjoyable tasks.....
52	53	1	Sentiment140 boom boom pow
53	54	0	Sentiment140 but i'm proud.
54	55	0	Sentiment140 congrats to helio though

Table 3.3.6.1: Tweets before processing targets.

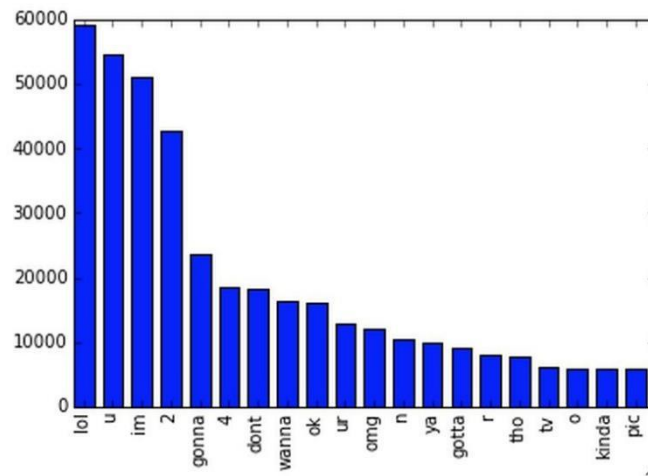
ItemID	Sentiment	SentimentSource	SentimentText
45	46	1	Sentiment140   target   <3 go to the show tonight
46	47	0	Sentiment140   target     target   it really makes me sad when i look at muslims reality now
47	48	0	Sentiment140 - all time low shall be my motivation for the rest of the week.
48	49	0	Sentiment140 and the entertainment is over, someone complained properly..   target   experimental you say? he should experiment with a melody...
49	50	0	Sentiment140 another year of lakers .. that's neither magic nor fun ...
50	51	0	Sentiment140 baddest day ever.
51	52	1	Sentiment140 bathroom is clean..... now on to more enjoyable tasks.....
52	53	1	Sentiment140 boom boom pow
53	54	0	Sentiment140 but i'm proud.
54	55	0	Sentiment140 congrats to helio though

Table 3.3.6.2: Tweets after processing targets.

### 2.3.2 Acronyms

We replace all acronyms with their translation. An acronym is an abbreviation formed from the initial components in a phrase or a word. Usually these components are individual letters (as in NATO or laser) or parts of words or names (as in Benelux). Many acronyms are used in our data set of tweets as you can see in the following bar chart.

At this point, tweets are going to be tokenized by getting rid of the punctuation and using split in order to do the process really fast. We could use `nlk.tokenizer` but it is definitely much much slower (also much more accurate).



*Figure 2.3.7.1: Top 20 of acronyms in the data set of tweets*

As you can see, “lol”, “u”, “im”, “2” are really often used by users. The table below shows the top 20 acronyms with their translation and their count.

1) lol => laughing out loud : 59000  
2) u => you : 54557  
3) im => instant message : 51099  
4) 2 => too : 42645  
5) gonna => going to : 23716  
6) 4 => for : 18610  
7) dont => don't : 18363  
8) wanna => want to : 16357  
9) ok => okay : 16104  
10) ur => your : 12960  
11) omg => oh my god : 12178  
12) n => and : 10415  
13) ya => yeah : 9948  
14) gotta => got to : 9243  
15) r => are : 8132  
16) tho => though : 7696  
17) tv => television : 6246  
18) o => oh : 6002  
19) kinda => kind of : 5953  
20) pic => picture : 5945

*Table 2.3.7.1: Top 20 of acronyms in the data set of tweets with their translation and count*

### 2.3.3 Negation

We replace all negation words such as “not”, “no”, “never” by the tag `||not||` using the negation dictionary in order to take more or less of sentences like "I don't like it". Here like should not be considered as positive because of the "don't" before. To do so we will replace "don't" by `||not||` and the word like will not be counted as positive. We should say that each time a negation is encountered, the words followed by the negation word contained in the positive and negative word dictionaries will be reversed, positive becomes negative, negative becomes positive, we will do this when we will try to find positive and negative words.

```
['i', "didn't", 'realize', 'it', 'was', 'that', 'deep', 'geez', 'give', 'a', 'girl', 'a', 'warning', 'atleast']
```

*Figure 2.3.8.1: A tweet before processing negation words.*

```
['i', '||not||', 'realize', 'it', 'was', 'that', 'deep', 'geez', 'give', 'a', 'girl', 'a', 'warning', 'atleast']
```

*Figure 2.3.8.2: A tweet after processing negation words.*

### 2.3.4 Sequence of repeated characters

Now, we replace all sequences of repeated characters by two characters (e.g: "helloooo" = "hello") to keep the emphasized usage of the word.

