A

Project Report

On

"Sentiment Analysis on Twitter Data"

Submitted in partial fulfillment of

the requirements for the 6th Semester Sessional Examination of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

By

RAHUL RAI - 1801210338

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GANDHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

GUNUPUR – 765022

2020 - 21

Gandhi Institute of Engineering & Technology

GUNUPUR – 765 022, Dist: Rayagada (Orissa), India

(Approved by AICTE, Govt. of Orissa and Affiliated to Biju Patnaik University of Technology)

☎: 06857 – 250172(Office), 251156(Principal), 250232(Fax),

e-mail: gandhi_giet@yahoo.com visit us at www.giet.org

ISO 9001:2000 Certified Institute

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project work entitled "SENTIMENT ANALYSIS ON TWITTER DATA" is done by RAHUL RAI, 1801210338 in partial fulfillment of the requirements for the 6th Semester Sessional Examination of Bachelor of Technology in INFORMATION TECHNOLOGY during the academic year 2020-21. This work is submitted to the department as a part of evaluation of 6th Semester Project.

Mr. S. Gopal Krishna Patro
Class Teacher

Prof. (Dr). Sanjay Kumar Kuanar
HoD, CSE

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. We take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

A special gratitude to our HOD Dr. Sanjay Kumar Kuanar,

Thanks to Department Project Coordinator Mr. Bhavani Sankar Panda,

Thanks to Mr. S. Gopal Krishna Patro,

Thanks to Department staff members.

1.	Title of the Project:	Sentiment Analysis on Twitter Data
2.	Project Members:	Rahul Rai Amit Kumar Deepak Kumar Singh Vishal Bauri

Project Version Control History

Version	Primary Author	Description of Version	Date Completed
Final	Rahul Rai Amit Kumar Vishal Bauri Deepak Kumar Singh	Project Report	4th April, 2021

ABSTRACT

Sentiment Analysis is the process of ‘computationally’ determining whether a piece of writing is positive, negative or neutral. It’s also known as opinion mining, deriving the opinion or attitude of a speaker.

Why sentiment analysis?

- Business: In the marketing field companies use it to develop their strategies, to understand customers’ feelings towards products or brands, how people respond to their campaigns or product launches and why consumers don’t buy some products.

- Politics: In the political field, it is used to keep track of political view, to detect consistency and inconsistency between statements and actions at the government level. It can be used to predict election results as well!

- Public Actions: Sentiment analysis also is used to monitor and analyse social phenomena, for the spotting of potentially dangerous situations and determining the general mood of the blogosphere.

In this report, we address the problem of sentiment classification on twitter dataset. We use a number of text preprocessing and Naive Bayes classifier methods to perform sentiment analysis. In the end, we use a accuracy classifying method of different text preprocessing to achieve the classification accuracy of 76.78% on Kaggle public leaderboard.

Keywords — Twitter Sentiment Analysis; Naive Bayes classifier, contraction, emoji classification, tokenization, stemming, lemmatization, stop words, count vectorizer, tf-idf vectorizer, unigram, bigram, trigram, ROC curve, precision, recall, accuracy, python 3.

RELATED WORK

Applying sentiment analysis on Twitter is the upcoming trend with researchers recognizing the scientific trials and its potential applications. The challenges unique to this problem area are largely attributed to the dominantly IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 3, July 2012 ISSN (Online): 1694-0814 www.IJCSI.org 372 Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved. informal tone of the micro blogging. Pak and Paroubek [5] rationale the use of microblogging and more particularly Twitter as a corpus for sentiment analysis. They cited: Microblogging platforms are used by different people to express their opinion about different topics, thus it is a valuable source of people's opinions.

Twitter contains an enormous number of text posts and it grows every day. The collected corpus can be arbitrarily large. Twitter's audience varies from regular users to celebrities, company representatives, politicians, and even country presidents. Therefore, it is possible to collect text posts of users from different social and interests groups. Twitter's audience is represented by users from many countries. Parikh and Movassate implemented two Naive Bayes unigram models, a Naive Bayes bigram model and a Maximum Entropy model to classify tweets. They found that the Naive Bayes classifiers worked much better than the Maximum Entropy model could.

Go et al. proposed a solution by using distant supervision, in which their training data consisted of tweets with emoticons. This approach was initially introduced by Read. The emoticons served as noisy labels. They build models using Naive Bayes, MaxEnt and Support Vector Machines (SVM). Their feature space consisted of unigrams, bigrams and POS. The reported that SVM outperformed other models and that unigram were more effective as features. Pak and Paroubek have done similar work but classify the tweets as objective, positive and negative.

In order to collect a corpus of objective posts, they retrieved text messages from Twitter accounts of popular newspapers and magazines, such as "New York Times", "Washington Posts" etc. Their classifier is based on the multinomial Naïve Bayes classifier that uses N-gram and POS-tags as features. Barbosa et al. too classified tweets as objective or subjective and then the subjective tweets were classified as positive or negative. The feature space used included features of tweets like retweets, hashtags, link, punctuation and exclamation marks in conjunction with features like prior polarity of words and POS of words. Mining for entity opinions in Twitter, Batra and Rao used a dataset of tweets spanning two months starting from June 2009. The dataset has roughly 60 million tweets.

The entity was extracted using the Stanford NER, user tags and URLs were used to augment the entities found. A corpus of 200,000 product reviews that had been labeled as positive or negative was used to train the model. Using this corpus the model computed the probability that a given unigram or bigram was being used in a positive context and the probability that it was being used in a negative context.

Bifet and Frank used Twitter streaming data provided by Firehouse, which gave all messages from every user in real-time. They experimented with three fast incremental methods that were well-suited to deal with data streams: multinomial naive Bayes, stochastic gradient descent, and the Hoeffding tree. They concluded that the SGD-based model, used with an appropriate learning rate was the best. Agarwal et al. approached the task of mining sentiment from twitter, as a 3-way task of classifying sentiment into positive, negative and neutral classes. They experimented with three types of models: unigram model, a feature based model and a tree kernel based model. For the tree kernel based model they designed a new tree representation for tweets. The feature based model that uses 100 features and the unigram model uses over 10,000 features. They concluded features that combine prior polarity of words with their parts-of-speech tags are most important for the classification task. The tree kernel based model outperformed the other two.

The Sentiment Analysis tasks can be done at several levels of granularity, namely, word level, phrase or sentence level, document level and feature level . As Twitter allows its users to share short pieces of information known as “tweets” (limited to 140 characters), the word level granularity aptly suits its setting. Survey through the literature substantiates that the methods of automatically annotating sentiment at the word level fall into the following two categories:

i. Dictionary-based approaches
ii. Corpus-based approaches.

Further, to automate sentiment analysis, different approaches have been applied to predict the sentiments of words, expressions or documents. These include Natural Language Processing (NLP) and Machine Learning (ML) algorithms. In our attempt to mine the sentiment from twitter data we introduce a hybrid approach which combines the advantages of both dictionary & corpus based methods along with the combination of NLP & ML based techniques. The following sections illustrate the proposed paradigm.

DEFINITION AND MOTIVATION

Sentiment analysis is a strategy for checking assessments of people or groups; for example, a portion of a brand’s followers for an individual customer in correspondence with a customer support representative. With regard to a scoring mechanism, (IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 10, No. 2, 2019362 | P a g ewww.ijacsa.thesai.orgsentiment analysis monitors discussions and assesses dialogue and voice affectations to evaluate moods and feelings,especially those associated with a business, product or service,or theme.Sentiment analysis is a means of assessing written orspoken languages to decide whether articulation is positive,negative or neutral and to what degree. The current analysis tools in the market are able to deal with tremendous volumes of customer criticism reliably and precisely.

In conjunction with contents investigation, sentiment analysis discovers customer’s opinions on various topics, including the purchase of items,provision of services, or presentation of promotions.Immense quantities of client-created web-based social networking communications are being persistently delivered in the forms of surveys, online journals, comments, discourses,pictures, and recordings. These correspondences offer significant opportunities to obtain and comprehend the point of view of clients on themes such as intrigue and provide data equipped for clarifying and anticipating business and social news, such as product offers, stock returns, and the results of political decisions.

Integral to these examinations is the assessment of the notions communicated between clients in their content interchanges.“Notion examination” is a dynamic area of research designed to enhance computerized understanding of feelings communicated in content, with increases in implementation prompting more powerful utilization of the inferred data.Among the different web-based social networking platforms,Twitter has incited particularly far-reaching client appropriation and rapid development in terms of correspondence volume.Twitter is a small-scale blogging stage where clients generate 'tweets' that are communicated to their devotees or to another client.

