PROJECT REPORT

**On**

**Twitter Sentimental Analysis**

**Submitted by**

**Aman Singh**

**171500037**

**Ayush Gupta**

**171500071**

**Submitted to**

**Mr. Juginder Pal Singh**

**ASSISTANT PROFESSOR**

Department of Computer Engineering & Applications

**Institute of Engineering & Technology**



**GLA University**

**Mathura- 281406, INDIA**

**MAY, 2021**

**Department of computer Engineering and Applications GLAUniversity,Mathura**

**17km. Stone NH#2, Mathura-Delhi Road, P.O.–Chaumuha, Mathura–281406**

**Declaration**

I hereby declare that the work which is being presented in the B.Tech. Project **“Twitter Sentimental Analysis”**, in partial fulfilment of the requirements for the award ofthe ***Bachelor of Technology*** in Computer Science and Engineering and submitted to the Department of Computer Engineering and Applications of GLA University, Mathura, is an authentic record of my own work carried under the supervision of **Mr. Juginder Pal Singh, Assistant professor.**

The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree.

Sign \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Name of Candidates:** AMAN SINGH (171500037)

**Course:** B.TECH (CS)

**Year:** Fourth Year

**Semester:** VIIIth

**Name of Candidates:** AYUSH GUPTA (171500071)

**Course:** B.TECH (CS)

**Year:** Fourth Year

**Semester:** VIIIth

**Certificate**

This is to certify that the above statements made by the candidate are correct to the best of our knowledge and belief.

**Supervisor**

Mr. Juginder Pal Singh

Assistant Professor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Project Coordinator** **Program Co-ordinator**

(Mr. Mayank Srivastava) (Dr. Anant Ram)

Date: 03 June 2021

**ACKNOWLEDGEMENT**

It is a matter of great pleasure and privilege for us to present this report of project. Through this report, we would like to thank numerous people whose consistent support and guidance has been the standing pillar in architecture of this report.

To begin with, our sincere thanks to **Mr. Neeraj Agarwal .** I express thanks to **Mr.Anand Jalal (HOD**

**– CSE)** who gave encouragement and valuable suggestions throughout the training.

I would like to express my sincere thanks to **Mr. Juginder Pal Singh**, for guiding me during the completion of the project.

**ABSTRACT**

*This project addresses the problem of sentiment analysis in twitter; that is classifying tweets according to the sentiment expressed in them: positive, negative or neutral. Twitter is an online micro-blogging and social-networking platform which allows users to write short status updates of maximum length 140 characters. It is a rapidly expanding service with over 200 million registered users out of which 100 million are active users and half of them log on twitter on a daily basis - generating nearly 250 million tweets per day. Due to this large amount of usage we hope to achieve a reflection of public sentiment by analysing the sentiments expressed in the tweets. Analysing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like stock exchange. The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream.*

**Table of Contents**

Declaration ii

Certificate iii

Acknowledgement iv

Abstract v

List of Figures vii

List of Tables viii

1. Introduction 1
   1. Motivation 1
   2. Domain Introduction 1
   3. Idea 3
2. Literature Survey 4
   1. Twitter Sentiment Analysis: The Good the Bad and the OMG! 4
   2. Sentiment Analysis in Twitter using Machine Learning Techniques 5
   3. Twitter Sentiment Analysis 7
   4. SENTIMENT ANALYSIS ON TWITTER DATA 8
3. Proposed work 11
4. Implementation and Results 25
5. Conclusion 40
6. References 41

**List of Figures**

1. Proposed Framework 11
2. Histogram of the tweets according to their sentiment 14
3. A tweet before processing HTML entities. 19
4. A tweet after processing HTML entities. 19
5. Top 20 of acronyms in the data set of tweets 22
6. Top 20 of acronyms in the data set of tweets with their translation and count 23
7. A tweet before processing negation words. 23
8. A tweet after processing negation words. 23
9. All Formulas 25
10. 10­fold cross­validation 29
11. Result of the naive bayes classifier with the score representing the average

of the results of each 10­fold cross­validation, and the overall confusion matrix. 30

1. Example of confusion matrix 31
2. Colormap of the confusion matrix related to the naive bayes classifier used. 32
3. Most frequent words in the data set with their corresponding count. 33
4. Tags in the data set with their corresponding count 34
5. Result of the naive bayes classifier with stopwords removed. 34
6. Result of the naive bayes classifier after stemming. 35
7. Results of the naive bayes classifier with bigram features. 39
8. Results of the naive bayes classifier with unigram and bigram features. 40
9. shows the accuracy of Logistic Regression 40
10. shows the accuracy of Multinomial Naïve Bayes Theorem 40
11. shows the accuracy of Stochastic Gradient Descent Algorithm 40
12. shows the accuracy of Voting Classifier 41

**List of Table**

1. Example of twitter posts annotated with their corresponding sentiment,

0 if it is negative, 1 if it is positive. 13

1. Before processing emoticons, list of tweets where some of them contain emoticons. 17
2. After processing emoticons, they have been replaced by their corresponding tag 17
3. Tweets before processing URLs. 18
4. Tweets after processing URLs. 18
5. Tweets before processing Unicode. 19
6. Tweets after processing Unicode. 19
7. Tweets before processing lowercase. 20
8. Tweets after processing lowercase. 20
9. Tweets before processing targets. 21
10. Tweets after processing targets. 21
11. Tweets before processing sequences of repeated characters. 24
12. Tweets after processing sequences of repeated character 24

**Chapter 1 Introduction**

**1.1 Motivation**

We have chosen to work with twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover the response on twitter is more prompt and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could be done by analysing overall public sentiment towards that firm with respect to time and using economics tools for finding the correlation between public sentiment and the firm’s stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market is it having a favorable response and in which a negative response (since twitter allows us to download stream of geo-tagged tweets for particular locations. If firms can get this information they can analyze the reasons behind geographically differentiated response, and so they can market their product in a more optimized manner by looking for appropriate solutions like creating suitable market segments. Predicting the results of popular political elections and polls is also an emerging application to sentiment analysis. One such study was conducted by Tumasjan et al. in Germany for predicting the outcome of federal elections in which concluded that twitter is a good reflection of offline sentiment.

**1.2 Domain Introduction**

This project of analyzing sentiments of tweets comes under the domain of “Pattern Classification” and “Data Mining”. Both of these terms are very closely related and intertwined, and they can be formally defined as the process of discovering “useful” patterns in large set of data, either automatically (unsupervised) or semiautomatically (supervised). The project would heavily rely on techniques of “Natural Language Processing” in extracting significant patterns and features from the large data set of tweets and on “Machine Learning” techniques for accurately classifying individual unlabelled data samples (tweets) according to whichever pattern model best describes them. The features that can be used for modeling patterns and classification can be divided into two main groups: formal language based and informal blogging based. Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example the word “excellent” has a strong positive connotation while the word “evil” possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment. Parts of Speech tagging, on the other hand, is a syntactical approach to the problem. It means to automatically identify which part of speech each individual word of a sentence belongs to: noun, pronoun, adverb, adjective, verb, interjection, etc. Patterns can be extracted from analyzing the frequency distribution of these parts of speech (ether individually or collectively with some other part of speech) in a particular class of labeled tweets. Twitter based features are more informal and relate with how people express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, word lengthening, question marks, presence of url in tweets, exclamation marks, internet emoticons and internet shorthand/slangs. Classification techniques can also be divided into a two categories: Supervised vs. unsupervised and non-adaptive vs. adaptive/reinforcement techniques. Supervised approach is when we have pre-labeled data samples available and we use them to train our classifier. Training the classifier means to use the pre-labeled to extract features that best model the patterns and differences between each of the individual classes, and then classifying an unlabeled data sample according to whichever pattern best describes it. For example if we come up with a highly simplified model that neutral tweets contain 0.3 exclamation marks per tweet on average while sentiment-bearing tweets contain 0.8, and if the tweet we have to classify does contain 1 exclamation mark then (ignoring all other possible features) the tweet would be classified as subjective, since 1 exclamation mark is closer to the model of 0.8 exclamation marks. Unsupervised classification is when we do not have any labeled data for training. In addition to this adaptive classification techniques deal with feedback from the environment. In our case feedback from the environment can be in form of a human telling the classifier whether it has done a good or poor job in classifying a particular tweet and the classifier needs to learn from this feedback. There are two further types of adaptive techniques: Passive and active. Passive techniques are the ones which use the feedback only to learn about the environment (in this case this could mean improving our models for tweets belonging to each of the three classes) but not using this improved learning in our current classification algorithm, while the active approach continuously keeps changing its classification algorithm according to what it learns at real-time.

**1.3 Idea**

This project was motivated by my desire to investigate the sentiment analysis field of machine learning since it allows to approach natural language processing which is a very hot topic actually. Following my previous experience where it was about classifying short music according to their emotion, I applied the same idea with tweets and try to figure out which is positive or negative.

**Chapter 2**

**Literature Survey**

# Sentiment analysis, also refers as opinion mining, is a sub machine learning task where we want to determine which the general sentiment of a given document is. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive, neutral or negative. It is a really useful analysis since we could possibly determine the overall opinion about a selling objects, or predict stock markets for a given company like, if most people think positive about it, possibly its stock markets will increase, and so on. Sentiment analysis is actually far from to be solved since the language is very complex (objectivity/subjectivity, negation, vocabulary, grammar) but it is also why it is very interesting to working on.

# In this project I choose to try to classify tweets from Twitter into “positive” or “negative” sentiment by building a model based on probabilities. Twitter is a microblogging website where people can share their feelings quickly and spontaneously by sending a tweets limited by 140 characters. You can directly address a tweet to someone by adding the target sign “@” or participate to a topic by adding a hashtag “#” to your tweet. Because of the usage of Twitter, it is a perfect source of data to determine the current overall opinion about anything.

