2021-2022

CS F469

Information Retrieval

Twitter Sentiment Polarity Analysis using Naive Bayes

Aadit Deshpande, 2019A7PS0077P
Apoorv Badar, 2019A7PS0060P
Suchismita Tripathy, 2019A7PS0554P

29 April 2022

Allotted Application Domain: Evaluation of public/voters' opinion
Approved Paper: N. Yadav, O. Kudale, S. Gupta, A. Rao, and A. Shitole, *"Twitter Sentiment Analysis Using Machine Learning For Product Evaluation,"* 2020 International Conference on Inventive Computation Technologies (ICICT), 2020, pp. 181-185.

Table of Contents

1. Problem Statement

Sentiment Polarity Classification on Tweets in the “Sentiment140” Kaggle dataset using supervised learning methods like Multinomial Naive Bayes and Bernoulli Naive Bayes.

2. Background

The internet and web are unprecedented sources of information in many ways. The most crucial distinction is that content creation is decentralized. This democratization has led to vast amounts of publicly available user-created content based on the users’ opinions and interests. This new level of granularity in the opinion, whether ‘positive/negative,’ ‘truth/lies,’ ‘contradictions/assumptions,’ and so on, has given rise to the research area of Sentiment Analysis.

In this context, Sentiment Analysis refers to inferring people’s opinions, attitudes, and emotions regarding various topics expressed through written text. Sentiment Analysis tasks include classification of sentiment polarity expressed in the text (positive, negative, or neutral), opinion holder identification, and identifying sentiment for various aspects of a topic. This field of study has many commercial applications like predicting product sales, stock returns, and the outcomes of political elections.

As the importance of internet Social media platforms grows, Twitter has emerged as the subject of most sentiment analysis studies, as tweets provide valuable insight into users’ opinions. As of 2022, Twitter has over 217 million monetizable daily active users, with 500 million tweets shared daily. Most of Twitter’s audience is 25-34 years old, making it an interesting case study of a far-reaching social network. Twitter Sentiment Analysis (TSA) is a specialized area within sentiment analysis and is a prominent research topic in computational linguistics.

In this report, we perform TSA on the “Sentiment140” dataset from Kaggle, with 1.6 million tweets (800K Positive and 800K Negative tweets). We follow the supervised learning approach - Naive Bayes for Sentiment Polarity Classification and then compare the results of the different models used. The report is structured as follows: Sections 3 and 4 discuss the existing literature and current research gaps. In sections 5 and 6, we describe the methods used for the sentiment analysis and their evaluation. Finally, in sections 7 and 8, we discuss the results, draw conclusions and briefly look at avenues for future research.

3. Related Work

Sentiment Analysis work was pioneered by Pang et al. [5] when they used three supervised ML methods (Naive Bayes, Maximum entropy classification, Support Vector Machines) to classify movie reviews as positive or negative. Since then, sentiment analysis has found applications in many different types of problems like real-time opinion mining, marketing, healthcare, etc.

Specifically, Stock prediction systems via market sentiment analysis are of economic importance. Li et al. [6], in their 2020 paper, explored this concept using a long short-term memory (LSTM) neural network-based stock prediction model on textual news articles. Xing et al. [7] discussed the role of market sentiment in an asset allocation problem. They used an ensemble learning approach that combined an evolving clustering algorithm with an LSTM neural network to compute sentiment time series and profitability.

Sentiment analysis is also applicable to multilingual language content. Araújo et al. [8] discuss the vast potential for sentiment analysis in different languages such as Arabic, Portuguese, Spanish, and German. They propose machine translation-based techniques as a baseline for new multilingual sentiment analysis methods.

With the exponential growth of social media, there has been a rising interest in semi-supervised learning methods to exploit the enormous amounts of available unlabeled data. To this end, Fu et al. [9] showed Variational Autoencoders as an effective semi-supervised approach for Aspect-term Sentiment Analysis.

TSA (Twitter Sentiment Analysis) has become an NLP research hotspot for Social Media sentiment analysis. One of the significant challenges of text classification for short texts (such as tweets) is the choice of feature representation.

The most widely-used and straightforward approach is N-gram features to represent tweets. They've been used in studies such as the paper by Miranda-Jiménez et al.[10] on ensemble-based TSA. An alternative to n-grams is to use meta-level features, including summations and counts of part-of-speech words, which have given a slightly improved F-score for positive, negative, and neutral classes.