As of 2016, Twitter has more than 313 million dynamic clients inside a given month, including 100 million clients daily. Client origins are widespread, with 77%situated outside of the US, producing more than 500 million tweets every day. The Twitter site positioned twelfth universally for activity in 2017 and reacted to more than 15billion API calls every day. Twitter content likewise shows up in more than one million outsider sites. In Accordance with this enormous development, Twitter has oflate been the subject of much scrutiny, as Tweets frequently express client's sentiment on controversial issues. In the social media context, sentiment analysis and mining opinions are highly challenging tasks, and this is due to the enormous information generated by humans and machines.

IMPORTANCE AND BACKGROUND

Natural Language Processing (NLP) deals with actual text element processing. The text element is transformed into machine format by NLP. Artificial Intelligence (AI) uses information provided by the NLP and applies a lot of maths to determine whether something is positive or negative. Several methods exist to determine an author’s view on a topic from natural language textual information.

Some form of machine learning approach is employed and which has varying degree of effectiveness. One of the types of natural language processing is opinion mining which deals with tracking the mood of the people regarding a particular product or topic. This software provides automatic extraction of opinions, emotions and sentiments in text and also tracks attitudes and feelings on the web. People express their views by writing blog posts, comments, reviews and tweets about all sorts of different topics. Tracking products and brands and then determining whether they are viewed positively or negatively can be done using the web. The opinion mining has slightly different tasks and many names, e.g. sentiment analysis, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, review mining, etc.

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation. Modern NLP algorithms are based on machine learning, especially statistical machine learning.

The paradigm of machine learning is different from that of most prior attempts at language processing. Prior implementations of language-processing tasks typically involved the direct hand coding of large sets of rules. The machine-learning paradigm calls instead for using general learning algorithms — often, although not always, grounded in statistical inference — to automatically learn such rules through the analysis of large corpora of typical real-world examples. A corpus (plural, "corpora") is a set of documents (or sometimes, individual sentences) that have been hand-annotated with the correct values to be learned. Many different classes of machine learning algorithms have been applied to NLP tasks. These algorithms take as input a large set of "features" that are generated from the input data. Some of the earliest-used algorithms, such as decision trees, produced systems of hard if-then rules similar to the systems of hand-written rules that were then common. Increasingly, however, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to each input feature.

Such models have the advantage that they can express the relative certainty of many different possible answers rather than only one, producing more reliable results when such a model is included as a component of a larger system. Systems based on machine-learning algorithms have many advantages over hand-produced rules: The learning procedures used during machine learning automatically focus on the most common cases, whereas when writing rules by hand it is often not at all obvious where the effort should be directed. Automatic learning procedures can make use of statistical inference algorithms to produce models that are robust to unfamiliar input (e.g. containing words or structures that have not been seen before) and to erroneous input (e.g. with misspelled words or words accidentally omitted).

Generally, handling such input gracefully with hand-written rules — or more generally, creating systems of hand-written rules that make soft decisions — is extremely difficult, error-prone and time-consuming. Systems based on automatically learning the rules can be made more accurate simply by supplying more input data. However, systems based on hand-written rules can only be made more accurate by increasing the complexity of the rules, which is a much more difficult task. In particular, there is a limit to the complexity of systems based on hand-crafted rules, beyond which the systems become more and more unmanageable.

However, creating more data to input to machine-learning systems simply requires a corresponding increase in the number of man-hours worked, generally without significant increases in the complexity of the annotation process. The subfield of NLP devoted to learning approaches is known as Natural Language Learning (NLL) and its conference CoNLL and peak body SIGNLL are sponsored by ACL, recognizing also their links with Computational Linguistics and Language Acquisition. When the aim of computational language learning research is to understand more about human language acquisition, psycholinguistics, NLL overlaps into the related field of Computational Psycholinguistics.

Opinions are fundamental to every single human action since they are key influencers of our practices. At whatever point we have to settle on a choice, we need to know others' thoughts. In reality, organizations and associations dependably need to discover users' popular sentiments about their items and services. Clients use different types of online platforms for social engagement including web-based social networking sites; for example, Facebook and Twitter. Through these web based social networks, buyer engagement happens progressively. This kind of connection offers a remarkable open door for advertising knowledge.

Individuals of every nationality, sexual orientation, race and class utilize the web to share encounters and impressions about virtually every feature of their lives. Other than composing messages, blogging or leaving remarks on corporate sites, a great many individuals utilize informal organization destinations to log opinions,express feelings and uncover insights about their everyday lives. Individuals compose correspondence on nearly anything,including films, brands, or social exercises. These logs circulate throughout online groups and are virtual gatherings where shoppers illuminate and impact others.

To the advertiser,these logs provide profound snippets of insight into purchasers behavioral inclinations and present a continuous opportunity to find out about client emotions and recognitions, as they happen without interruption or incitement. Be that as it may, recent explosions in client-produced content on social sites are introducing unique difficulties in capturing, examining andtranslating printed content since information is scattered,confused, and divided.Opinion investigation is a method of information mining that can overcome these difficulties by methodically separating and dissecting web-based information without causing delays.With conclusion examination, advertisers are able to discover shoppers' emotions and states of mind continuously, in spite of the difficulties of information structure and volume. The Enthusiasm in this study for utilizing sentiment analysis as an instrument for promoting research instruments is twofold.Sentiment analysis critically encourages organizations to determine customers' likes and dislikes about products and company image. In addition, it plays a vital role in analyzing data of industries and organizations to aid them in making business decisions.

PROBLEM STATEMENT

Twitter is a popular social networking website where members create and interact with messages known as “tweets”. This serves as a means for individuals to express their thoughts or feelings about different subjects. Various different parties such as consumers and marketers have done sentiment analysis on tweets to gather insights into products or to conduct market analysis. Furthermore, with the recent advancements in machine learning algorithms, we are able to improve the accuracy of our sentiment analysis predictions.

In this report, we will attempt to conduct sentiment analysis on “tweets” using various different text preprocessing and Naïve Bayes Classifier algorithms. We attempt to classify the polarity of the tweet where it is either positive or negative. If the tweet has both positive and negative elements, the more dominant sentiment should be picked as the final label.

We use the dataset from Kaggle which was crawled and labeled positive/negative. The data provided comes with emoticons, usernames and hashtags which are required to be processed and converted into a standard form. We also need to extract useful features from the text such as unigrams, bigrams and trigrams which is a form of representation of the “tweet”. We use Naïve-Bayes Classifier to conduct sentiment analysis using the extracted features. However, just relying on an individual model did not give a high accuracy.

WORKING PRINCIPLE

What is Sentiment Analysis?

Sentiment analysis (a.k.a opinion mining) is the automated process of identifying and extracting the subjective information that underlies a text. This can be either an opinion, a judgment, or a feeling about a particular topic or subject. The most common type of sentiment analysis is called ‘polarity detection’ and consists in classifying a statement as ‘positive’, ‘negative’ or ‘neutral’.

- For example, let’s take this sentence: “I don’t find the app useful: it’s really slow and constantly crashing”. A sentiment analysis model would automatically tag this as Negative.

A sub-field of Natural Language Processing (NLP), sentiment analysis has been getting a lot of attention in recent years due to its many exciting applications in a variety of fields, ranging from business to political studies.

Thanks to sentiment analysis, companies can understand the reputation of their brand. By analyzing social media posts, product reviews, customer feedback, or NPS responses (among other sources of unstructured business data), they can be aware of how their customers feel about their product. They can also track specific topics and get relevant insights on how people are talking about those topics.

Sentiment analysis is particularly useful for social media monitoring because it goes beyond metrics that focus on the number of likes or retweets, and provides a qualitative point of view.

Let’s say a company has just launched a new product feature and you notice a sharp increase in mentions on Twitter. However, receiving tons of mentions does not necessarily mean a good thing. Are customers tweeting more because they are expressing good things about this new product feature? Or, are customers actually complaining about the feature having lots of bugs? Performing Twitter sentiment analysis can be an excellent way to understand the tone of those mentions and obtain real-time insights on how users are perceiving your new product.

Sentiment analysis uses:

Sentiment analysis is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics. Social media monitoring tools like Brandwatch Analytics make that process quicker and easier than ever before, thanks to real-time monitoring capabilities.