**2.1 Twitter Sentiment Analysis: The Good the Bad and the OMG!**

**Abstract**

In this paper, we investigate the utility of linguistic features for detecting the sentiment of Twitter messages. We evaluate the usefulness of existing lexical resources as well as features that capture information about the informal and creative language used in microblogging. We take a supervised approach to the problem, but leverage existing hashtags in the for building training data.

**Introduction**

In the past few years, there has been a huge growth in the use of microblogging platforms such as Twitter. Spurred by that growth, companies and media organizations are increasingly seeking ways to mine Twitter for information about what people think and feel about their products and services. Companies such as Twitratr (twitrratr.com), tweetfeel (www.tweetfeel.com), and Social Mention (www.socialmention.com) are just a few who advertise Twitter sentiment analysis as one of their services.

While there has been a fair amount of research on how sentiments are expressed in genres such as online reviews and news articles, how sentiments are expressed given the informal language and message-length constraints of microblogging has been much less studied. Features such as automatic part-of-speech tags and resources such as sentiment lexicons have proved useful for sentiment analysis in other domains, but will they also prove useful for sentiment analysis in Twitter? In this paper, we begin to investigate this question.

**Conclusion**

Our experiments on twitter sentiment analysis show that part-of-speech features may not be useful for sentiment analysis in the microblogging domain. More research is needed to determine whether the POS features are just of poor quality due to the results of the tagger or whether POS features are just less useful for sentiment analysis in this domain. Features from an existing sentiment lexicon were somewhat useful in conjunction with microblogging features, but the microblogging features (i.e., the presence of intensifiers and positive/negative/neutral emoticons and abbreviations) were clearly the most useful.

**2.2 Sentiment Analysis in Twitter using Machine Learning Techniques**

**Abstract**

Sentiment analysis deals with identifying and classifying opinions or sentiments expressed in source text. Social media is generating a vast amount of sentiment rich data in the form of tweets, status updates, blog posts etc. Sentiment analysis of this user generated data is very useful in knowing the opinion of the crowd. Twitter sentiment analysis is difficult compared to general sentiment analysis due to the presence of slang words and misspellings. The maximum limit of characters that are allowed in Twitter is 140. Knowledge base approach and Machine learning approach are the two strategies used for analysing sentiments from the text. In this paper, we try to analyze the twitter posts about electronic products like mobiles, laptops etc using Machine Learning approach. By doing sentiment analysis in a specific domain, it is possible to identify the effect of domain information in sentiment classification. We present a new feature vector for classifying the tweets as positive, negative and extract peoples’ opinion about products.

**Proposed Solution**

A dataset is created using twitter posts of electronic products. Tweets are short messages with full of slang words and misspellings. So we perform a sentence level sentiment analysis. This is done in three phases. In first phase preprocessing is done. Then a feature vector is created using relevant features. Finally using different classifiers, tweets are classified into positive and negative classes. Based on the number of tweets in each class, the final sentiment is derived.

**Conclusion**

There are different Symbolic and Machine Learning techniques to identify sentiments from text. Machine Learning techniques are simpler and efficient than Symbolic techniques. These techniques can be applied for twitter sentiment analysis. There are certain issues while dealing with identifying emotional keyword from tweets having multiple keywords. It is also difficult to handle misspellings and slang words. To deal with these issues, an efficient feature vector is created by doing feature extraction in two steps after proper preprocessing. In the first step, twitter specific features are extracted and added to the feature vector. After that, these features are removed from tweets and again feature extraction is done as if it is done on normal text. These features are also added to the feature vector. Classification accuracy of the feature vector is tested using different classifiers like Nave Bayes, SVM, Maximum Entropy and Ensemble classifiers. All these classifiers have almost similar accuracy for the new feature vector. This feature vector performs well for electronic products domain.

**2.3 Twitter Sentiment Analysis**

**Abstract**

Social media have received more attention nowadays. Public and private opinion about a wide variety of subjects are expressed and spread continually via numerous social media. Twitter is one of the social media that is gaining popularity. Twitter offers organizations a fast and effective way to analyse customers’ perspectives toward the critical to success in the market place. Developing a program for sentiment analysis is an approach to be used to computationally measure customers’ perceptions. This paper reports on the design of a sentiment analysis, extracting a vast amount of tweets. Prototyping is used in this development. Results classify customers’ perspective via tweets into positive and negative, which is represented in a pie chart and html page. However, the program has planned to develop on a web application system, but due to limitation of Django which can be worked on a Linux server or LAMP, for further this approach need to be done.

**Result**

To associate with Twitter API, developer need to agree in terms and conditions of development Twitter platform which has been provided to get an authorization to access a data. The output from this process will be saved in JSON file. The reason is, JSON (JavaScript Object Notation) is a lightweight data-interchange format which is easy for humans to write and read. Moreover, stated that, JSON is simple for machines to generate and parse. JSON is a text format that is totally language independent, but uses a convention that is known to programmers of the C-family of languages, including Python and many others. However, output\_s size depends on the time for retrieving tweets from Twitter.

Nevertheless, the output will be categorized into 2 forms, which are encoded and un-encoded. According to security issue for accessing a data, some of the output will be shown in an ID form such as string ID. Sentiment Analysis. The tweets will be assigned the value of each word, together with categorize into positive and negative word, according to lexicon dictionary. The result will be shown in .txt, .csv and html.

**Conclusion**

Twitter sentiment analysis is developed to analyse customers\_ perspectives toward the critical to success in the marketplace. The program is using a machine-based learning approach which is more accurate for analyzing a sentiment; together with natural language processing techniques will be used.

As a result, program will be categorized sentiment into positive and negative, which is represented in a pie chart and html page Although, the program has been planned to be developed as a web application, due to limitation of Django which can only work on Linux server or LAMP. Thus, it cannot be realized. Therefore, further enhancement of this element is recommended in future study.

**2.4 SENTIMENT ANALYSIS ON TWITTER DATA**

**Abstract**

Sentiment analysis is a type of natural language processing for tracking the mood of the public about a particular product or topic. Sentiment analysis, which is also called opinion mining, involves in building a system to collect and examine opinions about the product made in blog posts, comments, reviews or tweets. Sentiment analysis can be useful in several ways. In fact, it has spread from computer science to management sciences and social sciences due to its importance to business and society as a whole. In recent years, industrial activities surrounding sentiment analysis have also thrived. Numerous startups have emerged. Many large corporations have built their own in-house capabilities. Sentiment analysis systems have found their applications in almost every business and social domain. The goal of this report is to give an introduction to this fascinating problem and to present a framework which will perform sentiment analysis on online mobile phone reviews by associating modified K means algorithm with Naïve bayes classification and KNN.

**Proposed Methodology**

The proposed architecture of four modules: user interface, log pre-processing, Feature Clustering using Modified K-means, Naïve Bayes Classification, Training and testing using KNN for more accurate categorization of opinion. This system can solve irrelevant data and more accuracy by associating Modified K means with Naïve Bayes Classification algorithm.

**Conclusion**

Above methods has been applied on mobile review .We proposed a method using Naïve Bayes, KNN and modified k means clustering and found that it is more accurate than Naïve Bayes and KNN techniques individually. We obtained an overall classification accuracy of 91% on the test set of 500 mobile reviews. The running time of our algorithm is O (n + V log V) for training and O (n) for testing, where n is the number of words in the documents (linear) and V the size of the reduced vocabulary. It is much faster than other machine learning algorithms like Naïve Bayes classification or Support Vector Machines which take a long time to converge to the optimal set of weights. The accuracy is comparable to that of the current state-of-the-art algorithms used for sentiment classification on mobile reviews.

**2.5 Sentiment Analysis of Twitter Feeds**

**Abstract**

This paper focuses on classifying tweets based on the sentiments expressed in them, with the aim to classify them into three categories: positive, negative and neutral. In particular, we investigate the relevance of using a two-step classifier and negation detection in the space of Twitter Sentiment analysis. An efficient sentiment analyzer is deemed to be a must in the era of big data where preponderance of electronic communication is a major bottleneck. Major difficulties in handling of tweets are, their limited size, and the cryptic style of writing that makes them difficult to comprehend at times. We have used different datasets publicly available online and designed a comprehensive set of pre-processing steps that make the tweets more amenable to Natural Language Processing techniques. Two classifiers are designed based on Naive-Bayes and Maximum Entropy classifiers, and their accuracies are compared on different feature sets. We feel that such classifiers will help business or corporate houses, political parties or analysts etc. to evaluate public sentiments about them and design appropriate policies to address their concerns.

**Methodology**

We use different feature sets and machine learning classifiers to determine the best combination for sentiment analysis of twitter. We also experiment with various pre-processing steps like – punctuations, emoticons, twitter specific terms and stemming. We investigated the following features – unigrams, bigrams, trigrams and negation detection. We finally train our classifier using various machine-learning algorithms – Naive Bayes and Maximum Entropy. We use a modularized approach with feature extractor and classification algorithm as two independent components. This enables us to experiment with different options for each component. Different steps taken in the entire process are illustrated.

**Conclusion**

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

Our baseline classifier that uses just the unigrams achieves an accuracy of around 73.00%. Accuracy of the classifier increases if we use negation detection or introduce bigrams and trigrams. Thus we can conclude that both Negation Detection and higher order n-grams are useful for the purpose of text classification. Moreover, if we use both n-grams and negation detection, the accuracy is higher than both of them individually. We also note that Single step classifiers outperform double step classifiers. Maximum Entropy Classifier performs better than Naive Bayes Classifier in case of unigrams and negation detection, but if we use higher n-grams also, Naive Bayes Classifier is seen to perform better than Maximum Entropy Classifier. We achieve the best accuracy of 75.33% in the case of Unigrams + Bigrams + Trigrams + Negation, trained on Naive Bayes Classifier.

The amount of data available on twitter that we can use for classification purposes is huge. It needs to be seen how our method works on big data. Appropriate distributed computing algorithms need to be developed for the implementation of the algorithms. This is one aspect that we are examining currently. Apart from this, following is a list of ideas we would like to explore in the future.