	ItemID	Sentiment	SentimentSource	SentimentText
1578604	1578620	1	Sentiment140	[zzzz, no, work, tomorrow, yayyy]
1578605	1578621	1	Sentiment140	[zzzzz, time, tomorrow, will, be, a, busy, day, for, serving, loving, people, love, you, all]
1578606	1578622	0	Sentiment140	[zzzzz, want, to, sleep, but, at, sister's, in, laws's, house]
1578607	1578623	1	Sentiment140	[zzzzzz, finally, night, tweeters]
1578608	1578624	1	Sentiment140	[zzzzzzz, sleep, well, people]
1578609	1578625	0	Sentiment140	[zzzzzzzzzz, wait, no, i, have, homework]
1578610	1578626	0	Sentiment140	[zzzzzzzzzzzzzz, whatever, what, am, i, doing, up, again]
1578611	1578627	0	Sentiment140	[zzzzzzzzzzzzzzzzzz, i, wish]

*Table 2.3.9.1: Tweets before processing sequences of repeated characters.*

	ItemID	Sentiment	SentimentSource	SentimentText
1578604	1578620	1	Sentiment140	[zz, no, work, tomorrow, yayy]
1578605	1578621	1	Sentiment140	[zz, time, tomorrow, will, be, a, busy, day, for, serving, loving, people, love, you, all]
1578606	1578622	0	Sentiment140	[zz, want, to, sleep, but, at, sister's, in, laws's, house]
1578607	1578623	1	Sentiment140	[zz, finally, night, tweeters]
1578608	1578624	1	Sentiment140	[zz, sleep, well, people]
1578609	1578625	0	Sentiment140	[zz, wait, no, i, have, homework]
1578610	1578626	0	Sentiment140	[zz, whatever, what, am, i, doing, up, again]
1578611	1578627	0	Sentiment140	[zz, i, wish]

Table 2.3.9.2: Tweets after processing sequences of repeated characters

### 3.2 Machine Learning

Once we have applied the different steps of the preprocessing part, we can now focus on the machine learning part. There are three major models used in sentiment analysis to classify a sentence into positive or negative: SVM, Naive Bayes and Language Models (N-Gram). SVM is known to be the model giving the best results but in this project we focus only on probabilistic model that are Naive Bayes and Language Models that have been widely used in this field. Let's first introduce the Naive Bayes model which is well-known for its simplicity and efficiency for text classification.

#### 3.4.1 Naive Bayes

In machine learning, 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 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 (mathematical expression that can be evaluated in a finite number of operations), which takes linear time.

It is based on the application of the Baye's rule given by the following formula:

$$P(C = c|D = d) = \frac{P(D = d|C = c)P(C = c)}{P(D = d)}$$

Formula 2.4.1.1: Baye's rule

where  $D$  denotes the document and  $C$  the category (label),  $d$  and  $c$  are instances of  $D$  and  $C$

and  $P(D = d) = \sum_{c \in C} P(D = d|C = c)P(C = c)$ . We can simplify this expression by,

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

*Formula 2.4.1.2: Baye's rule simlified*

In our case, a tweet  $d$  is represented by a vector of  $K$  attributes such as  $d = (w_1, w_2, \dots, w_k)$

Computing  $P(d|c)$  is not trivial and that's why the Naive Bayes introduces the assumption that all of the feature values  $w_j$  are independent given the category label  $c$ . That is, for  $i \neq j$ ,  $w_i$  and  $w_j$  are conditionally independent given the category label  $c$ . So the Baye's rule can be rewritten as,

$$P(c|d) = P(c) \times \frac{\prod_{j=1}^K P(w_j|c)}{P(d)}$$

*Formula 2.4.1.3: Baye's rule rewritten*

Based on this equation, maximum a posterior (MAP) classifier can be constructing by seeking the optimal category which maximizes the posterior  $P(c|d)$ :

$$c^* = \arg \max_{c \in C} P(c|d)$$

$$c^* = \arg \max_{c \in C} \left\{ P(c) \times \frac{\prod_{j=1}^K P(w_j|c)}{P(d)} \right\}$$

$$c^* = \arg \max_{c \in C} \left\{ P(c) \times \prod_{j=1}^K P(w_j|c) \right\}$$

*Formula 2.4.1.4: Classifier maximizing the posterior probability  $P(c|d)$*

Note that  $P(d)$  is removed since it is a constant for every category  $c$ . There are several variants of Naive Bayes classifiers that are:

- The **Multi-variate Bernoulli Model**: Also called binomial model, useful if our feature vectors are binary (e.g 0s and 1s). An application can be text classification with bag of words model where the 0s 1s are "word does not occur in the document" and "word occurs in the document" respectively.
- The **Multinomial Model**: Typically used for discrete counts. In text classification, we extend the Bernoulli model further by counting the number of times a word  $w_i$  appears over the number of words rather than saying 0 or 1 if word occurs or not.