The applications of sentiment analysis are broad and powerful. The ability to extract insights from social data is a practice that is being widely adopted by organisations across the world.

Shifts in sentiment on social media have been shown to correlate with shifts in the stock market.

The Obama administration used sentiment analysis to gauge public opinion to policy announcements and campaign messages ahead of 2012 presidential election. Being able to quickly see the sentiment behind everything from forum posts to news articles means being better able to strategise and plan for the future.

It can also be an essential part of your market research and customer service approach. Not only can you see what people think of your own products or services, you can see what they think about your competitors too. The overall customer experience of your users can be revealed quickly with sentiment analysis, but it can get far more granular too.

The ability to quickly understand consumer attitudes and react accordingly is something that Expedia Canada took advantage of when they noticed that there was a steady increase in negative feedback to the music used in one of their television adverts.

Understanding Sentiment: Tweet

Sentiment analysis conducted by the brand revealed that the music played on the commercial had become incredibly irritating after multiple airings, and consumers were flocking to social media to vent their frustrations.

A couple of weeks after the advert first aired, over half of online conversation about the campaign was negative.

Rather than chalking up the advert as a failure, Expedia was able to address the negative sentiment in a playful and self-knowing way by airing a new version of the advert which featured the offending violin being smashed.

Sentiment analysis is the automated process of analyzing text data and classifying opinions as negative, positive or neutral. Usually, besides identifying the opinion, these systems extract attributes of the expression e.g.:

- Polarity: if the speaker express a positive or negative opinion,
- Subject: the thing that is being talked about,
- Opinion holder: the person, or entity that expresses the opinion.

Sentiment Analysis Challenges

Natural Language Processing (NLP) deals with actual text element processing. The text element is transformed into machine format by NLP. Artificial Intelligence (AI) uses information provided by the NLP and applies a lot of maths to determine whether something is positive or negative. Several methods exist to determine an author's view on a topic from natural language textual information. Some form of machine learning approach is employed and which has varying degree of effectiveness. One of the types of natural language processing is opinion mining which deals with tracking the mood of the people regarding a particular product or topic. This software provides automatic extraction of opinions, emotions and sentiments in text and also tracks attitudes and feelings on the web.

People express their views by writing blog posts, comments, reviews and tweets about all sorts of different topics. Tracking products and brands and then determining whether they are viewed positively or negatively can be done using web. The opinion mining has slightly different tasks and many names, e.g. sentiment analysis, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, review mining, etc. Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation. Modern NLP algorithms are based on machine learning, especially statistical machine learning. The paradigm of machine learning different from that of most prior attempts at language processing.

Prior implementations of language-processing tasks typically involved the direct hand coding of large sets of rules. The machine-learning paradigm calls instead for using general learning algorithms — often, although not always, grounded in statistical inference — to automatically learn such rules through the analysis of large corpora of typical real-world examples. A corpus (plural, "corpora") is a set of documents (or sometimes, individual sentences) that have been hand-annotated with the correct values to be learned. Many different classes of machine learning algorithms have been applied to NLP tasks. These algorithms take as input a large set of "features" that are generated from the input data. Some of the earliest-used algorithms, such as decision trees, produced systems of hard if-then rules similar to the systems of hand-written rules that were then common.

Increasingly, however, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to each input feature. Such models have the advantage that they can express the relative certainty of many different possible answers rather than only one, producing more reliable results when such a model is included as a component of a larger system. Systems based on machine-learning algorithms have many advantages over hand-produced rules:

The learning procedures used during machine learning automatically focus on the most common cases, whereas when writing rules by hand it is often not at all obvious where the effort should be directed. Automatic learning procedures can make use of statistical inference algorithms to produce models that are robust to unfamiliar input (e.g. containing words or structures that have not been seen before) and to erroneous input (e.g. with misspelled words or words accidentally omitted). Generally, handling such input gracefully with hand-written rules — or more generally, creating systems of hand-written rules that make soft decisions — is extremely difficult, error-prone and time-consuming. Systems based on automatically learning the rules can be made more accurate simply by supplying more input data. However, systems based on hand-written rules can only be made more accurate by increasing the complexity of the rules, which is a much more difficult task.

In particular, there is a limit to the complexity of systems based on hand-crafted rules, beyond which the systems become more and more unmanageable. However, creating more data to input to machine-learning systems simply requires a corresponding increase in the number of man-hours worked, generally without significant increases in the complexity of the annotation process. The subfield of NLP devoted to learning approaches is known as Natural Language Learning (NLL) and its conference CoNLL and peak body SIGNLL are sponsored by ACL, recognizing also their links with Computational Linguistics and Language Acquisition.

When the aim of computational language learning research is to understand more about human language acquisition, or psycholinguistics, NLL overlaps into the related field of Computational Psycholinguistics.

Most of the work in sentiment analysis in recent years has been around developing more accurate sentiment classifiers by dealing with some of the main challenges and limitations in the field.

- Subjectivity and Tone :
  The detection of subjective and objective texts is just as important as analyzing their tone. In fact, so-called objective texts do not contain explicit sentiments. Say, for example, we intend to analyze the sentiment of the following two texts:

  - The package is nice.
  - The package is red.

  Most people would say that sentiment is positive for the first one and neutral for the second one, right? All predicates (adjectives, verbs, and some nouns) should not be treated the same with respect to how they create sentiment. In the examples above, nice is more subjective than red.

- Context and Polarity:

  All utterances are uttered at some point in time, in some place, by and to some people. All utterances are uttered in context. Analyzing sentiment without context gets pretty difficult. However, machines cannot learn about contexts if they are not mentioned explicitly. One of the problems that arise from context is changes in polarity. Look at the following responses to a survey:

  - Everything of it.
  - Absolutely nothing!

  Imagine the responses above come from answers to the question What did you like about the event? The first response would be positive and the second one would be negative, right? Now, imagine the responses come from answers to the question What did you Dislike about the event? The negative in the question will make sentiment analysis change altogether.

  A good deal of preprocessing or post processing will be needed if we are to take into account at least part of the context in which texts were produced. However, how to preprocess or post process data in order to capture the bits of context that will help analyze sentiment is not straightforward.

- Irony and Sarcasm

  Differences between literal and intended meaning (i.e. irony) and the more insulting or ridiculing version of irony (i.e. sarcasm) usually change positive sentiment into negative whereas negative or neutral sentiment might be changed to positive. However, detecting irony or sarcasm takes a good deal of analysis of the context in which the texts are produced and, therefore, are really difficult to detect automatically.

  For example, look at some possible answers to the question Have you had a nice customer experience with us? below.

  - Yeah. Sure.
  - Not one, but many!

  What sentiment would we assign to the responses above? Probably, we have listened to the first response so many times, we would have said negative, right? The problem is there is no textual cue that will make a machine learn that negative sentiment since most often, yeah and sure belong to positive or neutral texts.

  How about the second response? In this context, sentiment is positive, but we’re sure we can come up with many different contexts in which the same response can express negative sentiment.

- Comparisons

  How to treat comparisons in sentiment analysis is another challenge worth tackling. Look at the texts below:

  - This product is second to none.
  - This is better than the old tools.
  - This is better than nothing.

  There are some comparisons like the first one above that do not need any contextual clues in order to be classified correctly.

  The second and third texts are a little more difficult to classify, though. Would we classify them as neutral or positive? Probably, we are more likely to choose positive for the second one and neutral for the third, right? Once again, context can make a difference. For example, if the old tools the second text talks about were considered useless in context, then the second text turns out to be pretty similar to the third text. However, if no context is provided, these texts feel different.

- Emojis

  There are two types of emojis according to Guibon et al.. Western emojis (e.g. :D) are encoded in only one character or in a combination of a couple of them whereas Eastern emojis (e.g. ¯ \ _ (ツ) _ / ¯) are a longer combination of characters of a vertical nature. Particularly in tweets, emojis play a role in the sentiment of texts.

  Sentiment analysis performed over tweets requires special attention to character-level as well as word-level. However, no matter how much attention we pay to each of them, a lot of preprocessing might be needed. For example, we might want to preprocess social media content and transform both Western and Eastern emojis into tokens and whitelist them (i.e. always take them as a feature for classification purposes) in order to help improve sentiment analysis performance.