**Chapter 3**

**Proposed Work**

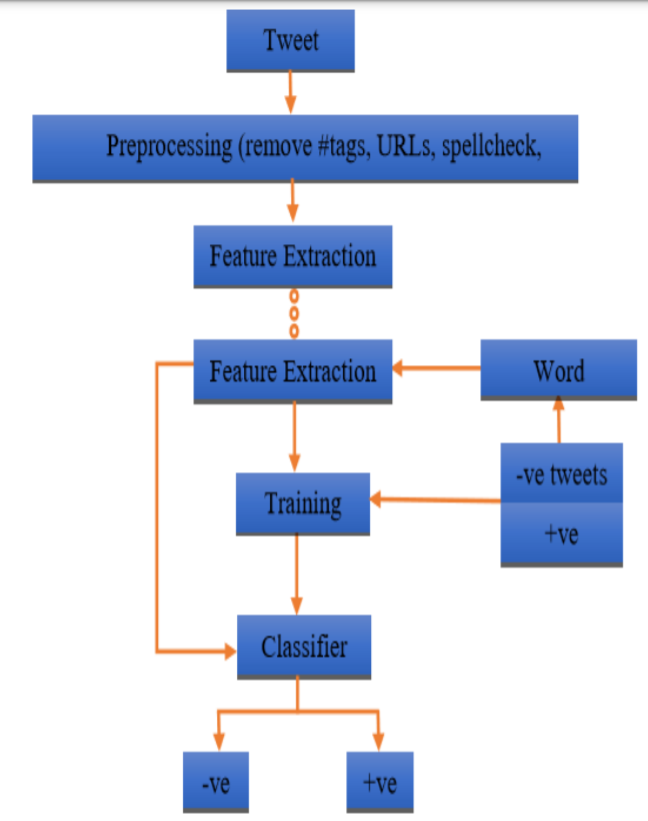


Fig. 1 Proposed Framework

Sentiment analysis, also known as opinion mining, is a type of machine learning activity in which we try to figure out what a document's overall sentiment is. We may extract subjective information from a text and try to classify it according to its polarity, such as positive, neutral, or negative, using machine learning techniques and natural language processing. It is a very useful study because it allows us to decide the overall opinion about a selling object or forecast stock markets for a given business, for example, if the majority of people think it is a good idea, its stock markets will rise, and so on. Sentiment analysis is far from being solved due to the complexity of the language (objectivity/subjectivity, negation, vocabulary, grammar), but this is also why it is so fascinating to work on.

We built a model based on probabilities to classify tweets from Twitter into "positive" or "poor" sentiment. Twitter is a microblogging platform where users can express themselves easily and randomly by sending 140-character tweets. To classify tweets from Twitter into "positive" or "bad" sentiment, we developed a model based on probabilities. Twitter is a microblogging website that allows users to quickly and spontaneously express themselves by sending 140-character tweets.

**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 my 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.

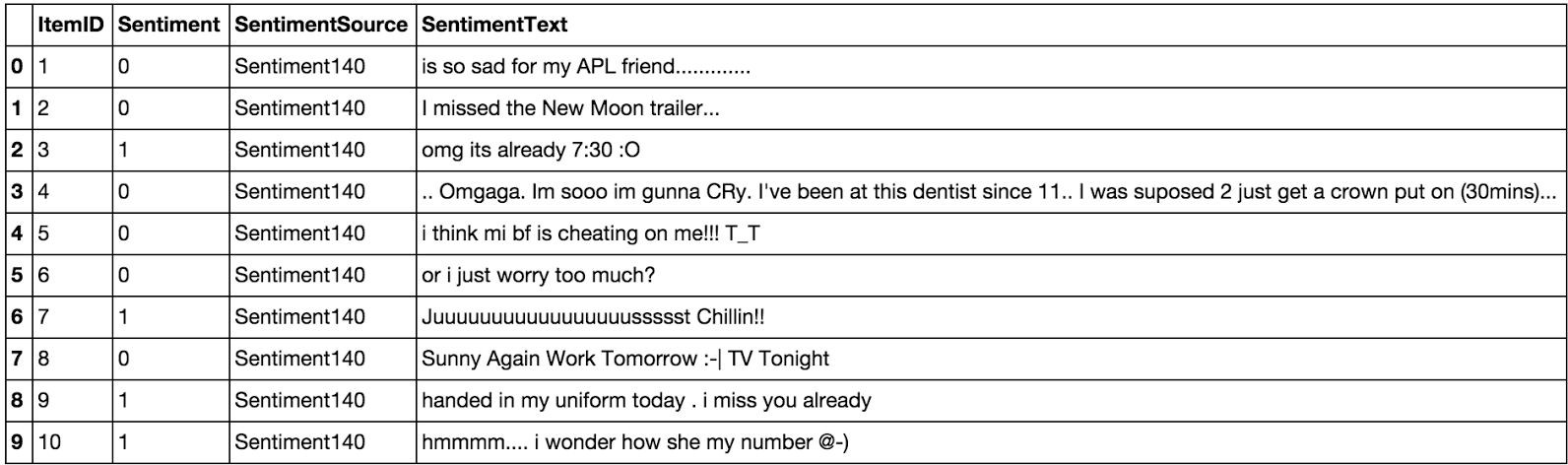


Table 1 Example of twitter posts annotated with their corresponding sentiment, 0 if it is negative, 1 if it is positive.

In the Table 1showing 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 means 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 "Juuuuuuuuuuuuuuuuussssst", "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 peope

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,

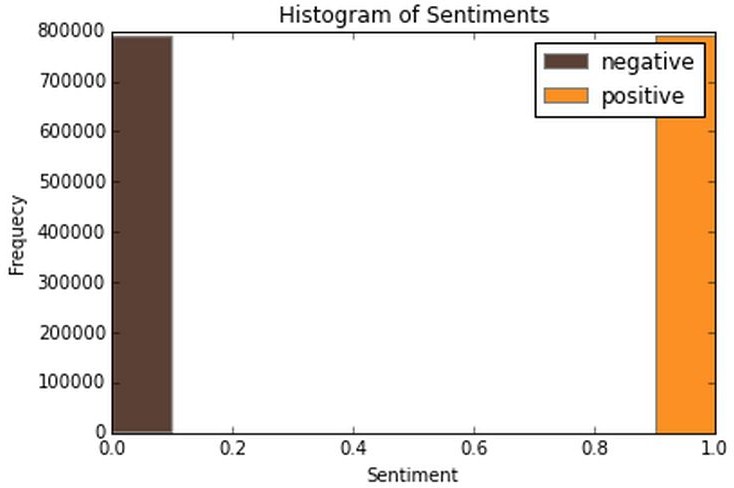


Fig. 2 Histogram of the tweets according to their sentiment

We have exactly 790177 positive tweets and 788435 negative tweets which signify that the dataset is well­balanced. 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 pre­processing step.

## ● Resources:

In order to facilitate the pre­processing 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 out­of­context) 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 scikit­learn ([http://scikit­learn.org/](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 well­explained. With Python, the libraries, Numpy (<http://www.numpy.org/>) and Panda (<http://pandas.pydata.org/>) for manipulating data easily and intuitively are just essential.

## ● Pre-processing

Now that we have the corpus of tweets and all the resources that could be useful, we can pre­process 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 pre­processing includes cleaning, normalization, transformation, feature extraction and selection, etc. The result of pre­processing will be consistent and uniform data that are workable to maximize the classifier's performance.

All of the tweets are pre­processed by passing through the following steps in the same

order.

### ● 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.

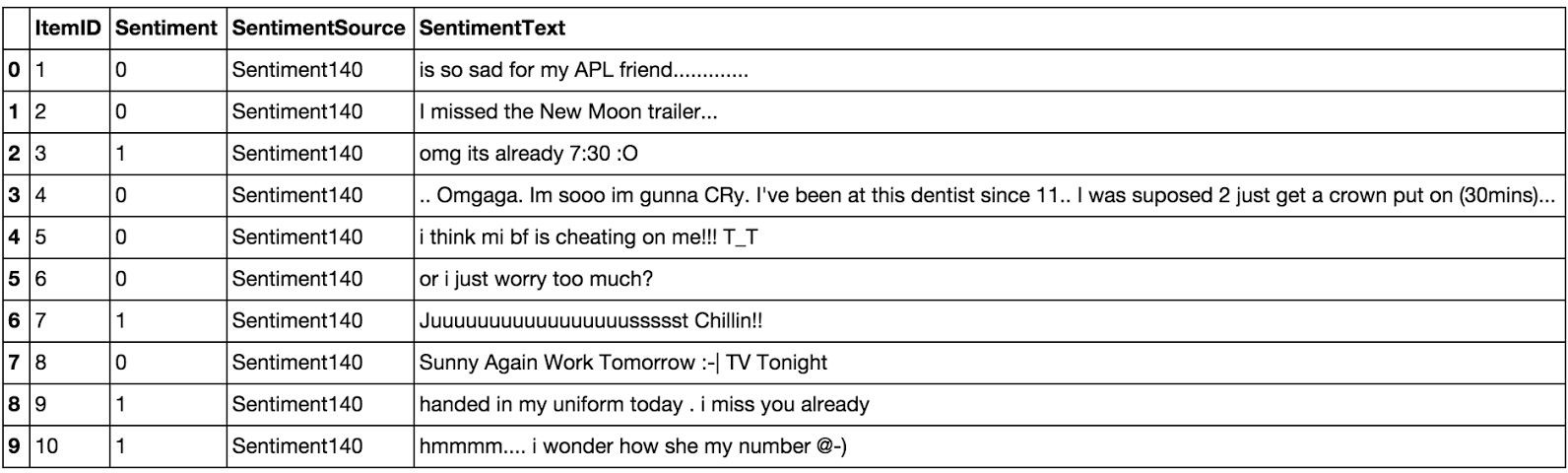


Table 2 efore processing emoticons, list of tweets where some of them contain emoticons.