- the **Gaussian Model**: We assume that features follow a normal distribution. Instead of discrete counts, we have continuous features.

The likelihood  $P(w_j|c)$  is usually computed using the formula:

$$P(w_j|c) = \frac{1 + \text{count}(w_j, c)}{|V| + N_c}$$

*Formula 2.4.1.6: Likelihood  $P(w_j|c)$*

where  $\text{count}(w_j, c)$  is the number of times that word  $w_j$  occurs within the training tweets of class  $c$ , and  $|V| = \sum_j$  the size of the vocabulary. This estimation uses the simplest smoothing

method to solve the **zero-probability problem**, that arises when our model encounters a word seen in the test set but not in the training set, **Laplace** or add-one since we use 1 as constant. We will see that Laplace smoothing method is not really effective compared to other smoothing methods used in language models.

### 3.3.2 Baseline

In every machine learning task, it is always good to have what we called a baseline. It often a “quick and dirty” implementation of a basic model for doing the first classification and based on its accuracy, try to improve it.

We use the Multinomial Naive Bayes as learning algorithm with the Laplace smoothing representing the classic way of doing text classification. Since we need to extract features from our data set of tweets, we use the **bag of words model** to represent it.

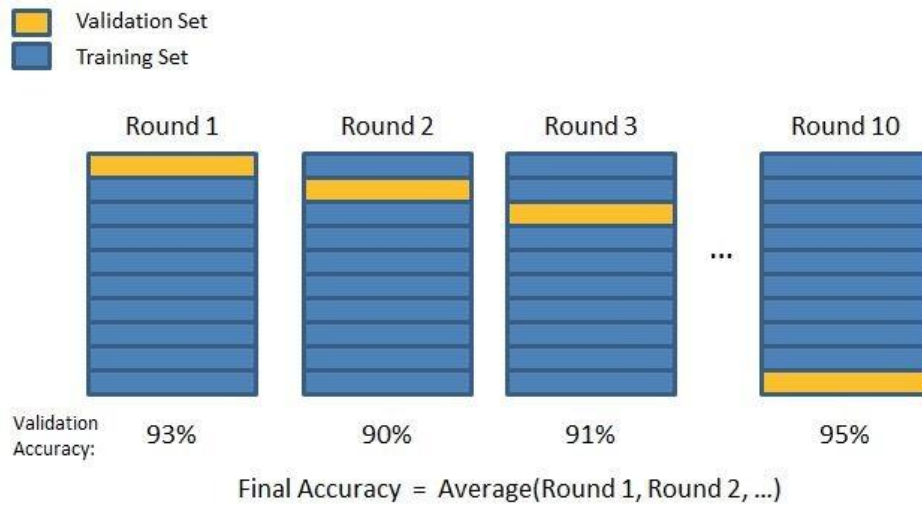


The bag of words model is a simplifying representation of a document where it is represented as a bag of its words without taking consideration of the grammar or word order. In text classification, the count (number of time) of each word appears in a document is used as a feature for training the classifier.

Firstly, we divide the data set into two parts, the training set and the test set. To do this, we first shuffle the data set to get rid of any order applied to the data, then we form the set of positive tweets and the set of negative tweets, we take 3/4 of tweets from each set and merge them together to make the training set. The rest is used to make the test set. Finally the size of the training set is 1183958 tweets and the test set is 394654 tweets. Notice that they are balanced and follow the same distribution of the initial data set.

Once the training set and the test set are created we actually need a third set of data called the **validation set**. It is really useful because it will be used to **validate our model against unseen data and tune the possible parameters** of the learning algorithm to avoid underfitting and overfitting for example. We need this validation set because our test set should be used only to verify how well the model will **generalize**. If we use the test set rather than the validation set, our model could be **overly optimistic and twist the results**. To make the validation set, there are two main options:

- Split the training set into two parts (60%, 20%) with a ratio 2:8 where each part contains an equal distribution of example types. We train the classifier with the largest part, and make prediction with the smaller one to validate the model. This technique works well but has the disadvantage of our classifier not getting trained and validated on all examples in the data set (without counting the test set).
- The **K-fold cross-validation**. We split the data set into k parts, hold out one, combine the others and train on them, then validate against the held-out portion. We repeat that process k times (each fold), holding out a different portion each time. Then we average the score measured for each fold to get a more accurate estimation of our model's performance.



*Figure 3.4.2.1: 10-fold cross-validation*

We split the training data into 10 folds and cross validate on them using scikit-learn as shown in the figure 3.4.2.1 above. The number of K-folds is arbitrary and usually set to 10 it is not a rule. In fact, determine the best K is still an unsolved problem but with lower K: computationally cheaper, less variance, more bias. With large K: computationally expensive, higher variance, lower bias.