However, n-grams are a sparse representation of tweets and hence suffer from the curse of dimensionality. With recent advances in Deep Learning, highly dense vector representations have produced better results in NLP tasks. Agrawal et al. [11] use LSTMs to train Emotion Word Embeddings to address words that appear in similar contexts but with different polarities. In their study, Ma et al. [2] used LSTMs consisting of target-level and sequence-level attention for Aspect-based sentiment analysis.

Finally, Combination strategies have also proven to be a very effective approach for Sentiment Analysis. Carvalho et al. [1] discuss a majority voting strategy by combining classifiers. The classifier ensemble was formed by Multinomial Naive Bayes, Support Vector Machines, Random Forest, and Logistic Regression, using the Bag of Words Model.

4. Research Gap

Having seen the current state of research in this domain, we must also briefly discuss the major challenges (Kharde et al. [4]). Firstly, Domain Dependence means that the same phrase can have different meanings in different contexts. A major issue, especially in TSA, is Sarcasm

Detection, wherein sentences use positive words but ultimately express negative opinions about a target. Another issue for ML methods that use the bag-of-words model is Order Detection. Sentences like “A is better than B” that are sensitive to order, lose their meaning in such approaches. Finally, many forms of Negation are not Explicit. The bag-of-words model would have difficulty inferring negative sentiment in a sentence like “hardly any good films.”

Twitter Sentiment Analysis also poses its own unique challenges (Zimbra et al. [3]). First, if the approach uses N-grams, it suffers forms extremely sparse feature representations and the curse of dimensionality [1]. Second, Twitter has frequent use of slang and acronyms prevalent in social media, making the language difficult to analyze. Furthermore, the constraints on Tweet lengths and frequent use of emojis affect TSA performance. Thirdly, the extreme Sentiment imbalance, (such as overwhelmingly positive or negative reviews) affects the quality of the training. Finally, tweets show temporal dependency, which is not captured in most ML methods.

5. System Description

5.1 Flowchart: The following describes the methodology followed by us for the current task:

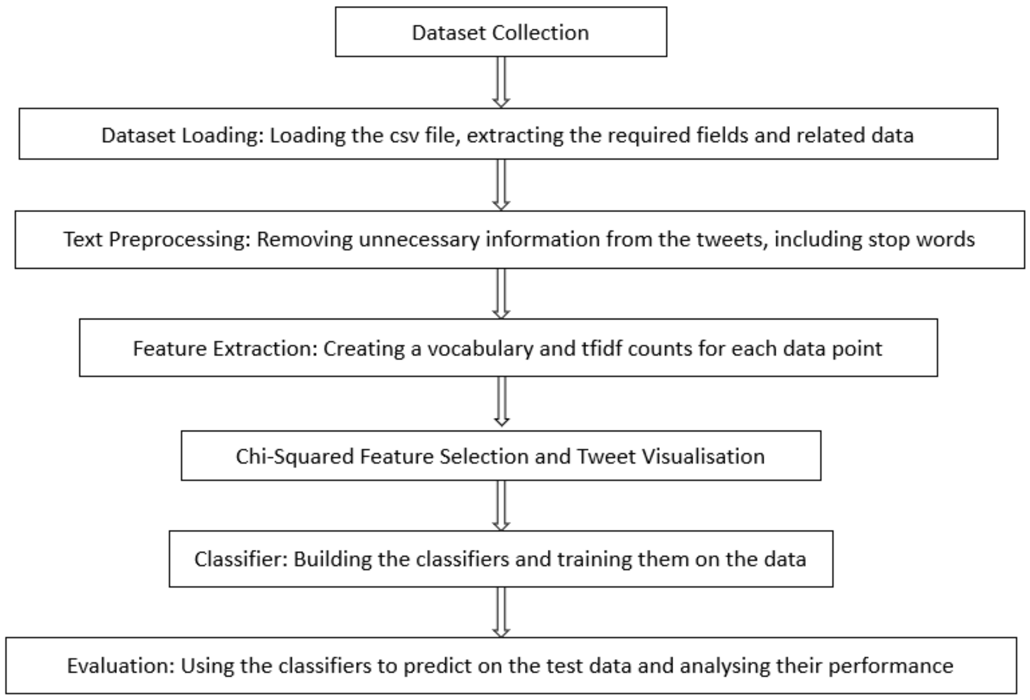

Dataset Collection
Dataset Loading: Loading the csv file, extracting the required fields and related data
Text Preprocessing: Removing unnecessary information from the tweets, including stop words
Feature Extraction: Creating a vocabulary and tfidf counts for each data point
Chi-Squared Feature Selection and Tweet Visualisation
Classifier: Building the classifiers and training them on the data
Evaluation: Using the classifiers to predict on the test data and analysing their performance