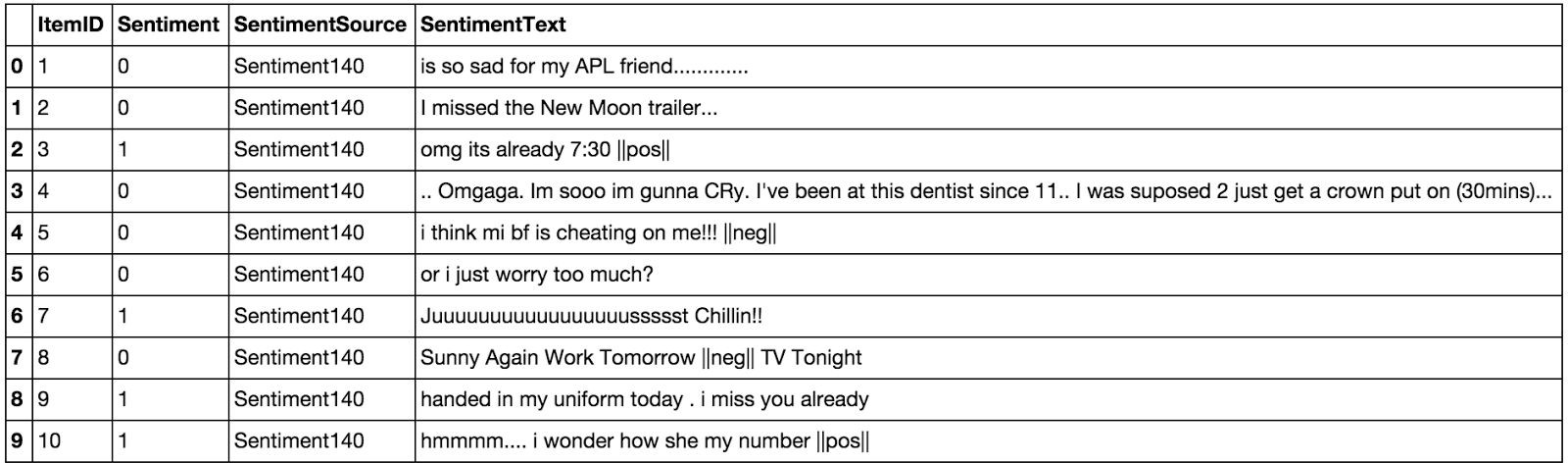


Table 3 After processing emoticons, they have been replaced by their corresponding tag

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

### ● 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.

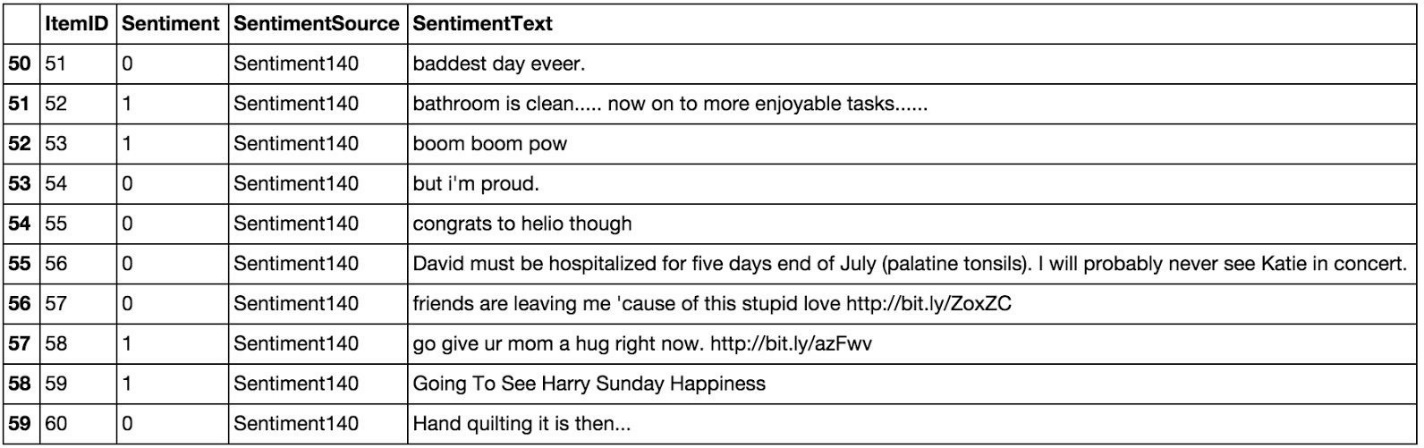


Table 4 Tweets before processing URLs.

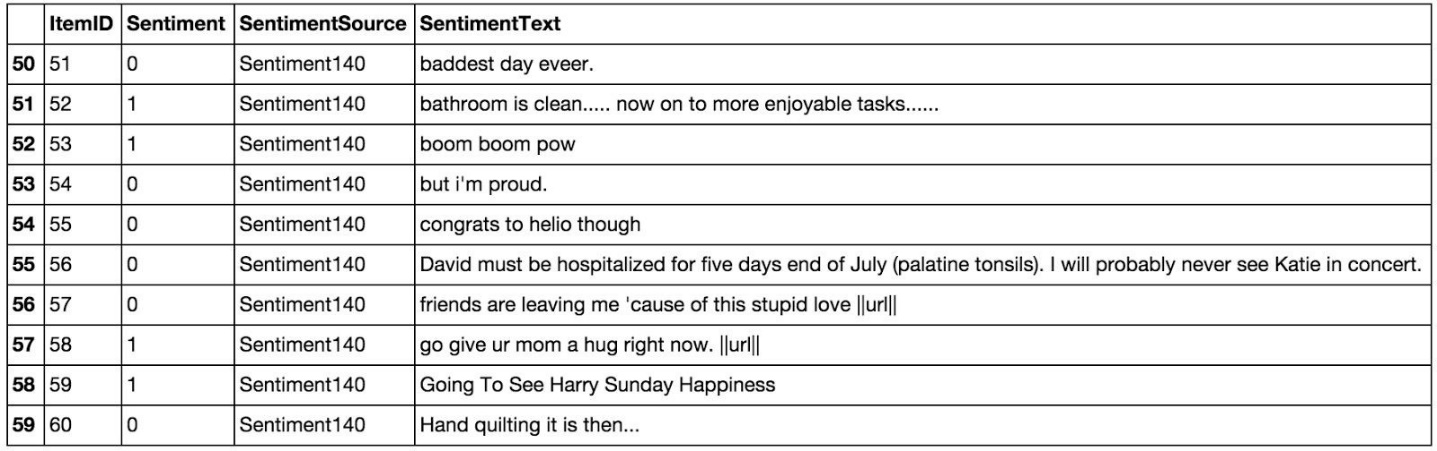


Table 5 Tweets after processing URLs.

### ● 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.

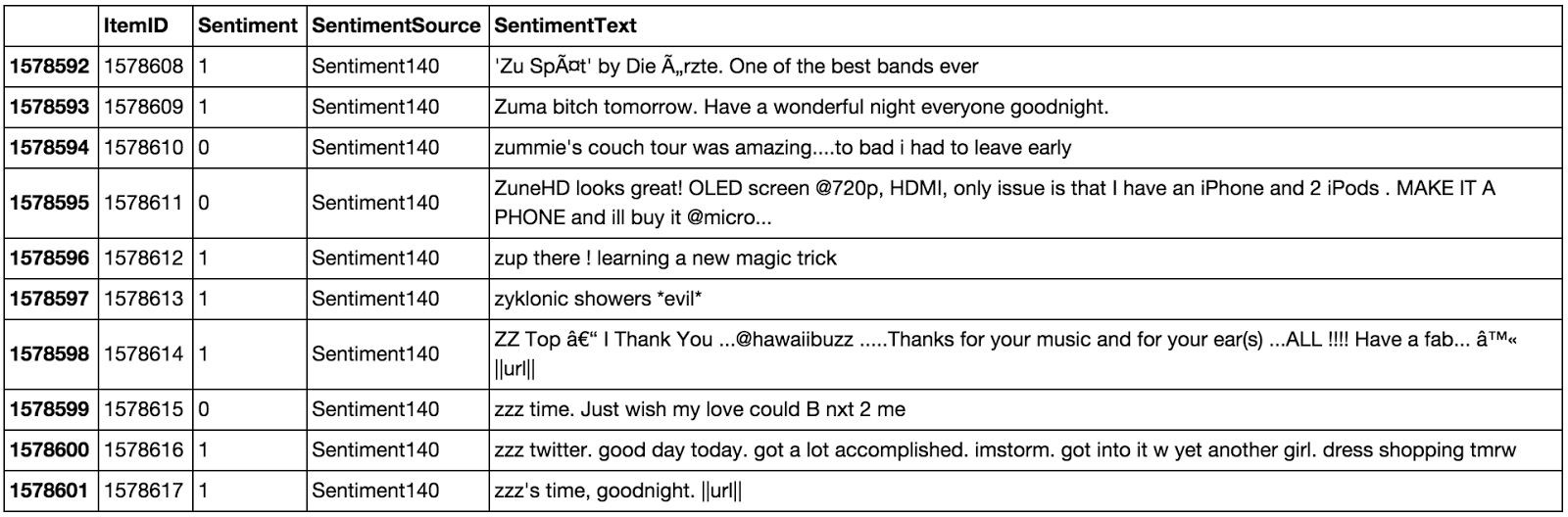


Table 6 Tweets before processing Unicode.