We can now train the naive bayes classifier with the training set, validate it using the hold out part of data taken from the training set, the validation set, repeat this 10 times and average the results to get the final accuracy which is about **0.77** as shown in the screen results below,

```
Total tweets classified: 1183958
Score: 0.77653600187
Confusion matrix:
[[465021 126305]
 [136321 456311]]
```

*Figure 3.4.2.2: Result of the naive bayes classifier with the score representing the average of the results of each 10-fold cross-validation, and the overall confusion matrix.*

Notice that to evaluate our classifier we two methods, the F1 score and a confusion matrix. The **F1 Score** can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. It a measure of a **classifier's accuracy**. The F1 score is given by the following formula,

$$F1 = \frac{2 \times (\textit{precision} \times \textit{recall})}{(\textit{precision} + \textit{recall})}$$

*Formula 2.4.2.1: F1 score*

where the precision is the number of true positives (the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class,

$$\textit{Precision} = \frac{TP}{TP + FP}$$

*Formula 2.4.2.1: Precision*

and the recall is the number of true positives divided by the total number of elements that actually belong to the positive class,

$$Recall = \frac{TP}{TP + FN}$$

Formula 2.4.2.1: Recall

A precision score of 1.0 means that every result retrieved was relevant (but says nothing about whether all relevant elements were retrieved) whereas a recall score of 1.0 means that all relevant documents were retrieved (but says nothing about how many irrelevant documents were also retrieved). There is a **trade-off between precision and recall** where increasing one decrease the other and we usually use measures that combine precision and recall such as F-measure or MCC.

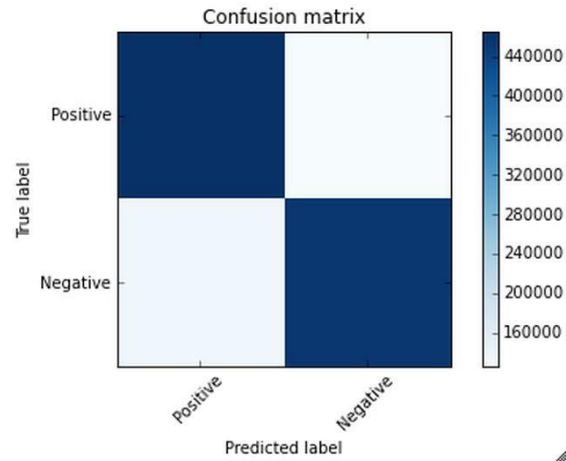
A **confusion matrix** helps to visualize how the model did during the classification and evaluate its accuracy. In our case we get about 156715 false positive tweets and 139132 false negative tweets. It is "about" because these numbers can vary depending on how we shuffle our data for example.

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

Figure 2.4.2.3: Example of confusion matrix

Notice that we still didn't use our test set, since we are going to tune our classifier for improving its results.

The confusion matrix of the naive bayes classifier can be expressed using a color map where dark colors represent high values and light colors represent lower values as shown in the corresponding color map of the naive bayes classifier below,



*Figure 3.4.2.4: Colormap of the confusion matrix related to the naive bayes classifier used.*

Hopefully we can distinguish that the number of true positive and true negative classified tweets is higher than the number of false and positive and negative tweets. However from this result we

try to improve the accuracy of the classifier by experimenting different techniques and we repeat the same process using the k-fold cross validation to evaluate its averaged accuracy.

### 3.4.2 Improvements

From the baseline, the goal is to improve the accuracy of the classifier, which is 0.77, in order to determine better which tweet is positive or negative. There are several ways of doing this and we present only few possible improvements (or not).

First we could try to removed what we called, stop words. Stop words usually refer to the most common words in the English language (in our case) such as: "the", "of", "to" and so on. They do not indicate any valuable information about the sentiment of a sentence and it can be necessary to remove them from the tweets in order to keep only words for which we are interested. To do this we use the list of 635 stopwords that we found. In the table below, you can see the most frequent words in the data set with their counts,

```
[('|target|', 780664),
 ('i', 778070),
 ('to', 614954),
 ('the', 538566),
 ('a', 383910),
 ('you', 341545),
 ('my', 336980),
 ('and', 316853),
 ('is', 236393),
 ('for', 236018),
 ('it', 235435),
 ('in', 217350),
 ('of', 192621),
 ('on', 169466),
 ('me', 163900),
 ('so', 158457),
 ('have', 150041),
 ('that', 146260),
 ('out', 143567),
 ('but', 132969)]
```

*Table 2.4.3.1: Most frequent words in the data set with their corresponding count.*