PROCESS

Section A: Preparing The Test Set

- Step A.1: Getting the authentication credentials
- Step A.2: Authenticating our Python script
- Step A.3: Creating the function to build the Test set

Section B: Preparing The Training Set

Section C: Pre-processing Tweets in The Data Sets

Section D: Naive Bayes Classifier

- Step D.1: Building the vocabulary
- Step D.2: Matching tweets against our vocabulary
- Step D.3: Building our feature vector
- Step D.4: Training the classifier

Section E: Testing The Model

Dataset

The data given is in the form of a comma-separated values files with tweets and their corresponding sentiments. The training dataset is a csv file of type tweet_id, sentiment_tweet where the tweet_id is a unique integer identifying the tweet, sentiment is either 1 (positive) or 0 (negative), and tweet is the tweet enclosed in "". We will use this single dataset to cross validate our model.

The dataset is a mixture of words, emoticons, symbols, URLs, hashtags and references to people. Words and emoticons contribute to predicting the sentiment, but URLs and references to people don't. Therefore, URLs and references can be ignored. The words are also a mixture of misspelled words, extra punctuation, and words with many repeated letters. The tweets, therefore, have to be preprocessed to standardize the dataset.

Range Index: 99989 entries, 0 to 99988

Why Twitter Data?

Twitter is an online micro blogging tool that disseminates more than 400 million messages per day, including vast amounts of information about almost all industries from entertainment to sports, health to business etc. One of the best things about Twitter—indeed, perhaps its greatest appeal—is in its accessibility. It's easy to use both for sharing information and for collecting it. Twitter provides unprecedented access to our lawmakers and our celebrities, as well as to news as it's happening. Twitter represents an important data source for the business models of huge companies as well.

All the above characteristics make twitter a best place to collect real time and latest data to analyse and do any sought of research for real life situations.

If you need other datasets, you can download pre-existing datasets of various use cases like cancer detection to Q&A dataset to sports comments to chatbots. Here are various text classification datasets.

Or if you have your own unique use case, you can create your very own dataset for it. You can download images from the web and to make a big dataset in no time, use an annotation tool like Dataturks, where you upload the raw data and tag manually in a ziffy.

Why is Twitter Sentiment Analysis important?

Nearly 80% of the world's digital data is unstructured, and data obtained from social media sources is no exception to that. Since the information is not organized in any predefined way, it's difficult to sort and analyze. Fortunately, thanks to the developments in Machine Learning and NLP, it is now possible to create models that learn from examples and can be used to process and organize text data.

Twitter sentiment analysis systems allow you to sort large sets of tweets and detect the polarity of each statement automatically. And the best part, it's fast and simple, saving teams valuable hours and allowing them to focus on tasks where they can make a bigger impact.

These are some of the main advantages of Twitter sentiment analysis:

- **Scalability**: let's say you need to analyze hundreds of tweets mentioning a brand. While you could do that manually, it would take hours and hours of manual processing and would end up being inconsistent and impossible to scale. By performing Twitter sentiment analysis you can automate this task and obtain cost-effective results in a very short time.

- **Real-Time Analysis**: Twitter sentiment analysis is critical to notice sudden shifts in customer moods, detect if critics and complaints are increasing and take action before the problem escalates. You can be monitoring your brand in real-time and get valuable insights that allow you to make changes or improvements when needed.

- **Consistent Criteria**: analyzing sentiment in a text is a subjective task. When done manually, the same tweet may be perceived differently by two members of the same team, and the results will probably be biased. By training a machine learning model to perform sentiment analysis on Twitter, you can set the parameters to analyze all your data and obtain more consistent and accurate results.

How to Use Sentiment Analysis with Twitter Data?

So far we have learned what sentiment analysis is and the benefits it delivers when using it to analyze Twitter data. Now it's time to learn how to actually do Twitter sentiment analysis.

We can identify four main steps in this process:

- Data gathering
- Data preparing
- The creation of the sentiment analysis model
- Visualization of the results

In this section, we'll explain each of these stages and provide tools for both coders and non-coders so you can get started with sentiment analysis of tweets right away.

Data Gathering: How to Get Twitter Data?

If you want to perform Twitter sentiment analysis, the first step is to gather the data. This is the data that you will use for:

1. training the machine learning model, and
2. running the actual sentiment analysis on Twitter data.

At this point, there's something important to consider regarding Twitter. There are two main types of tweets you can extract from Twitter:

Current Tweets: this is useful to track keywords or hashtags in real-time.

Historical Tweets: you can use this to search past tweets during a predefined time frame.

One of the most frequent questions is how to extract data from Twitter. There are different ways to do this, some of them are free, and some require purchasing the data. Besides, some tools are oriented to developers, while others don't need any coding skills.

Pre-processing

- Basic feature extraction

  Pre-processing Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as retweets, hashtags, emoticons, user mentions, etc. which have to be suitably extracted. Therefore, raw twitter data has to be normalized to create a dataset which can be easily learned by various classifiers. We have applied an extensive number of pre-processing steps to standardize the dataset and reduce its size.

  We first do some general pre-processing on tweets which is as follows.

  - Convert the tweet to lower case.
  - Replace 2 or more dots (.) with space.
  - Strip spaces and quotes (" and ') from the ends of tweet.

  We handle special twitter features as follows.

  Usernames and mentions:
  Every twitter user has a handle associated with them. Users often mention other users in their tweets by @handle. We delete all user mentions as they are unique and doesn't contribute anything to sentiment.

URL:
Users often share hyperlinks to other webpages in their tweets. Any particular URL is not important for text classification as it would lead to very sparse features. Therefore, we delete all the URLs in tweets. Thus calculation becomes less resource hungry.

Emoticon:
Users often use a number of different emoticons in their tweet to convey different emotions. It is impossible to exhaustively match all the different emoticons used on social media as the number is ever increasing. However, we match some common emoticons which are used very frequently. We replace the matched emoticons with either ‘positiveemoji’ or ‘negetiveemoji’ depending on whether it is conveying a positive or a negative emotion. A list of all emoticons matched by our method is given below.

Emoticons	Type	Replacement
:) , :) , :-) , (: , (: , (-: , :’) , :O	Smile	positiveemoji
:D , : D , :-D , xD , x-D , XD , X-D	Laugh	positiveemoji
;-) , ;) , ;-D , ;D , (; , (-; , @-)	Wink	positiveemoji
<3 , :*	Love	positiveemoji
:-(, : (, :(,): ,)-: , :-/ , :-|	Sad	negetiveemoji
:,(, :’(, :"(	Cry	negetiveemoji

Hashtag:
Hashtags are un-spaced phrases prefixed by the hash symbol (#) which is frequently used by users to mention a trending topic on twitter. We replace all the hashtags with the words with the hash symbol. For example, #hello is replaced by hello.

After applying tweet level pre-processing, we processed individual words of tweets as follows.

- Strip any punctuation [’"?!,.():;] from the word.
- Convert 2 or more letter repetitions to 2 letters. Some people send tweets like ‘I am sooooo happppppy’ adding multiple characters to emphasize certain words. This is done to handle such tweets by converting them to ‘I am soo happy’.
- Removes any single characters.
- Remove - and ’. This is done to handle words like t-shirt and their’s by converting them to the more general form tshirt and theirs. Here we use contractions. We have created a separate dictionary file to handle contractions. Like ‘can’t’ will be converted into ‘cannot’.
- Replace 2 or more spaces with a single space.

- Advanced text processing

Normalization:

Stemming and Lemmatization are Text Normalization (or sometimes called Word Normalization) techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing.

Tokenization:

Tokenization refers to dividing the text into a sequence of words or sentences. We could use NLTK library, but we used string.split() which is faster.

Stemming:

Stemming refers to the removal of suffices, like “ing”, “ly”, “s”, etc. by a simple rule-based approach. Stemming algorithms work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. This indiscriminate cutting can be successful in some occasions, but not always, and that is why we affirm that this approach presents some limitations. For this purpose, we will use PorterStemmer from the NLTK library.

Form	Suffix	Stem
studies	-es	studi
studying	-ing	stufy

Lemmatization:

Lemmatization is a more effective option than stemming because it converts the word into its root word, rather than just stripping the suffices. It makes use of the vocabulary and does a morphological analysis to obtain the root word. Lemmatization takes into consideration the morphological analysis of the words. To do so, it is necessary to have detailed dictionaries which the algorithm can look through to link the form back to its lemma. Therefore, we usually prefer using lemmatization over stemming.