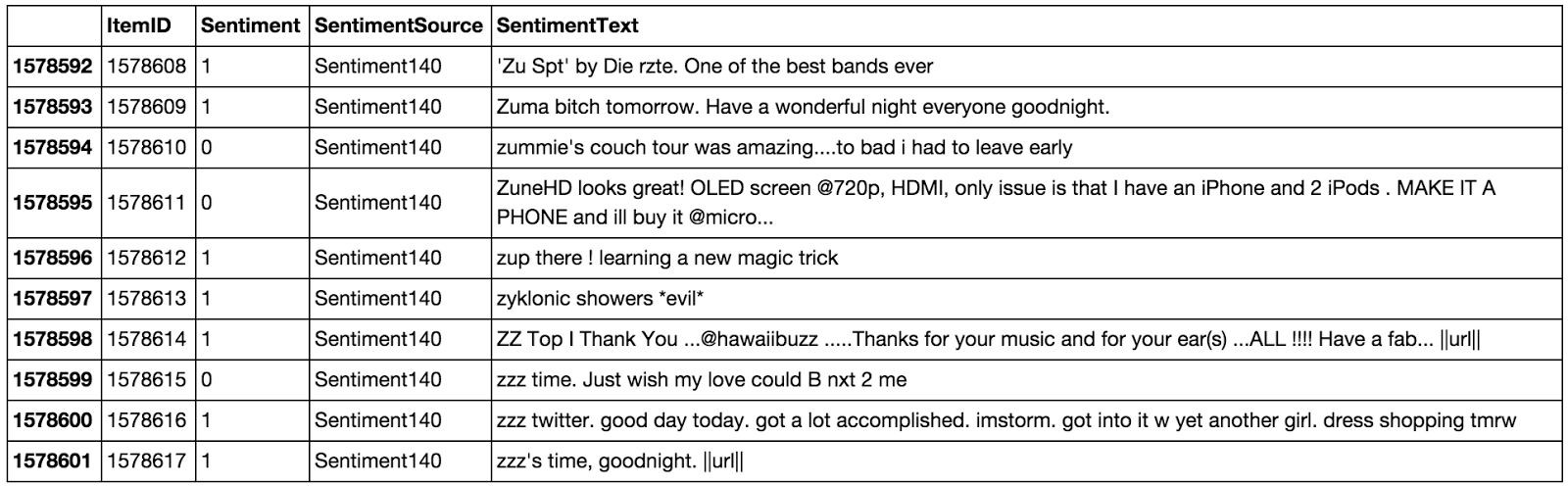


Table 7 Tweets after processing Unicode.

### ● HTML entities:

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



Fig. 3 A tweet before processing HTML entities.



Fig. 4 A tweet after processing HTML entities.

### 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.

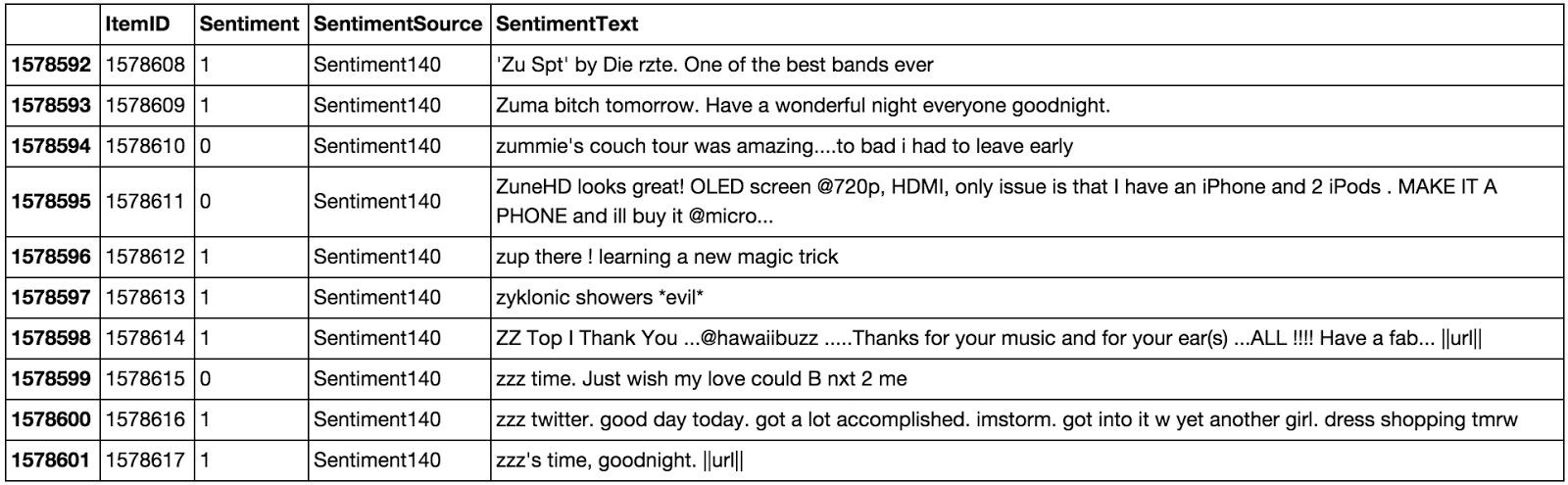


Table 8 Tweets before processing lowercase.

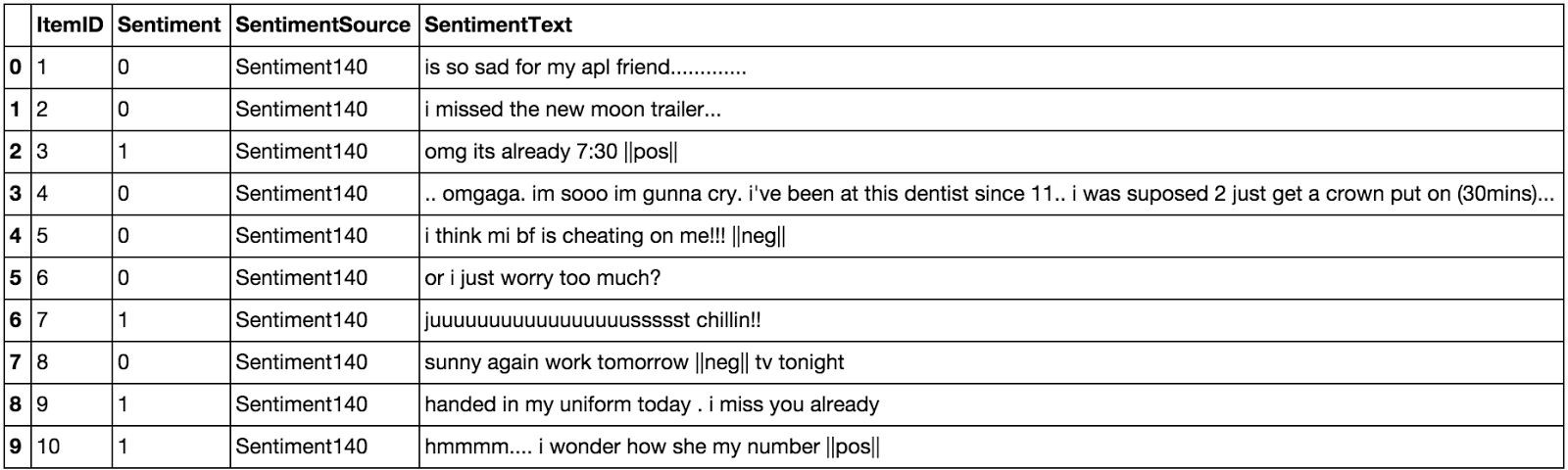


Table 9 Tweets after processing lowercase.

### ● 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.

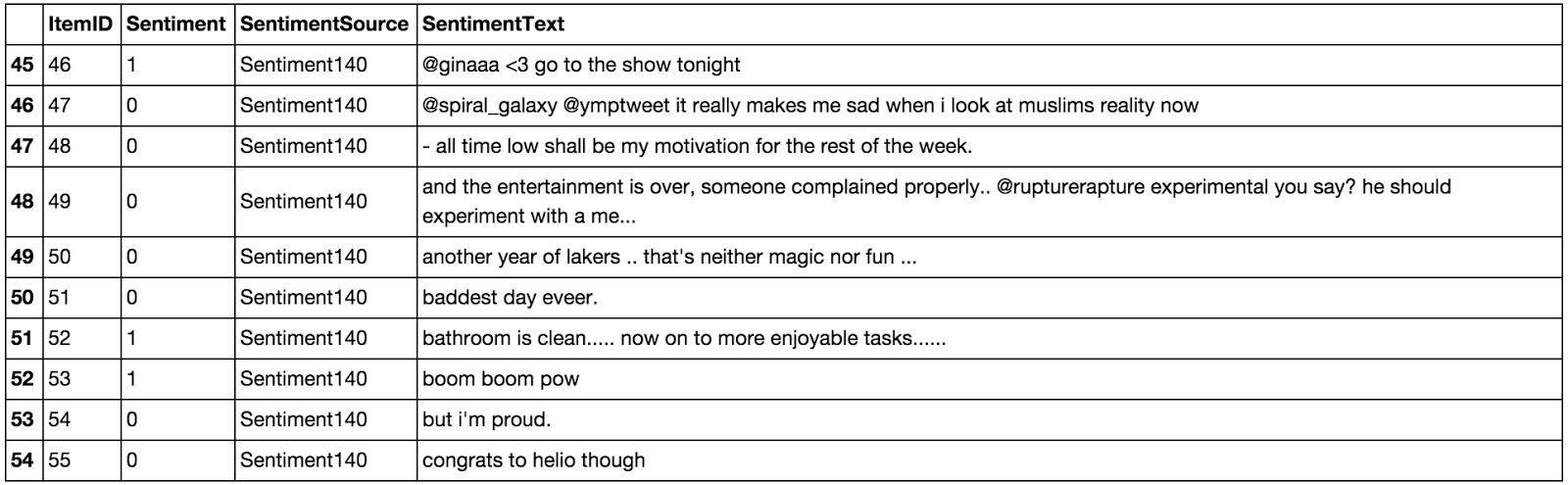


Table 10 Tweets before processing targets.