We can derive from the table, some interesting statistics like the number of times the tags used in the pre-processing step appear,

Recall that `||url||` corresponds to the URLs, `||target||` the twitter usernames with the symbol “@” before, `||not||` replaces the negation words, `||pos||` and `||neg||` replace the positive and negative smiley respectively. After removing the stop words we get the results below,

```
Total tweets classified: 1183958
Score: 0.758623708326
Confusion matrix:
[[437311 154015]
 [136343 456289]]
```

*Figure 2.4.2.2: Result of the naive bayes classifier with stopwords removed.*

Compared to the previous result, we lose 0.02 in accuracy and the number of false positive goes from 126305 to 154015 . We conclude that stop words seem to be useful for our classification task and remove them do not represent an improvement.

We could also try to stem the words in the data set. **Stemming** is the process by which endings are removed from words in order to remove things like tense or plurality. The stem form of a word could not exist in a dictionary (different from Lemmatization). This technique allows to unify words and reduce the dimensionality of the dataset. It's not appropriate for all cases but can make it easier to connect together tenses to see if you're covering the same subject matter. It is faster than **Lemmatization** (remove inflectional endings only and return the base or dictionary form of a word, which is known as the lemma). Using the library NLTK which is a library in Python specialized in natural language processing, we get the following results after stemming the words in the data set,

```
Total tweets classified: 1183958
Score: 0.773106857186
Confusion matrix:
[[462537 128789]
 [138039 454593]]
```

*Figure 2.4.2.2: Result of the naive bayes classifier after stemming.*

We actually lose 0.002 in accuracy score compared to the results of the baseline. We conclude that stemming words does not improve the classifier’s accuracy and actually do not make any sensible changes.

### 3.1 PROJECT DESIGN DIAGRAMS

**Fig. 1: An overview of the first approach**

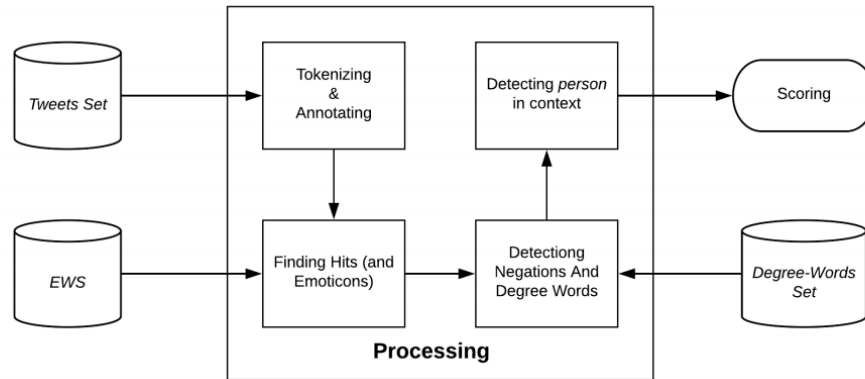
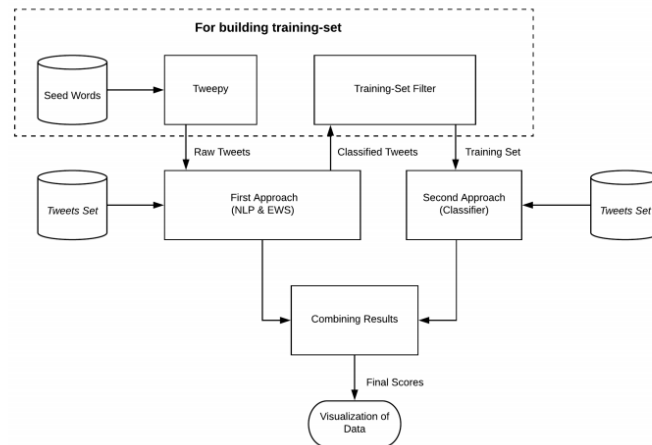


Fig. 1: An overview of the first approach

Our model consists of two completely different, yet interdependent approaches. The first approach uses Natural Language Processing, *Emotion-Words Set* and several textual features. It attempts to classify and score text according to the emotions present in it. The second approach uses standard classifiers like SMO and J48 to classify tweets.

**Fig. 2: Combining the first and the second approach**



The above flowchart showing the flow diagram of our project approach. Here two different yet interdependent approach of (NLP & EWS) and Classifier approach is being used whose flow is described in above flow chart.



Finally, we combine both these approaches to propose a Hybrid approach to detect emotions in text more effectively. Note that, even though we are extensively using the tweets example throughout this paper, this algorithm is very generic and can be used to detect and quantify emotions in any piece of text.