Form	Morphological information	Lemma
studies	Third person, singular number, present tense of verb study	study
studying	Gerund of the verb study	study

Stop words:

Not all words determine sentiments. Some of them (mainly prepositions and articles) don’t contribute anything to the sentiment. So we can remove these commonly occurring words from our text data. But NLTK library contains stop words like ‘no’, ‘not’, ‘nor’ etc., which determines positive or negative sentiments. So we built custom set of stop words, which doesn't contain any words that can affect sentiment.

NLTK stop words	"i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "no", "nor", "not", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now"
Custom stop words	"i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "only", "own", "same", "so", "than", "too", "very", "can", "will", "just", "should", "now"

Feature Extraction

We extract two types of features from our dataset, namely unigrams and bigrams. We create a frequency distribution of the unigrams and bigrams present in the dataset and choose top N unigrams and bigrams for our analysis.

Unigrams:

Probably the simplest and the most commonly used features for text classification is the presence of single words or tokens in the text. We extract single words from the training dataset and create a frequency distribution of these words.

Bigrams:

Bigrams are word pairs in the dataset which occur in succession in the corpus. These features are a good way to model negation in natural language like in the phrase – This is not good. A total of 1954953 unique bigrams were extracted from the dataset. Out of these, most of the bigrams at end of frequency spectrum are noise and occur very few times to influence classification. We therefore use only top 10000 bigrams from these to create our vocabulary.

N-Grams:

We extract three types of features from our dataset, namely unigrams bigrams and trigrams. N-grams are the combination of multiple words used together. Ngrams with N=1 are called unigrams. Similarly, bigrams (N=2), trigrams (N=3) and so on can also be used.

Unigrams do not usually contain as much information as compared to bigrams and trigrams. The basic principle behind n-grams is that they capture the language structure, like what letter or word is likely to follow the given one. The longer the n-gram (the higher the n), the more context we have to work with. Optimum length really depends on the application – if our n-grams are too short, we may fail to capture important differences. On the other hand, if they are too long, we may fail to capture the "general knowledge" and only stick to particular cases.

Bag of Words:

Bag of Words (BoW) refers to the representation of text which describes the presence of words within the text data. The intuition behind this is that two similar text fields will contain similar kind of words, and will therefore have a similar bag of words. Further, that from the text alone we can learn something about the meaning of the document. We use CountVectorizer from sklearn for this purpose.

Term Frequency – Inverse Document Frequency:

Term frequency is simply the ratio of the count of a word present in a sentence, to the length of the sentence. Therefore, we can generalize term frequency as:

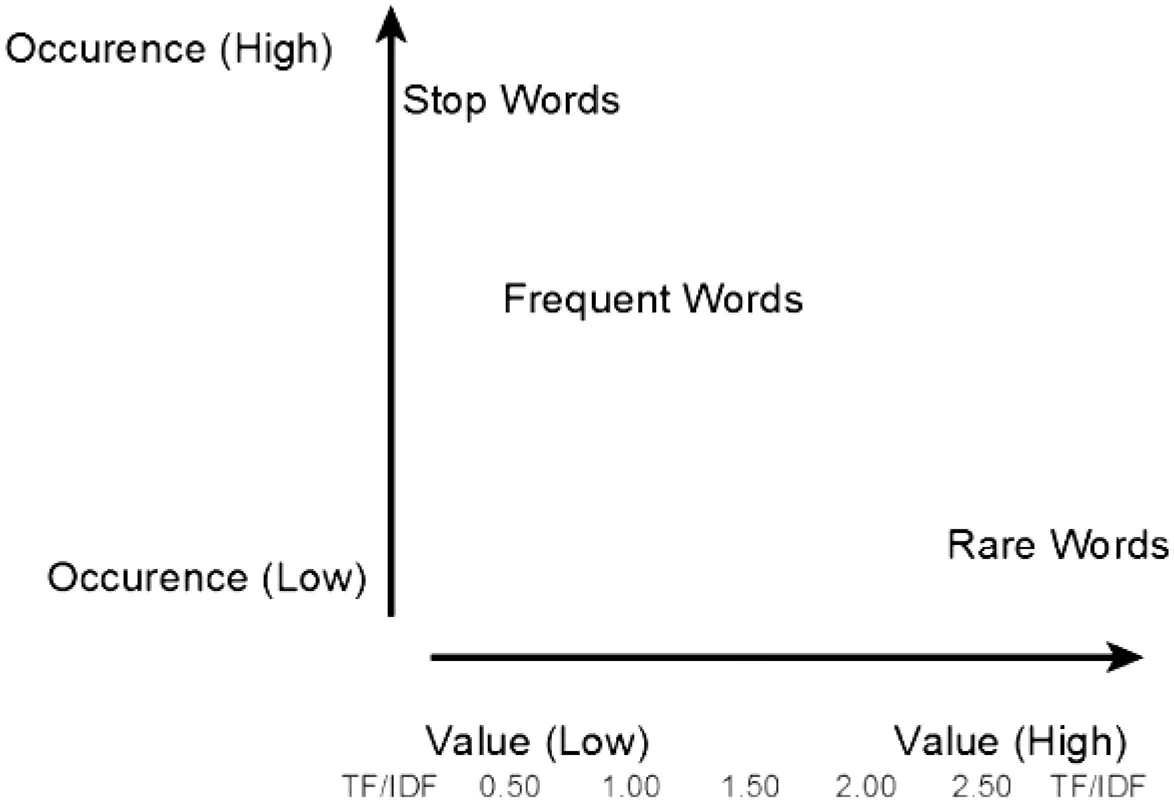
TF = (Number of times term T appears in the particular row) / (number of terms in that row)

The intuition behind inverse document frequency (IDF) is that a word is not of much use to us if it's appearing in all the documents. Therefore, the IDF of each word is the log of the ratio of the total number of rows to the number of rows in which that word is present.

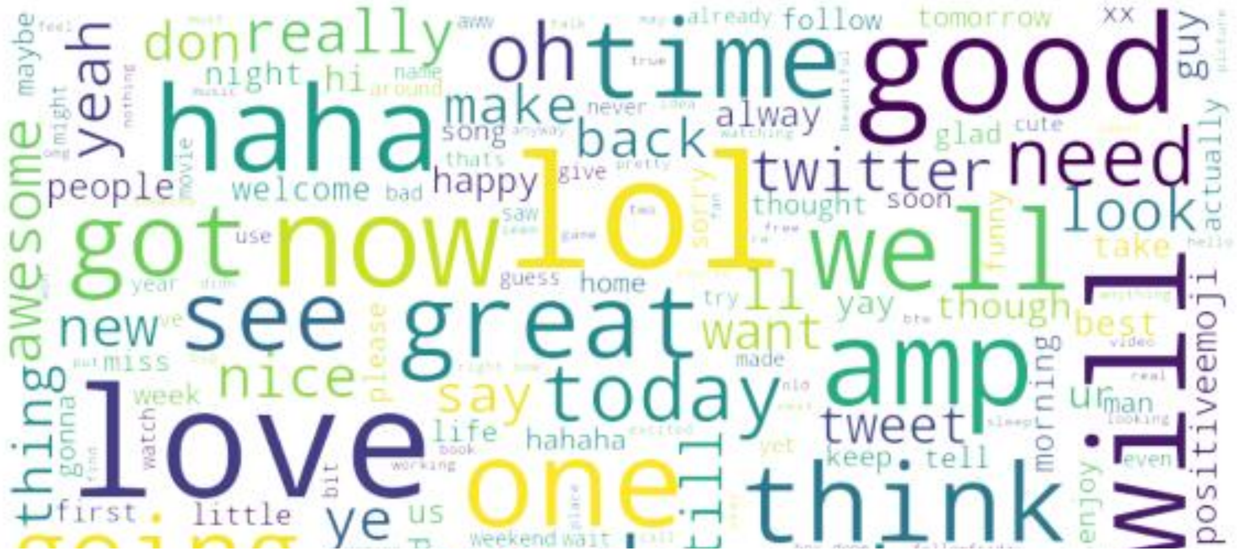
IDF = log (N/n), where, N is the total number of rows and n is the number of rows in which the word was present.