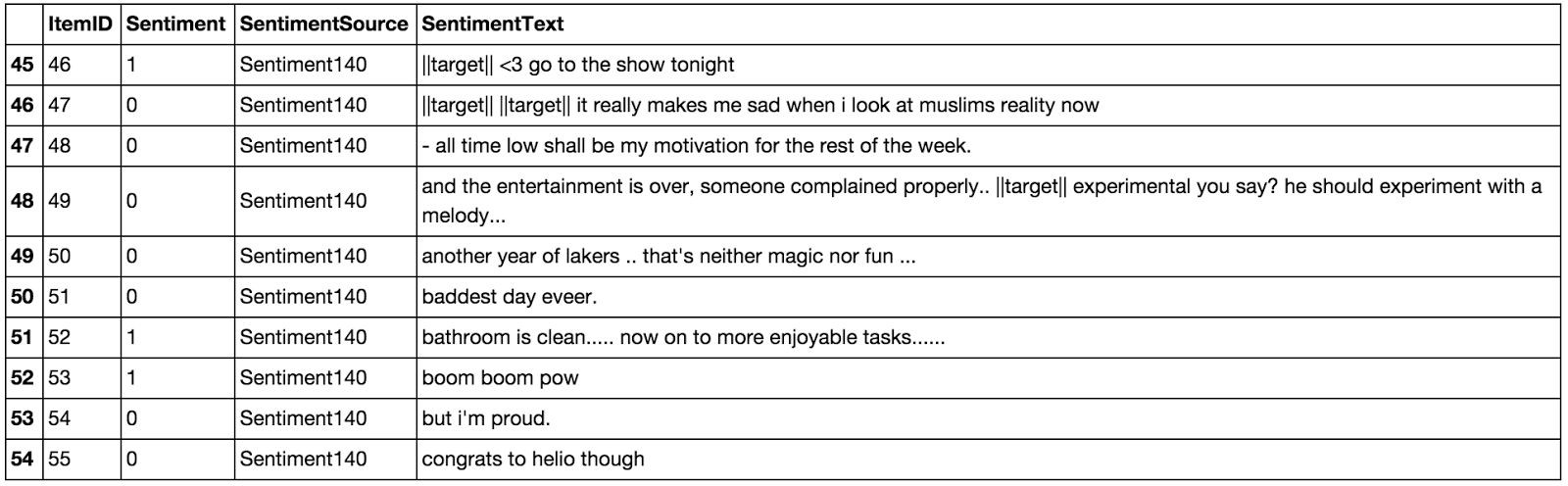


Table 11 Tweets after processing targets.

### ● 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 nltk.tokenizer but it is definitely much much slower (also much more accurate).

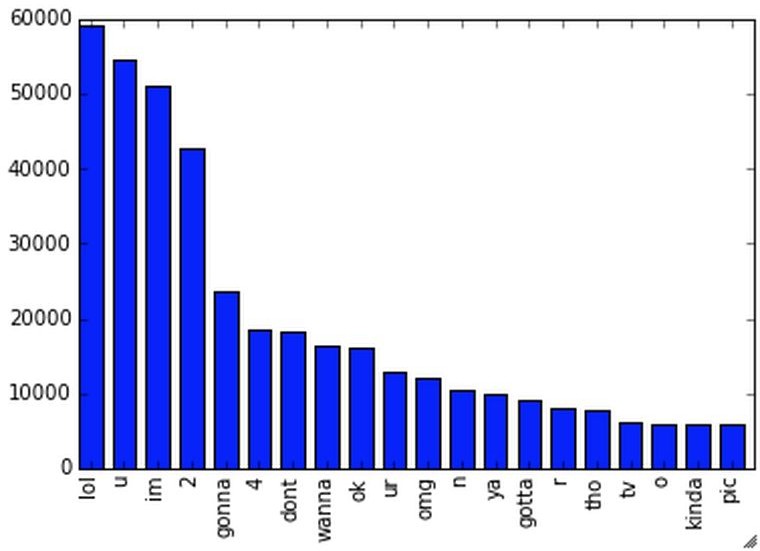


Fig. 5 Top 20 of acronyms in the data set of tweets

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.

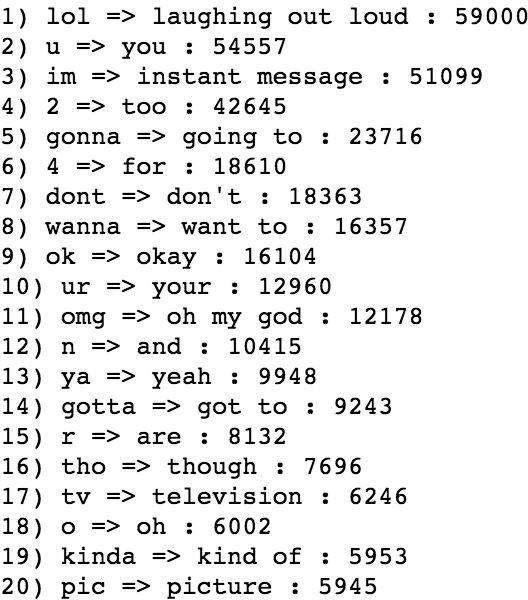


Fig. 6 Top 20 of acronyms in the data set of tweets with their translation and count

### ● 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.



Fig. 7 A tweet before processing negation words.



Fig. 8 A tweet after processing negation words.

### ● Sequence of repeated characters:

Now, we replace all sequences of repeated characters by two characters (e.g: "helloooo"

= "helloo") to keep the emphasized usage of the word.

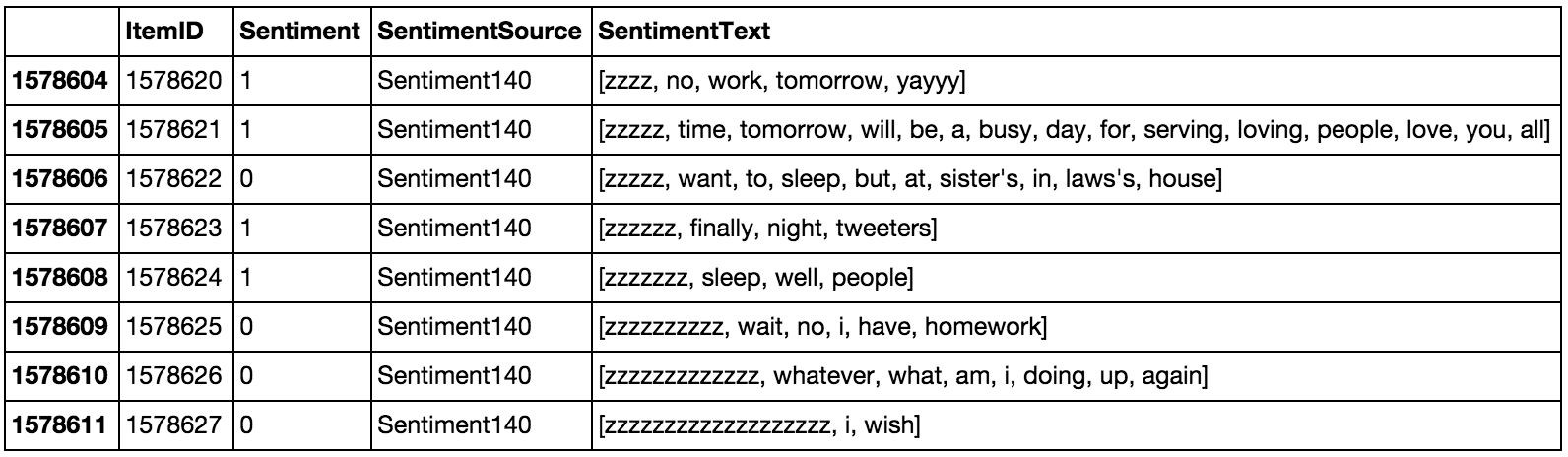


Table 12 Tweets before processing sequences of repeated characters.

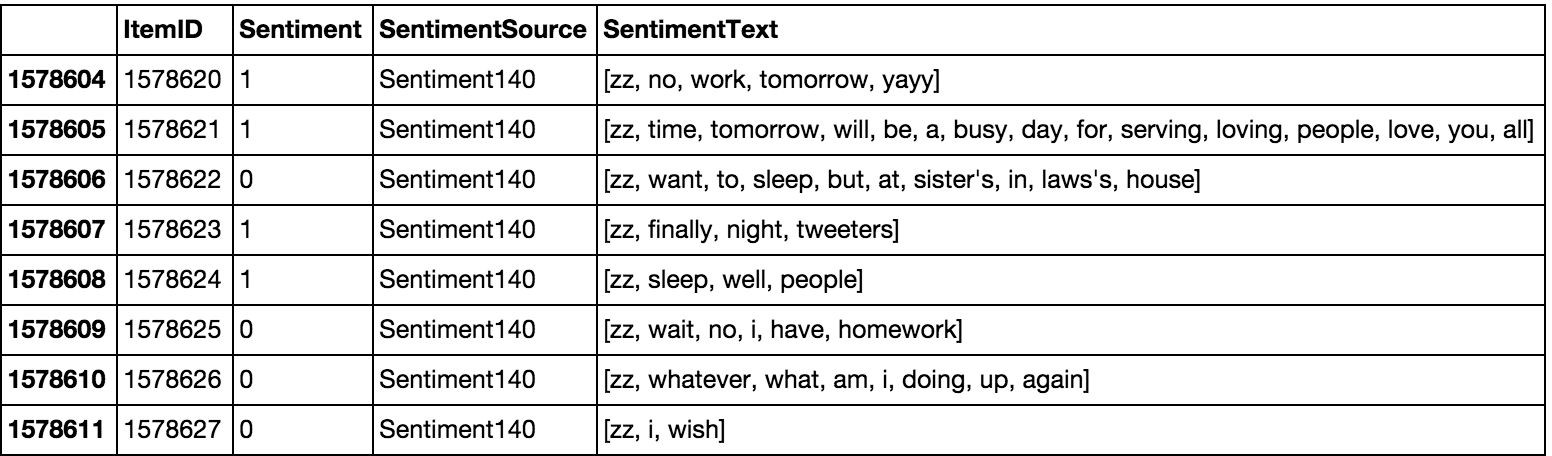


Table 13 Tweets after processing sequences of repeated characte

**Chapter 4**

**Implementation & Results**

**Naïve 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 Bayes’ rule given by the following formula:

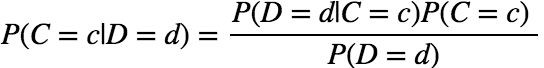


Fig. 9(a) Baye’s rule

where *D* denotes the document and *C*

the category (label), *d*

and *c* are instances of *D* and *C*

and *P* (*D* = *d*) = ∑

*c*∈*C*

*P* (*D* = *d*|*C* = *c*)*P* (*C* = *c*) . We can simplify this expression by,

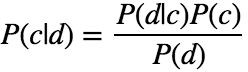


Fig. 9(b) Baye’s rule simlifed

In our case, a tweet *d*

is represented by a vector of *K*

attributes such as *d* = (*w*1, *w*2, ..., *wk*) .