*After this, the Emotion-Category with the maximum final score is decided as the final EmotionCategory of the tweet (or the piece of text). An overview of the combined approach is shown in Fig 2.*

## CHAPTER-4

### Modules Description

#### **import re**

The Python 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.

We would cover two important functions, which would be used to handle regular expressions. But a small thing first: There are various characters, which would have special meaning when they are used in regular expression. To avoid any confusion while dealing with regular expressions, we would use Raw Strings as **r'expression'**.

#### **import pandas as pd**

pandas (all lowercase) is a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array. This makes pandas a trusted ally in data science and machine learning.

#### **import numpy as np**

It also helps to avoid namespace issues.

#### **import matplotlib.pyplot as plt**

matplotlib is a package, essentially a collection of related modules. At its simplest, a package can be just a directory containing the module files with an empty `__init__.py` file that tells python that the directory is to be treated as a package. A module B within package A (i.e. in the file structure A/B.py) is imported as `import A.B`

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures

#### **import seaborn as sns**

.Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and

statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

### **import string**

The string module contains a set of useful constants, such as `ascii_letters` and `digits`, and the module is often still imported for that reason.

### **import nltk**

NLTK is a set of libraries for Natural Language Processing. It is a platform for building Python programs to process natural language. NLTK is written in the Python programming language. It supports research and teaching in NLP or closely related areas, including cognitive science, empirical linguistics, information retrieval, artificial intelligence, and machine learning. NLTK provides an easy to use interface.

### **import warnings**

Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn't warrant raising an exception and terminating the program. For example, one might want to issue a warning when a program uses an obsolete module.

Python programmers issue warnings by calling the `warn()` function defined in this module. (C programmers use `PyErr_WarnEx()`; see Exception Handling for details).

## CHAPTER-5

### Results and Discussion

#### A. Testing Results

Similar to the training-set, we have created a testing-set of tweets by extracting tweets using some seed words and then using the first approach to filter and label these emotional tweets. This set consists of a total of 900 tweets, where each *Emotion-Category* has around 150 tweets, so as to maintain uniformity. Also, it is ensured that all tweets in the testing set are different from those in the training set. The two chosen classifiers, on testing with the testing-set, gave the results as shown in Table VI and VII. The correctly classified instances are the tweets for which the expected *Emotion Category* matches the actual *Emotion-Category*. As can clearly be seen from the tables, we have achieved a remarkable accuracy of 91.7% for SMO and 85.4% for J48, which proves the merits of our Emotion-Detection Algorithm.

TABLE VI: Accuracy of the SMO Classifier

SMO		
Correctly Classified Instances	826	91.7%
Incorrectly Classified Instances	74	8.22%
Total No. of Instances	900	

#### B. Surety Factor (Sf)

The surety factor indicates how confident and assertive our analysis and results are, on a given text/tweet. Its value is low, when there is a mismatch between the results of the two approaches or when the text seems to lack any emotions. On Fig. 2: Combining the first and the second approach

TABLE VII: Accuracy of the J48 Classifier

J48		
Correctly Classified Instances	769	85.4%
Incorrectly Classified Instances	131	14.5%
Total No. of Instances	900	

the other hand, its value is high when the two approaches concur with each other in results, there are too many *hits* or when one of the emotion-scores is very high. The surety factor is calculated on a scale of 6 and is dependent on several

factors:

- 

Classifier label match: Whether the classifier label matches the category of the maximum score of the first

approach. (Value: True/False).

- 

Max score: The value of the maximum of all the six scores.

- 

Max percent: The relative percentage of the Max score over all six scores.

- 

Second diff: The difference between the maximum and second maximum score value as a percent of the maximum.

- 

Hits: The number of *Emotion-words* matched in the text. Surety factor is calculated differently for the following two cases:

- If the *hits* in the tweet/text belong to the same *Emotion Category*, then only the factors Classifier label match and Max score are used.

- If the *hits* in the tweet/text belong to different *Emotion Categories*, all five factors are considered.

### ***C. Visualization of Results***

In order to demonstrate the usefulness of the proposed Emotion-Detection Algorithm, we have implemented the following applications:

- One-user Analysis (twitter account): We have developed an interface, which takes as input, the username of any twitter-user, processes all his/her tweets till date and displays a time-varying mood-swings plot as well as the relative and absolute emotional distribution in the form of pie charts. Fig. 3 is an example of the varying happiness of a twitter-user over time.

- Location Analysis: Using Google Maps API, a world map is displayed and each circle on it represents the emotional analysis of that particular region. The radius of each circle is proportional

to the area of the region. This analysis is done for around 20 major cities in India, using the *Location-Areas Set* (See Fig. 4)

- Document Analysis: Another interface takes as input, entire documents or blocks of text, processes it, and displays the relative and absolute emotional distribution of the document in the form of pie charts. (See Fig. 5 for an example sentence.)