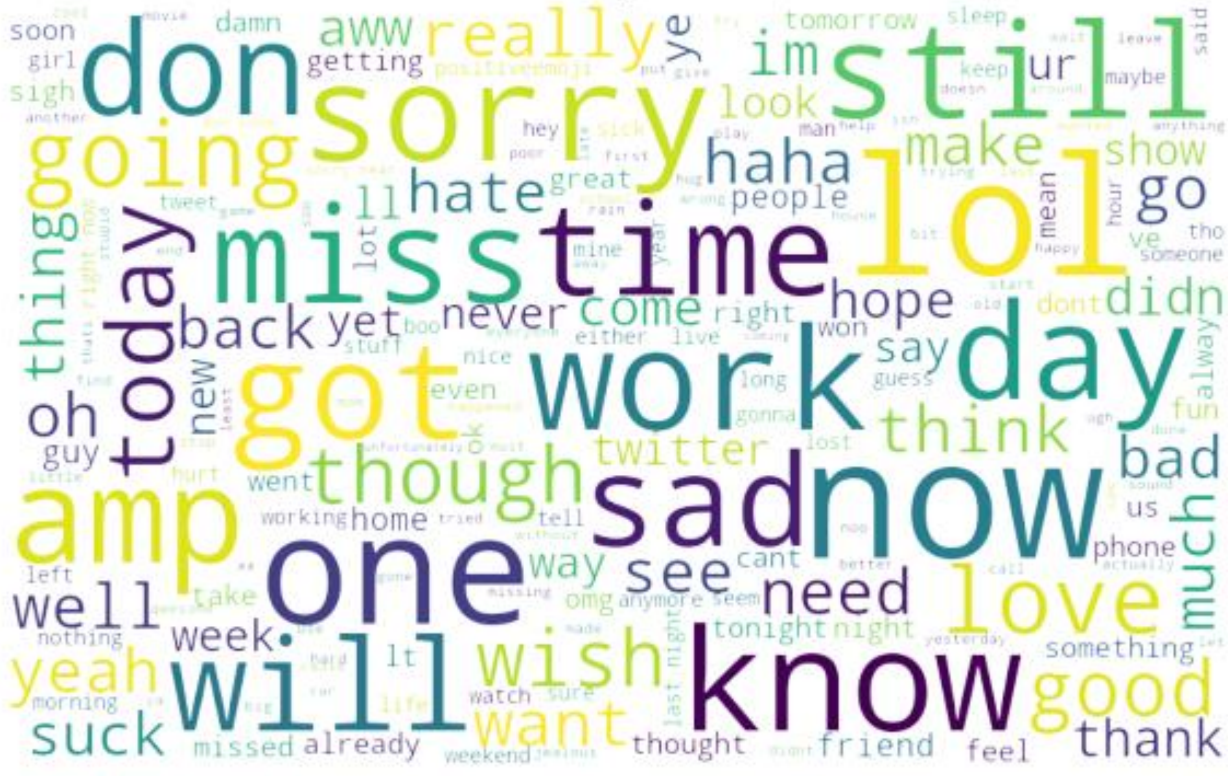
TF-IDF is the multiplication of the TF and IDF which we calculated above.

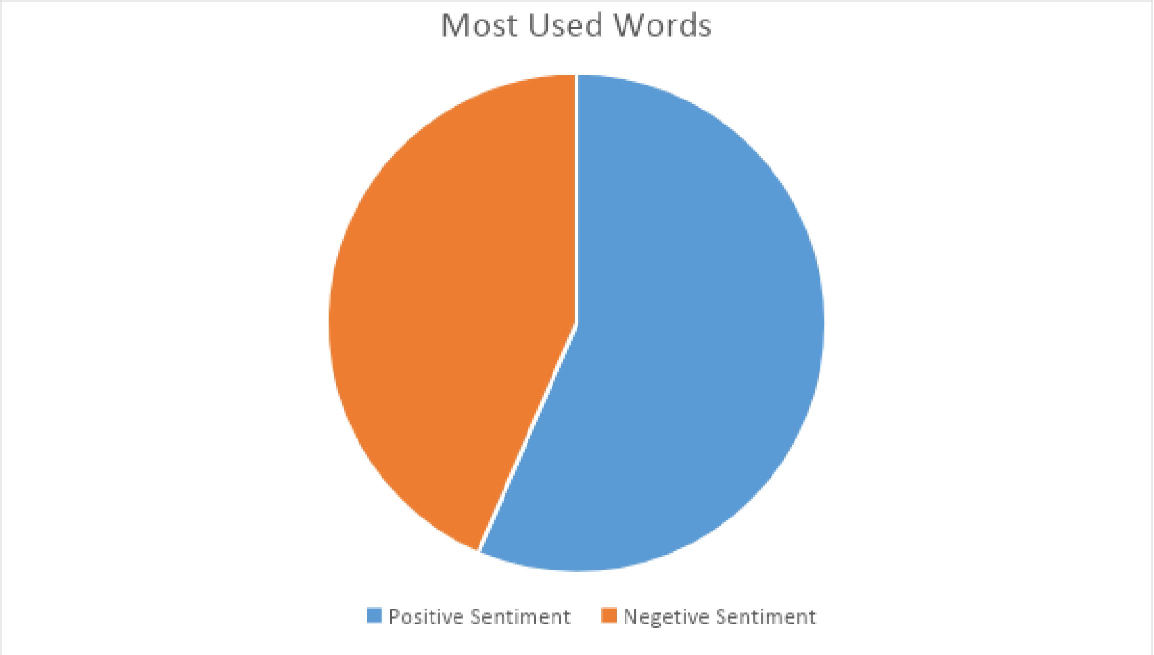
TF-IDF has penalized words like 'don't', 'can't', and 'use' because they are commonly occurring words. However, it has given a high weight to "disappointed" since that will be very useful in determining the sentiment of the tweet.


Occurence (High)
Stop Words
Frequent Words
Rare Words
Occurence (Low)
Value (Low)
Value (High)
TF/IDF
0.50
1.00
1.50
2.00
2.50
TF/IDF

MOST USED WORDS


Most Used Positive Words


Most Used Negetive Words



See the Word ‘lol’ it is used in both positive and negative (sarcastic) sentiments.

Splitting dataset

Spitting the dataset into train and test data for cross validation. We have split the dataset into 8:2 ratio of (train : test) data randomly.

X_train_shape : (79991, 1195309) X_test_shape : (19998, 1195309)

y_train_shape : (79991,) y_test_shape : (19998,)

Naive Bayes Classifier

It is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes Theorem for events A and B, provided that $P\ (B\ ) \neq 0$,

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

- P (A|B) is the posterior probability of class (c, target) given predictor (x, attributes).
- P (A) is the prior probability of class.
- P (B|A) is the likelihood which is the probability of predictor given class.
- P (B) is the prior probability of predictor.

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and we need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Naive Bayes models:

- Gaussian: It is used in classification and it assumes that features follow a normal distribution.
- Multinomial: It is used for discrete counts. For example, let’s say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of “word occurring in the document”, we have “count how often word occurs in the document”, we can think of it as “number of times outcome number x_i is observed over the n trials”.

- Bernoulli: The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with ‘bag of words’ model where the 1s & 0s are “word occurs in the document” and “word does not occur in the document” respectively.

Here we will be using Multinomial Classification.