Computing *P* (*d*|*c*) is not trivial and that's why the Naive Bayes introduces the assumption that

all of the feature values *wj* are independent given the category label *c* . That is, for *i* =/ *j* , *wi* and

*wj* are conditionally independent given the category label *c* . So the Baye's rule can be rewritten as,

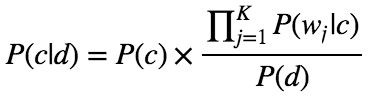


Fig. 9(c) Baye’s rule rewritten

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

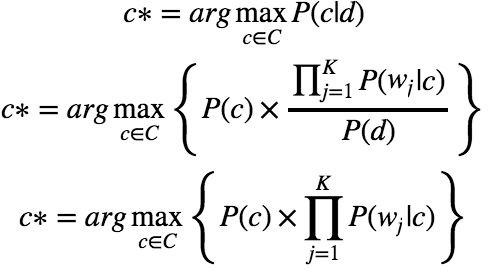


Fig. 9(d) Classifier maximizing the posterior probability P(c|d)

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 **Multi­variate 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.

For text classification, the most used considered as the best choice is the Multinomial Naive Bayes.

The prior distribution *P* (*c*) can be used to incorporate additional assumptions about the relative frequencies of classes. It is computed by:

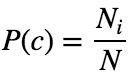


Fig. 9(e) Prior distribution P (c)

where *N* is the total number of training tweets and *Ni* is the number of training tweets in class

*c* .

The likelihood *P* (*wj*|*c*) is usually computed using the formula:

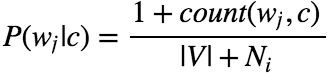


Fig. 9(f) Likelihood P (wj|c)

where *count*(*wj*, *c*) is the number of times that word *wj* occurs within the training tweets of class

*c*, and |*V* | = ∑*wj* the size of the vocabulary. This estimation uses the simplest smoothing

method to solve the **zero­probability problem**, that arises when our model encounters a word seen in the test set but not in the training set, **Laplace** or add­one 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.

### 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 is 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 from 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 **K­fold cross­validation**. We split the data set into k parts, hold out one, combine the others and train on them, then validate against the held­out 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.

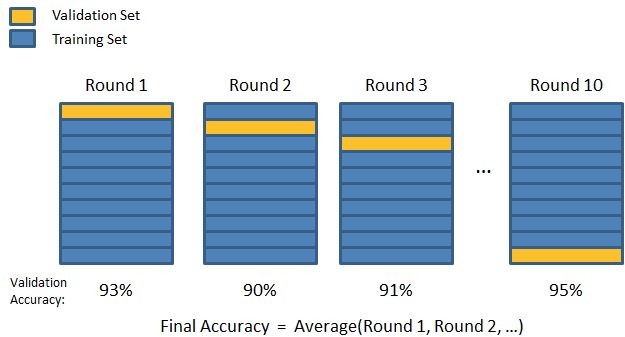


Fig. 10 10­fold cross­validation

We split the training data into 10 folds and cross validate on them using scikit­learn as shown in the figure 2.4.2.1 above. The number of K­folds 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,

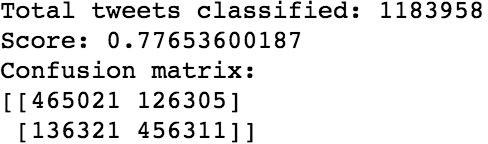


Fig. 11 Result of the naive bayes classifier with the score representing the average of the results of each 10­fold cross­validation, and the overall confusion matrix.

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,

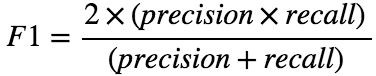


Fig. 9(g) F1 score

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,

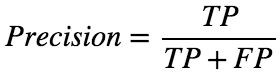


Fig. 9(h) Precision

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

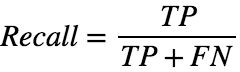


Fig. 9(i) Recall

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 **trade­off between precision and recall** where increasing one decrease the other and we usually use measures that combine precision and recall such as F­measure 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.



Fig. 12 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,

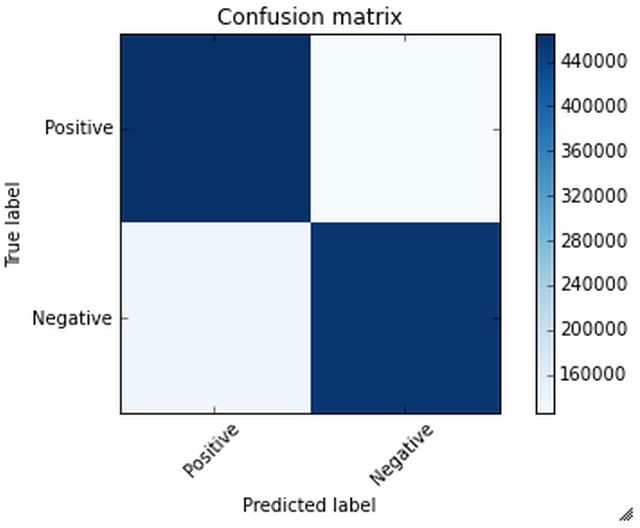


Fig. 13 Colormap of the confusion matrix related to the naive bayes classifier used.

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 k­fold cross validation to evaluate its averaged accuracy.

### 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,

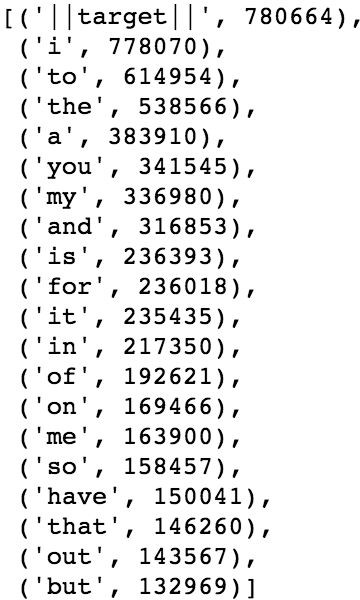


Fig. 14 Most frequent words in the data set with their corresponding count.

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

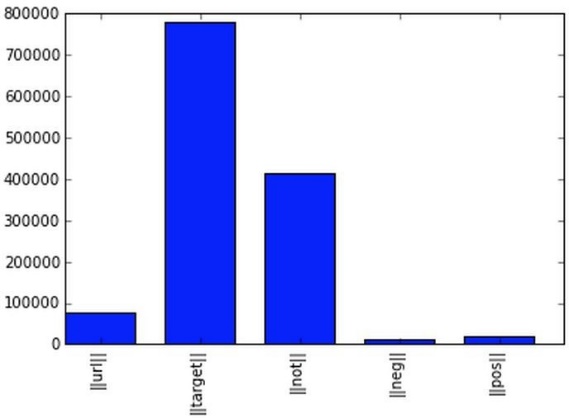


Fig. 15 Tags in the data set with their corresponding count.

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,

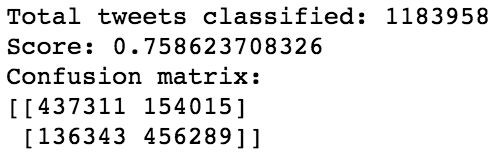


Fig. 16 Result of the naive bayes classifier with stopwords removed.

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,

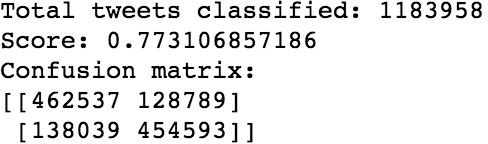


Fig. 17 Result of the naive bayes classifier after stemming.

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.

### Language Models:

Let’s introduce language models to see if we can have better results than those for our baseline. Language models are models assigning **probabilities to sequence of words**.

Initially, they are extensively used in speech recognition and spelling correction but it turns out that they give good results in text classification.

The quality of a language model can be measured by the empirical perplexity (or entropy) using:

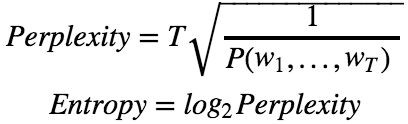


Fig. 9(j) Perplexity and Entropy to evaluate language models.

#### **The goal is to** minimize the perplexity which is the same as maximizing probability**.**

An **N­Gram model** is a type of probabilistic language model for predicting the next item in such a sequence in the form of (n ­ 1) order Markov Model. The Markov assumption is the probability of a word depends only on the probability of a limited history (previous words).



Fig. 9(k) General form of N­grams

A straightforward maximum likelihood estimate of n­gram probabilities from a corpus is given by the observed frequency,

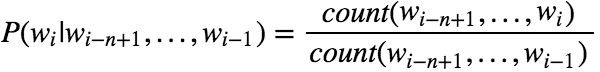


Fig. 9(l) MLE of N­grams

There are several kind of n­grams but the most common are the unigram, bigram and trigram. The **unigram model** make the assumption that every word is independent and so we compute the probability of a sequence using the following formula,

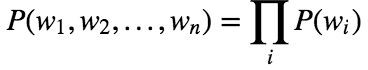


Fig. 9(m) Unigram

In the case of the **bigram model** we make the assumption that **a word is dependent of its previous word**,



Fig. 9(n) Bigram

To estimate the n­gram probabilities, we need to compute the **Maximum Likelihood Estimates**.