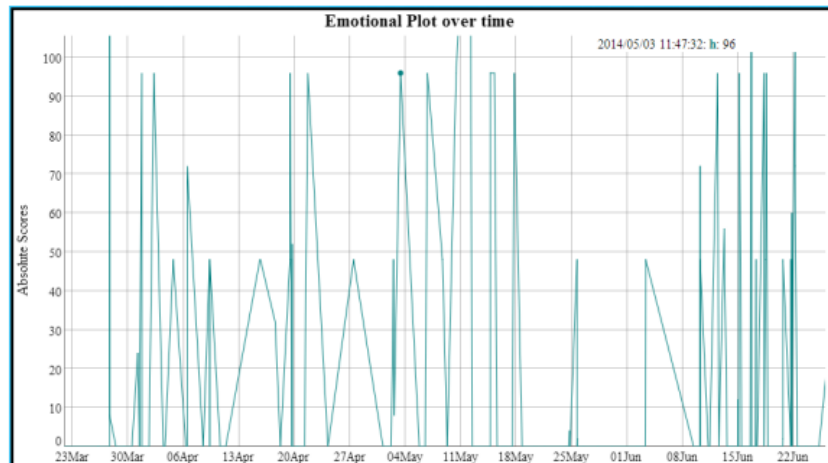


Fig. 3: Time varying happiness plot of a twitter-user

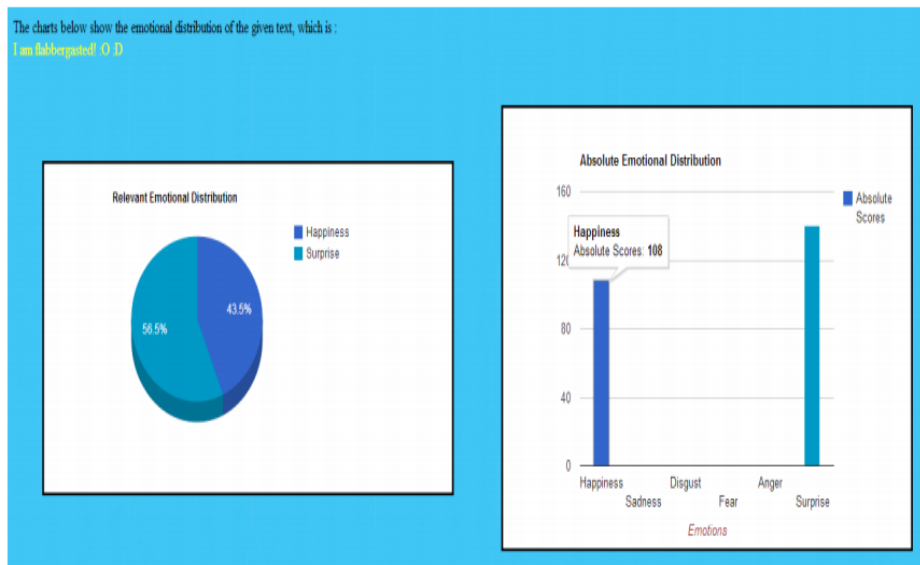


Fig. 5: Detecting emotions in a piece of text

## CHAPTER -5

### Conclusion and Future Scope

In this project, we have addressed the problem of classifying text into the six basic *Emotion-Categories*, rather than just labeling them as positive or negative. Through our research and a self-generated reliable bag of emotional words (*EWS*), we can now effectively quantify various emotions in any block of text. We have also automatically generated a labeled training-set (without manually labeling the tweets) of emotionally-biased tweets using a keyword-matching approach, which was then used to train various classifiers. Moreover, we have also introduced the concept of Surety Factor to suggest the reliability of our output and the degree of usefulness and correctness of our results. Finally, we visualized our results using pie-charts, bargraphs and maps, and demonstrated the various applications of our analysis. In future, a system could be established for automatically updating the bag-of-words which we created, on the basis of new tweets and data analysed. Using our approach, many interesting apps can be created, such as an add-on to a social-networking site displaying the recent mood of each of your friends. Also, our analysis of Twitter can be extended to the development of a real-time system, analyzing mood-swings and emotions on Twitter.