Fig. 1 | Methodology

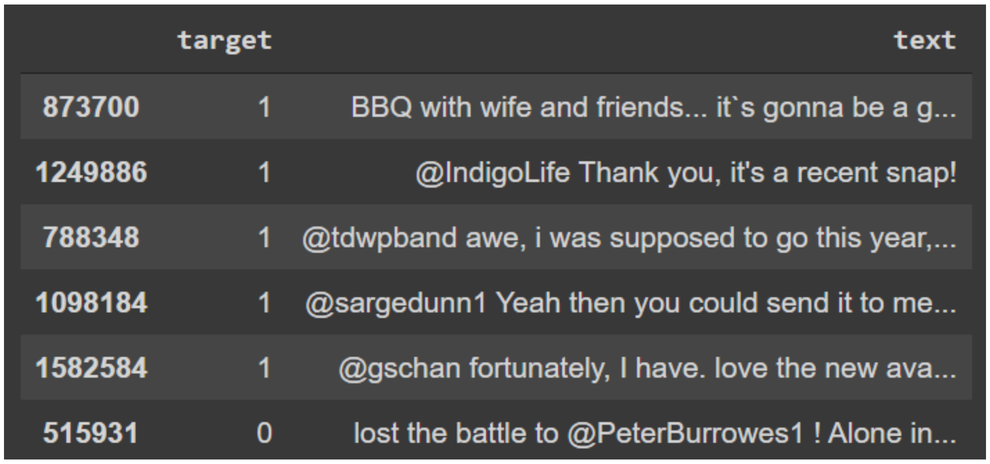
5.2 Dataset: The dataset used, as mentioned, earlier, is the Sentiment140 dataset available through Kaggle. It contains a total of 1600000 tweets, each with a specific sentiment label: **0 for negative, and 4 for positive**. The original dataset is perfectly balanced i.e. 800000 tweets labeled negative and 800000 tweets labeled positive.

Through a non-exhaustive observation of the tweets and labels, it was found that a few tweets were labeled incorrectly. Hence, it was decided to recalculate the sentiment labels of each tweet before moving on with the classification system. For this, after the data (as a CSV file) was loaded onto a data frame, the tweets were first preprocessed. The preprocessing is the same as that used for the actual classification system and is explained below. The VADER (Valence Aware Dictionary and sEntiment Reasoner) Tool was used for analyzing the processed tweets. This tool is a lexicon and rule-based sentiment analyzer (as opposed to a clustering tool) and is known to work well, particularly on social media text. The tool's SentimentIntensityAnalyser was used and tweets were labeled with 0 if the compound polarity score was negative and with 4 if the compound polarity score was positive.

This modified CSV file (**Fig 2.**) was then used as the base dataset. The labels were changed to 0 and 1 for convenience and the split-up is as follows:

Negative (Label : 0)	Positive (Label: 1)	Total
379526	1220476	1600000

Table 1 | Dataset split-up



```
         target                                         text
873700        1   BBQ with wife and friends... it`s gonna be a g...
1249886       1            @IndigoLife Thank you, it's a recent snap!
788348        1  @tdwpband awe, i was supposed to go this year,...
1098184       1   @sargedunn1 Yeah then you could send it to me...
1582584       1      @gschan fortunately, I have. love the new ava...
515931        0    lost the battle to @PeterBurrowes1 ! Alone in...
```

Fig. 2 | Dataframe obtained from the modified 'Sentiment140' dataset

5.3 Text Preprocessing:

Since the text, we are working with is in the form of tweets, the first step of preprocessing was to remove all tags and URLs from the text.This was done with the appropriate regular expressions for the same , ((http://)[^]*|(https://)[^]*|(www\.)[^]*) and "@[^\s]+" respectively.The tweets also contain repetitions of certain letters in words such as 'heyyyy'.To correct this, three or more repetitions of the same letter were replaced with two occurrences of the letter. Finally, special characters such as hashtags which occur very frequently in tweets were removed, retaining the words accompanying them since they frequently refer to the topic of the tweet, e.g. '#happy'.

Finally, the stop words were removed from the tweets using the NLTK stop words collection for English. Each tweet was split into words, and each word was then compared with the stop words to check if it should be removed. A total of 179 stop words were used, which included common prepositions, pronouns, common contractions, etc.

Alternately, **tags and URLs** could have been encoded and added to the downloaded **stop words** collection and then applied to the text in one move. This is similar to creating a custom stop words list, as stop words vary from application to application.