For Unigram:

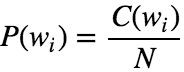


Fig. 9(o) MLE for unigram.

For Bigram:

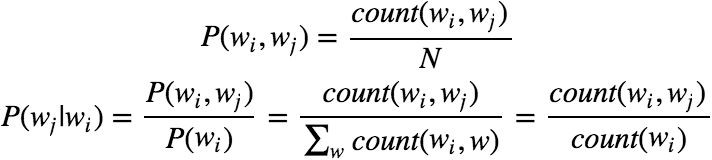


Fig. 9(p) MLE for bigram

Where *N* is the number of words, *C*

means count, *wi* and *wj*

are words.

There are two main practical issues:

* + - * We compute everything in log space (log probabilities) to avoid underflow (multiplying so many probabilities can lead to too small number) and because adding is faster than multiplying (*p*1 × *p*2 × *p*3) = (*logp*1 + *logp*2 + *logp* )

3

* + - * We use smoothing techniques such as Laplace, Witten­Bell Discounting, Good­Turing Discounting to deal with unseen words in the training occurring in the test set.

An N­gram language model can be applied to text classification like Naive Bayes model does. A tweet is categorized according to,



Fig. 9(q) Objective function of n­gram.

and using Baye's rule, this can be rewritten as,

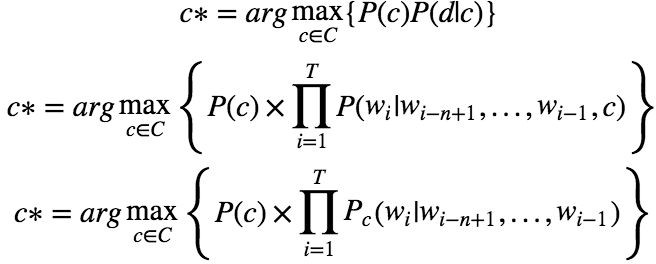


Fig. 9(r) Objective function rewritten using baye’s rule of n­gram.

*P* (*d*|*c*) is the likelihood of *d* under category *c* model.

which can be computed by an n­gram language

#### An important note is that n­gram classifiers are in fact a generalization of Naive Bayes. A unigram classifier with Laplace smoothing corresponds exactly to the traditional naive Bayes classifier.

Since we use bag of words model, meaning we translate this sentence: "I don't like chocolate" into "I", "don't", "like", "chocolate", we could try to use bigram model to take care of negation with "don't like" for this example. Using bigrams as feature in the classifier we get the following results,

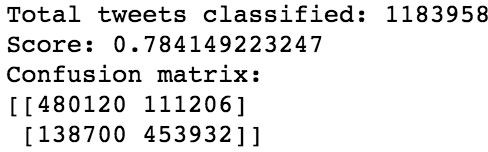


Fig. 18 Results of the naive bayes classifier with bigram features.

Using only bigram features we have slightly improved our accuracy score about 0.01. Based on that we can think of adding unigram and bigram could increase the accuracy score more.

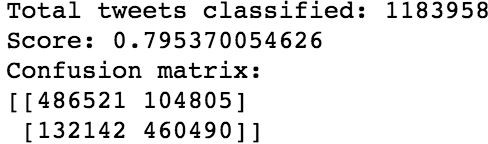


Fig. 19 Results of the naive bayes classifier with unigram and bigram features.

and indeed, we increased slightly the accuracy score about 0.02 compared to the baseline.

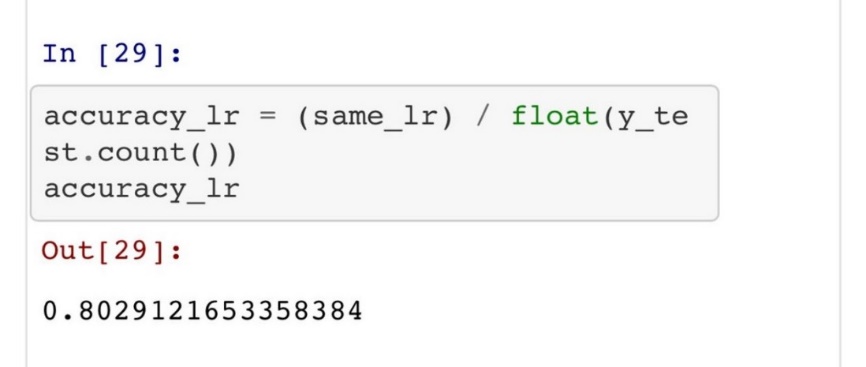


Fig. 20 shows the accuracy of Logistic Regression

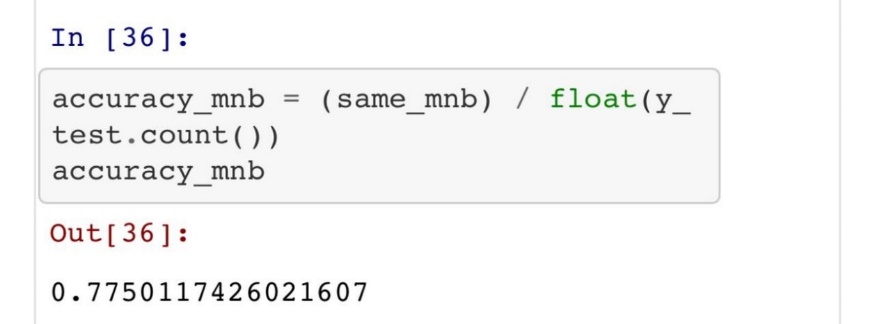


Fig. 21 shows the accuracy of Multinomial Naïve Bayes Theorem

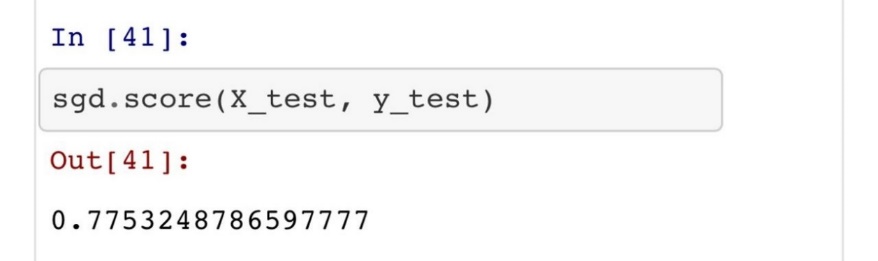


Fig. 22 shows the accuracy of Stochastic Gradient Descent Algorithm

**Chapter 5**

**Conclusion**

Nowadays, sentiment analysis or opinion mining is a hot topic in machine learning. We are still far to detect the sentiments of s corpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Chinese.

In this project we tried to show the basic way of classifying tweets into positive or negative category using Naive Bayes as baseline and how language models are related to the Naive Bayes and can produce better results. We could further improve our classifier by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the naïve Bayes classifier, or trying another classifier all together.

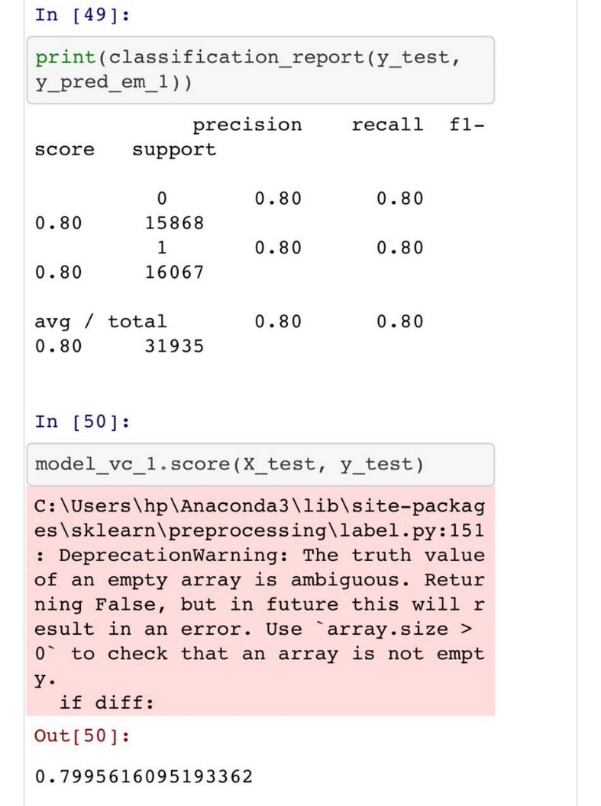


Fig. 23 shows the accuracy of Voting Classifier

**chapter 6**

**References**

1. Alexander Pak, Patrick Paroubek. 2010, Twitter as a Corpus for Sentiment Analysis and Opinion Mining.
2. Alec Go, Richa Bhayani, Lei Huang. Twitter Sentiment Classification using Distant Supervision.
3. Jin Bai, Jian­Yun Nie. Using Language Models for Text Classification.
4. Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau. Sentiment Analysis of Twitter Data.
5. T. J. M. Efthymios Kouloumpis, " Twitter Sentiment Analysis: The Good the Bad and the OMG! ," in *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, Barcelona, 2011.
6. R. R. Neethu M S, " Sentiment Analysis in Twitter using Machine Learning Techniques," in *4th ICCCNT*, Tiruchengode, 2013
7. C. N. S. B. Aliza Sarlan, "Twitter Sentiment Analysis," in *2014 International Conference on Information Technology and Multimedia (ICIMU)*, Putrajaya, Malaysia, 2014.
8. M. M. M. Onam Bharti, "SENTIMENT ANALYSIS ON TWITTER DATA," *International*