CODE

```
import numpy as np
import pandas as pd
import json, nltk
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import seaborn as sns
nltk.download('wordnet')   # for Lemmatization

total_data = pd.read_csv("dataset/train.csv", encoding="ISO-8859-1")
with open('assets/contractions.json', 'r') as f:
    contractions_dict = json.load(f)
contractions = contractions_dict['contractions']
pd.set_option('display.max_colwidth', -1)
total_data.head()
tweet = total_data.columns.values[2]
sentiment = total_data.columns.values[1]
tweet, sentiment
total_data.info()
def emoji(tweet):
    # Smile -- :), : ), :-), (:, ( :, (-:, :') , :O
    tweet = re.sub(r'(:\s?\)|:-\)|\(\s?:|\(-:|:\'\)|:O)', ' positiveemoji ', tweet)
    # Laugh -- :D, : D, :-D, xD, x-D, XD, X-D
    tweet = re.sub(r'(:\s?D|:-D|x-?D|X-?D)', ' positiveemoji ', tweet)
    # Love -- <3, :*
    tweet = re.sub(r'(<3|:\*)', ' positiveemoji ', tweet)
    # Wink -- ;-), ;), ;-D, ;D, (;,  (-; , @-)
    tweet = re.sub(r'(;-?\)|;-?D|\(-?;|@-\))', ' positiveemoji ', tweet)
    # Sad -- :-(, : (, :(, ):, )-:, :-/ , :-|
    tweet = re.sub(r'(:\s?\(|:-\(|\)\s?:|\)-:|:-/|:-\|)', ' negetiveemoji ', tweet)
    # Cry -- :,(, :'(, :"(
    tweet = re.sub(r'(:,\(|:\'\(|:"\()', ' negetiveemoji ', tweet)
    return tweet
import re
def process_tweet(tweet):
    tweet = tweet.lower()                                   # Lowercases the string
    tweet = re.sub('@[^\s]+', '', tweet)                       # Removes usernames
    tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', ' ', tweet)   # Remove URLs
    tweet = re.sub(r"\d+", " ", str(tweet))                      # Removes all digits
    tweet = re.sub('&quot;'," ", tweet)                         # Remove (&quot;)
    tweet = emoji(tweet)                                    # Replaces Emojis
    tweet = re.sub(r"\b[a-zA-Z]\b", "", str(tweet))                 # Removes all single characters
    for word in tweet.split():
        if word.lower() in contractions:
            tweet = tweet.replace(word, contractions[word.lower()])   # Replaces contractions
    tweet = re.sub(r"[^\w\s]", " ", str(tweet))                    # Removes all punctuations
    tweet = re.sub(r'(.)\1+', r'\1\1', tweet)                    # Convert more than 2 letter repetitions to 2 letter
    tweet = re.sub(r"\s+", " ", str(tweet))                      # Replaces double spaces with single space
    return tweet
total_data['processed_tweet'] = np.vectorize(process_tweet)(total_data[tweet])
total_data.head(10)
tokenized_tweet = total_data['processed_tweet'].apply(lambda x: x.split())
```

```
tokenized_tweet.head()
from nltk.stem.wordnet import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
tokenized_tweet = tokenized_tweet.apply(lambda x: [lemmatizer.lemmatize(i) for i in x])
tokenized_tweet.head()

stop_words = {"i", "me", "my", "myself", "we", "our", "ours", "ourselves",
            "you", "your", "yours", "yourself", "yourselves", "he", "him",
            "his", "himself", "she", "her", "hers", "herself", "it", "its",
            "itself", "they", "them", "their", "theirs", "themselves", "what",
            "which", "who", "whom", "this", "that", "these", "those", "am", "is",
            "are", "was", "were", "be", "been", "being", "have", "has", "had",
            "having", "do", "does", "did", "doing", "a", "an", "the", "and",
            "but", "if", "or", "because", "as", "until", "while", "of", "at",
            "by", "for", "with", "about", "against", "between", "into", "through",
            "during", "before", "after", "above", "below", "to", "from", "up",
            "down", "in", "out", "on", "off", "over", "under", "again", "further",
            "then", "once", "here", "there", "when", "where", "why", "how", "all",
            "any", "both", "each", "few", "more", "most", "other", "some", "such",
            "only", "own", "same", "so", "than", "too", "very",
            "can", "will", "just", "should", "now"}
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = ' '.join([word for word in tokenized_tweet[i] if word not in stop_words])

total_data['processed_tweet'] = tokenized_tweet
total_data.head()

sentiments = ['Positive Sentiment', 'Negetive Sentiment']
slices = [(total_data[sentiment] != 0).sum(), (total_data[sentiment] == 0).sum()]
colors = ['g', 'r']
plt.pie(slices, labels = sentiments, colors=colors, startangle=90, shadow = True,
        explode = (0, 0.1), radius = 1.5, autopct = '%1.2f%%')
plt.legend()
plt.show()

positive_words =' '.join([text for text in total_data['processed_tweet'][total_data[sentiment] == 1]])
wordcloud = WordCloud(width=800, height=500, random_state=21,
            max_font_size=110,background_color="rgba(255, 255, 255, 0)"
            , mode="RGBA").generate(positive_words)
plt.figure(dpi=600)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Most Used Positive Words")
plt.savefig('assets/positive_words.png')
plt.show()

negetive_words =' '.join([text for text in total_data['processed_tweet'][total_data[sentiment] == 0]])
wordcloud = WordCloud(width=800, height=500, random_state=21,
            max_font_size=110,background_color="rgba(255, 255, 255, 0)"
            , mode="RGBA").generate(negetive_words)
plt.figure(dpi=600)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("Most Used Negetive Words")
plt.savefig('assets/negetive_words.png')
plt.show()

from sklearn.feature_extraction.text import CountVectorizer
count_vectorizer = CountVectorizer(ngram_range=(1,2))    # Unigram and Bigram
final_vectorized_data = count_vectorizer.fit_transform(total_data['processed_tweet'])
final_vectorized_data

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(final_vectorized_data, total_data[sentiment],
                                                    test_size=0.2, random_state=69)
from sklearn.naive_bayes import MultinomialNB  # Naive Bayes Classifier

model_naive = MultinomialNB().fit(X_train, y_train)
predicted_naive = model_naive.predict(X_test)
from sklearn.metrics import confusion_matrix

plt.figure(dpi=600)
mat = confusion_matrix(y_test, predicted_naive)
sns.heatmap(mat.T, annot=True, fmt='d', cbar=False)

plt.title('Confusion Matrix for Naive Bayes')
plt.xlabel('true label')
plt.ylabel('predicted label')
plt.savefig("assets/confusion_matrix.png")
plt.show()
from sklearn.metrics import accuracy_score

score_naive = accuracy_score(predicted_naive, y_test)
print("Accuracy with Naive-bayes: ",score_naive)
from sklearn.metrics import classification_report
print(classification_report(y_test, predicted_naive))
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
probs = model_naive.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = roc_curve(y_test, preds)
roc_auc = auc(fpr, tpr)
plt.figure(dpi=600)                    # to plot high quality graph
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.ylabel('True Positive Rate', color='g')
plt.xlabel('False Positive Rate', color='r')
plt.savefig("assets/ROC_curve.png")
plt.show()
```

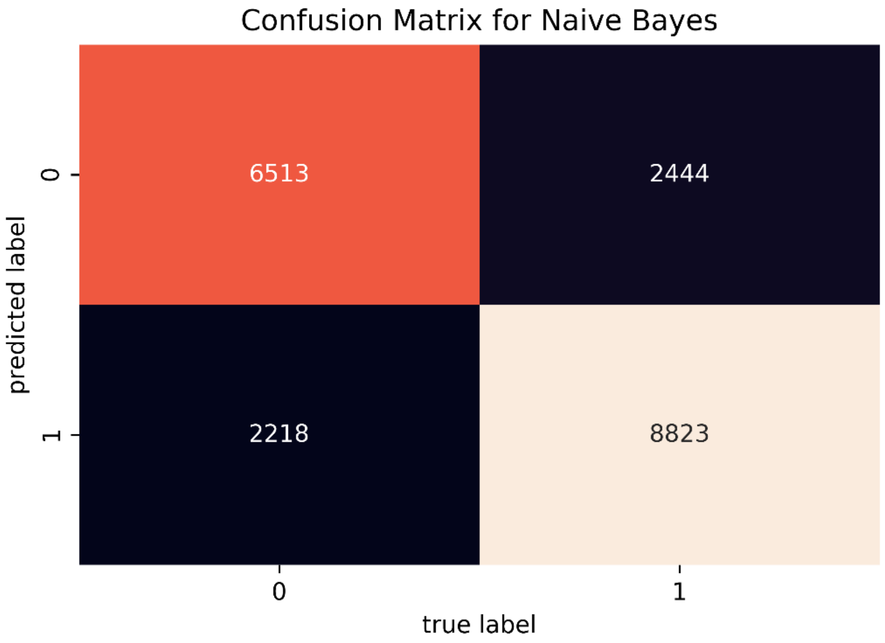
Precision Recall and Accuracy

Precision, recall, and accuracy are standard metrics used to evaluate the performance of a classifier.

- Precision measures how many texts were predicted correctly as belonging to a given category out of all of the texts that were predicted (correctly and incorrectly) as belonging to the category.
- Recall measures how many texts were predicted correctly as belonging to a given category out of all the texts that should have been predicted as belonging to the category. We also know that the more data we feed our classifiers with, the better recall will be.
- Accuracy measures how many texts were predicted correctly (both as belonging to a category and not belonging to the category) out of all of the texts in the corpus.

Most frequently, precision and recall are used to measure performance since accuracy alone does not say much about how good or bad a classifier is.