The **pseudocode** for the same is

1. Loop through the tweets.
2. Remove the tag and urls using the appropriate regular expressions.
3. Remove the special characters (non-alphanumeric).
4. Remove the repetition of the characters.
5. Split the tweet into words and discard the stop words.
6. Finally, join all the words back into sentence form by concatenating with white space for each tweet.

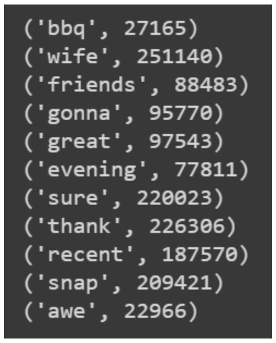
5.4 Feature Extraction: The first step here was to build a vocabulary. The pseudocode for this is as follows:

1. Loop through the tweets.
2. Split the tweet into words using white space as the delimiter.
3. Check if each word is already a part of the unique list of words that forms the vocabulary. If yes, ignore it. Else add it to the list.

The second step was to then create **TF-IDF vectors** for each tweet. The **pseudocode** for this would be

1. Loop through the tweets.
2. For each tweet, create lists of lengths equal to the length of the vocabulary.
3. For each word in the tweet, record its count and save it in the right position in the above list. This is the term frequency for a given term in the given document.
4. Calculate document frequency for each term i.e. the number of documents the word appears in. Using this and the total number of documents, calculate the inverse document frequency for each term.
5. Multiply each tf value for each tweet with the corresponding idf value.

For our dataset, instead of using the above algorithms, the CountVectorizer and Tfidfvectorizer tools from sklearn were used to complete both steps. The vocabulary was built on the complete set of tweets using the “fit” function available. This created a **vocabulary of size 260131**, where each unique word was given a unique term ID. The following shows a subset of the vocabulary built (**Fig 3.**).



```
('bbq', 27165)
('wife', 251140)
('friends', 88483)
('gonna', 95770)
('great', 97543)
('evening', 77811)
('sure', 220023)
('thank', 226306)
('recent', 187570)
('snap', 209421)
('awe', 22966)
```

Fig. 3 | Vocabulary terms and their respective tf-idf values

The vectors were then created using the "transform" function available.
The CountVectorizer simply created vectors of the same length as the vocabulary size, with each entry specifying the count of that term in the current tweet (same as the term frequency or *tf(t, d)*). The TfidfVectorizer calculated tf-idf values as follows:

The TF-IDF values for each word are just a multiplication of the tf and idf values:

$$tf - idf(t,d) = tf(t,d) \times idf(t)$$

The idf terms are normally calculated as follows:

$$\boxed{\begin{array}{l} idf(t,d) = log\frac{n}{1+df(t)} \\ \text{where} \\ n = \text{Number of documents (in our case tweets)} \\ df(t) = \text{Number of documents (tweets) that term } t \text{ appears in at least once} \end{array}}$$

The Tfidfvectorizer tool used, however, calculates the idf values as follows by default:

$$idf(t,d) = log\frac{1+n}{1+df(t)} + 1$$

Where 1 is added as a smoothing constant. This vectorization process thus results in a matrix of size (1600000, 260131) in both cases. The vectors are then normalized by the Euclidean norm and stored in Compressed Sparse Row format. Compressed Sparse Row format saves a 2D array in the form of 3 1D arrays:

1. Non-zero values in the 2D array
2. Corresponding column indices
3. Corresponding row numbers

5.5 Feature Selection:

The **χ^2 test** is used as a measure of the independence of two events. For feature selection, the two events are the occurrence of the term and the occurrence of the class. It is calculated as follows:

$$X^2(D,w,c) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_w e_c} - E_{e_w e_c})^2}{E_{e_w e_c}}$$

where N is the observed frequency in text D and E is the expected frequency which is calculated assuming the term and class to be independent. A small χ^2 value means that the observed and expected frequencies are close i.e. the term and class are independent. Thus, a high χ^2 value indicates that the independence between the term and the class is low i.e. it has good discriminating power. So, to select the appropriate feature as well as reduce the size of our feature which was 260131 (size of the vocabulary), we **select the 3000 best terms** with the highest χ^2 values.

The pseudo-code for the same is

1. Calculate the frequency of each word in the class by looping through the text.
2. Calculate the expected frequency by multiplying the probability of the word in the dataset and multiplying by the total number of the class's texts.
3. Calculate the χ^2 value using the formula given above.
4. Finally, take the k words with the highest χ^2 values since they have the highest confidence of association.