## REFERENCES

- [1] Y. Singh, P. K. Bhatia, and O.P. Sangwan," A Review of Studies on Machine Learning Techniques," International Journal of Computer Science and Security, Volume (1): Issue (1), pp. 70-84, 2007.
- [2] P.D. Turney," Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews," Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, pp. 417-424, July 2002.
- [3] Ch.L.Liu , W.H. Hsiao, C.H. Lee, and G.N.U., and E. Jou," Movie Rating and Review Summarization in Mobile Environment," IEEE Transactions on Systems, Man, and Cybernetics, Part C 42(3): pp.397-407, 2012.
- [4] R.Liu,R.Xiong , and L. Song," A Sentiment Classification Method for Chinese Document," Processed of the 5th International Conference on Computer Science and Education (ICCSE), pp. 918-922, 2010.
- [5] A.khan , Baharudin," Sentiment Classification Using Sentence-level Semantic Orientation of Opinion Terms from Blogs," Processed on National Postgraduate Conference (NPC), pp. 1 – 7, 2011.
- [6] L. Ramachandran, E.F.Gehringer, "Automated Assessment of Review Quality Using Latent Semantic Analysis," ICAIT, IEEE Computer Society, pp. 136-138, 2011.
- [7] B. Pang and L. Lee, Opinion mining and sentiment analysis, Foundations and Trends in Information Retrieval, vol. 2, pp. 8-10, 2008.
- [8] B. G. Malkiel, A Random Walk Down Wall Street: Including a Lifecycle Guide to Personal Investing. WW Norton & Company,1999.[3]
- [9] J. Bollen, H. Mao and X. Zeng, Twitter mood predicts the stock market, Journal of Computational Science, vol. 2, pp. 1-8, 2011.
- [10] M. A. Russell, Mining the Social Web: Data Mining Facebook  
Twitter, LinkedIn, Google+, GitHub, and More. O'Reilly Media, Inc,2013.



[11] L. Bing, K. C. Chan and C. Ou, Public sentiment analysis in twitter data for prediction of a company's stock price movements, in EBusiness Engineering (ICEBE), 2014 IEEE 11th International Conference on, 2014, pp. 232-239.

[12] M. Mittermayer, Forecasting intraday stock price trends with text mining techniques, in System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on, 2004, pp. 10-pp.

[13] R. Socher, A. Perlegen, J. Y. Wu, J. Chuang, C. D. Manning, A.

Y. Ng and C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2013, pp. 1642.Z .

[14] M. Sokolova and G. Lapalme, A systematic analysis of performance measures for classification tasks, Information Processing & Management, vol. 45, pp. 427-437, 2009.

[15] Giri, Kaiser J, and Towseef A Lone. (2014). "Big Data-Overview and Challenges." International Journal of Advanced Research in Computer Science and Software Engineering 4 (6).

[16] Sivarajah, Uthayasankar, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody. (2017) "Critical Analysis of Big Data Challenges and Analytical Methods." Journal of Business Research 70: 263-286.

[17] Agarwal, Basant, Namita Mittal, Pooja Bansal, and Sonal Garg. (2015) "Sentiment Analysis Using Common-Sense and Context Information." Journal of Computational Intelligence and Neuroscience 9 (2015).

[18] U. T. Gursoy, D. Bulut, and C. Yigit. (2017) "Social Media Mining and Sentiment Analysis for Brand Management." Global Journal of Emerging Trends in e-Business, Marketing and Consumer Psychology 3 (1): 497-551.

[19] Mäntylä, Mika V., Daniel Graziotin, and Miikka Kuutila. (2018) "The Evolution of Sentiment Analysis—A Review of Research Topics, Venues, and Top Cited Papers." Computer Science Review 27: 16-32.

[20] N, Mishra, and C. K. Jha. (2012) "Classification of Opinion Mining Techniques." International Journal of Computer Applications 56 (13).

[21] Song, Minchae, Hyunjung Park, and Kyung-shik Shin. (2019) "Attention-Based Long Short-Term Memory Network Using Sentiment Lexicon Embedding for Aspect-Level Sentiment Analysis in Korean." Information Processing & Management 56 (3): 637-653.

- [22] P. Sanguansat. (2016, 3-6 Feb. 2016) "Paragraph2Vec-Based Sentiment Analysis on Social Media for Business in Thailand", in the 2016 8th International Conference on Knowledge and Smart Technology (KST).
- [23] Itani, Maher, Chris Roast, and Samir Al-Khayatt. (2017) "Developing Resources for Sentiment Analysis of Informal Arabic Text in Social Media." *Procedia Computer Science* 117: 129-136
- [24] Chekima, Khalifa, and Rayner Alfred. (2018) *Sentiment Analysis of Malay Social Media Text*. pp. 205-219.
- [25] D. Cirqueira, M. Fontes Pinheiro, A. Jacob, F. Lobato, and Á. Santana. (2018, 3-6 Dec. 2018). "A Literature Review in Preprocessing for Sentiment Analysis for Brazilian Portuguese Social Media" in the 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI).
- [26] Peng, Haiyun, Erik Cambria, and Amir Hussain. (2017) "A Review of Sentiment Analysis Research in Chinese Language." *Cognitive Computation* 9 (4): 423-435