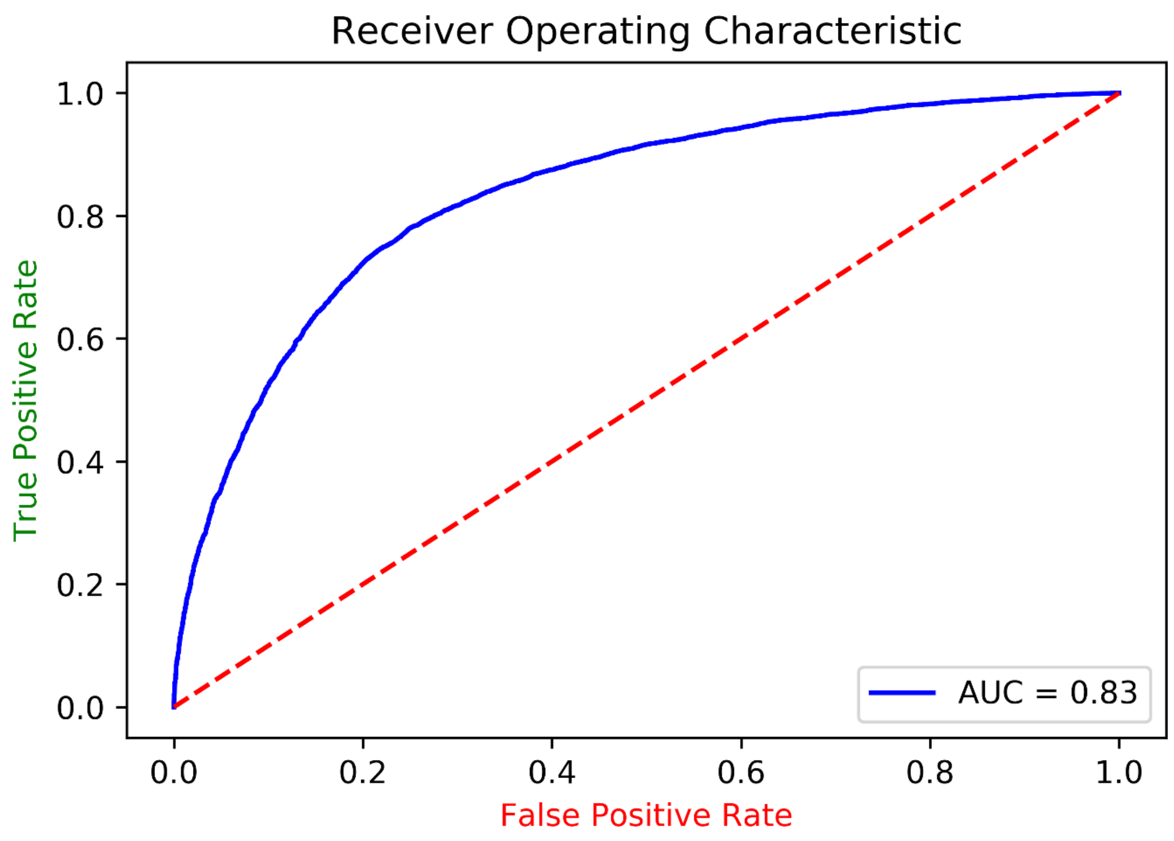
Confusion Matrix



Classification Report

	Precision	Recall	F1-score	Support
0	0.73	0.75	0.74	8731
1	0.80	0.78	0.79	11267
Accuracy			0.77	19998
Micro avg	0.76	0.76	0.76	19998
Weighted avg	0.77	0.77	0.77	19998

ROC Curve



Area Under the Curve of anything greater than 0.80 with real life data is not bad. We achieved 0.83.

Comparison (unigram only)

Normalization	Stop Words	Vectorizer	Accuracy
None	No	Count	0.7601260126012601
Lemmatizing	No	Count	0.757025702570257
Stemming	No	Count	0.755025502550255
Lemmatizing	Yes	Count	0.7483748374837483
None	No	Tf-Idf	0.7480748074807481
Stemming	Yes	Count	0.7477747774777478

Lemmatizing	Yes	Tf-Idf	0.7456245624562456
Stemming	No	Tf-Idf	0.7435743574357436
Lemmatizing	No	Tf-Idf	0.7403740374037404
Stemming	Yes	Tf-Idf	0.7383738373837384

So here we see the best combination for sentiment analysis with this dataset is using

- Without normalization
- Without stop words
- Countvectorizer

For document analysis the best combination will include lemmatization, stop words, but in sentiment analysis, they aren't providing an edge over the raw datas.

Accuracy (N-grams)

N-gram Methods	Accuracy
Unigram, Bigram and Trigram	0.76688
Unigram and Bigram	0.76788
Bigram and Trigram	0.72532

CONCLUSION

Achievements

The provided tweets were a mixture of words, emoticons, URLs, hashtags, user mentions, and symbols. Before training the dataset we pre-processed the tweets to make it suitable for feeding into models. We implemented machine learning algorithms Naive Bayes Classifier to classify the polarity of the tweet. We used three types of features namely unigrams, bigrams and trigrams for classification and observed that augmenting the feature vector with bigrams improved the accuracy and trigrams degraded the accuracy. Once the feature has been extracted it was represented as a sparse vector. It has been observed that presence in the sparse vector representation recorded a better performance than frequency. We finally achieved an accuracy of 76.78%

With unigram and bigram, without any normalization, stop words or tf-idf

Our model accuracy : 0.76788

Leaderboard highest : 0.79249

Future Directions

- Handling emotion ranges: We can improve and train our models to handle a range of sentiments. Tweets don’t always have positive or negative sentiment. At times they may have no sentiment i.e. neutral. Sentiment can also have gradations like the sentence, ‘This is good’, is positive but the sentence, ‘This is extraordinary’, is somewhat more positive than the first. We can therefore classify the sentiment in ranges, say from -2 to +2.
- Using hashtags and other symbols like ‘?’ and ‘!’: During our pre-processing, we discard most of the symbols like hashtags, question marks and exclamation marks. These symbols may be helpful in assigning sentiment to a sentence.

Text Processing and Sentiment analysis emerges as a challenging field with lots of obstacles as it involves natural language processing. It has a wide variety of applications that could benefit from its results, such as news analytics, marketing, question answering, readers do. Getting important insights from opinions expressed on the internet especially from social media blogs is vital for many companies and institutions, whether it is in terms of product feedback, public mood, or investors opinions.

Sentiment analysis is a difficult technology to get right. However, when you do, the benefits are great.

Look for a tool that has uses Natural Language Processing technology and ideally with machine learning capabilities. Look for a vendor that treats sentiment analysis seriously and shows advancements and updates in their sentiment analysis technology.

Sentiment Analysis is an interesting way to think about the applicability of Natural Language Processing in making automated conclusions about text. It is being utilized in social media trend analysis and, sometimes, for marketing purposes. Making a Sentiment Analysis program in Python is not a difficult task, thanks to modern-day, ready-for-use libraries. This program is a simple explanation to how this kind of application works. This is only for academic purposes, as the program described here is by no means production-level.

While it’s difficult to speculate how a relatively immature system might evolve in the future, there is a general assumption that sentiment analysis needs to move beyond a one-dimensional positive to negative scale.

The Future of Sentiment Analysis

In the same way that politics cannot always be reduced to a position on a left to right scale, there are other kinds of sentiment that cannot be placed on a simple barometer.

For the future, to truly understand and capture the broad range of emotions that humans express as written words, we need a more sophisticated multidimensional scale.

Can text analytics measure skepticism, hope, anxiety, excitement or lack thereof? Yes! And we’ll be making this technology available soon.

Organisations will certainly become more aware of the applications of sentiment analysis within their marketplace, fuelling the growth of sector specific services and technology delivering sentiment specific use cases – for example, intelligence tools that aid decision-making for financial traders and analysts.

We will see a shift in perception of the reliability of sentiment analysis. Users will become more comfortable with the idea that the automatic analysis of individual text material is hard to match human performance.

The insight that can be gained from large datasets (millions of Tweets) will overshadow the concerns about reliability at a granular level (a single Tweet).

Instead, the focus will be on how to make results interpretable and actionable. In the meantime, we’ll be ensuring we are working at making sentiment analysis as accurate and easy to understand as possible.

Bibliography

- Towards Data Science
- Analytics Vidya
- Project report on Github
- Medium
- Stackabuse
- Geeksforgeeks
- Kaggle
- Stack Overflow
- Stack Exchange
- Monkey Learn

Full project on GitHub