For our dataset, instead of using the above algorithm, we used the "selectKBest" and "chi2" tools from the sklearn library.

5.6 Tweet Visualisation:

The tweets in the dataset are represented as word embeddings in the form of Tf-Idf vectors. However, this vector space is extremely sparse high-dimensional (260,131 components). Hence, we perform dimensionality reduction using ***t-distributed stochastic neighbor embedding (t-SNE)*** to visualize the high dimensional data points in 2D or 3D space. The steps we followed to visualize using t-SNE are as follows:

1. We perform random shuffling on the dataset to get an even distribution of 0/1 labels.
2. TSNE on 1.6 million tweets is infeasible! So, we consider subsets of the dataset and plot the tweets at different subset sizes S (100, 1000, 5000 tweets, etc.).
3. The tweets are represented using a Scatter plot, with green points (Positive Tweets), and red points (Negative Tweets).
4. We use the sklearn.manifold library's implementation of t-SNE with the following parameters:
    a. N_components = 2 (for the 2D plot) or (3 for the 3-D plot)
    b. Iterations = 1000
    c. Perplexity = 30 nearest neighbors
    d. Initialization = Random Clusters
    e. Metric = Euclidean Distance

From the 2D plots in **Fig. 5** we can make some useful observations. When the sample space is small (100 tweets), the tweets seem evenly distributed. As we increase the subset size to 500 and 1000 tweets, we can observe a central cluster forming with both positive and negative points. At 2000 tweets, we can clearly see that the outliers are dominated by red points (Negative). The central cluster grows even larger and encompasses almost all the points at 5000, 8000 tweets, with a few red outliers. These observations hold in 3-D space too (**Fig. 4**).

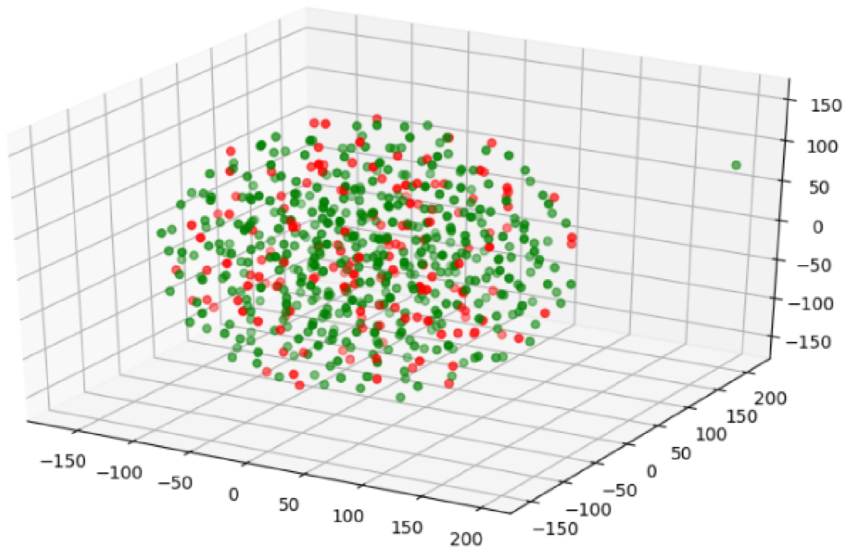


Fig. 4 | Tweet Representation in 3D Space (S = 600 tweets)

5.7 Classification using Naive Bayes:
Before creating the classifiers, the tweets and corresponding labels were split into train and test sets, with the test set being 10% of the total i.e. 160,000 data points.

Both Multinomial and Bernoulli models were used and their performance was compared. Tools for both from sklearn were used. Naive Bayes calculates the probability that a particular document belongs to a particular class given the set of words that appear in that document (and their tf-idf values) i.e. the posterior probability.

Starting with the following conditional probability where *i* represents a term in the document and y represents a class, the following is obtained:

$$P(x_1, x_2, ...x_n|y) = \frac{P(x_1,x_2,...x_n \cap y)}{P(y)}$$

Taking the terms in the reverse order i.e. for the posterior probability:

$$P(y|x_1, x_2, ...x_n) = \frac{P(x_1,x_2,...x_n \cap y)}{P(x_1,x_2,...x_n)}$$

With the conditional independence assumption (assuming that the attribute values are independent of each other given the class), we get:

$$P(y|x_1, x_2, ...x_n) = \frac{P(x_1 \cap y)P(x_2 \cap y)...P(x_n \cap y)}{P(x_1,x_2,...x_n)}$$

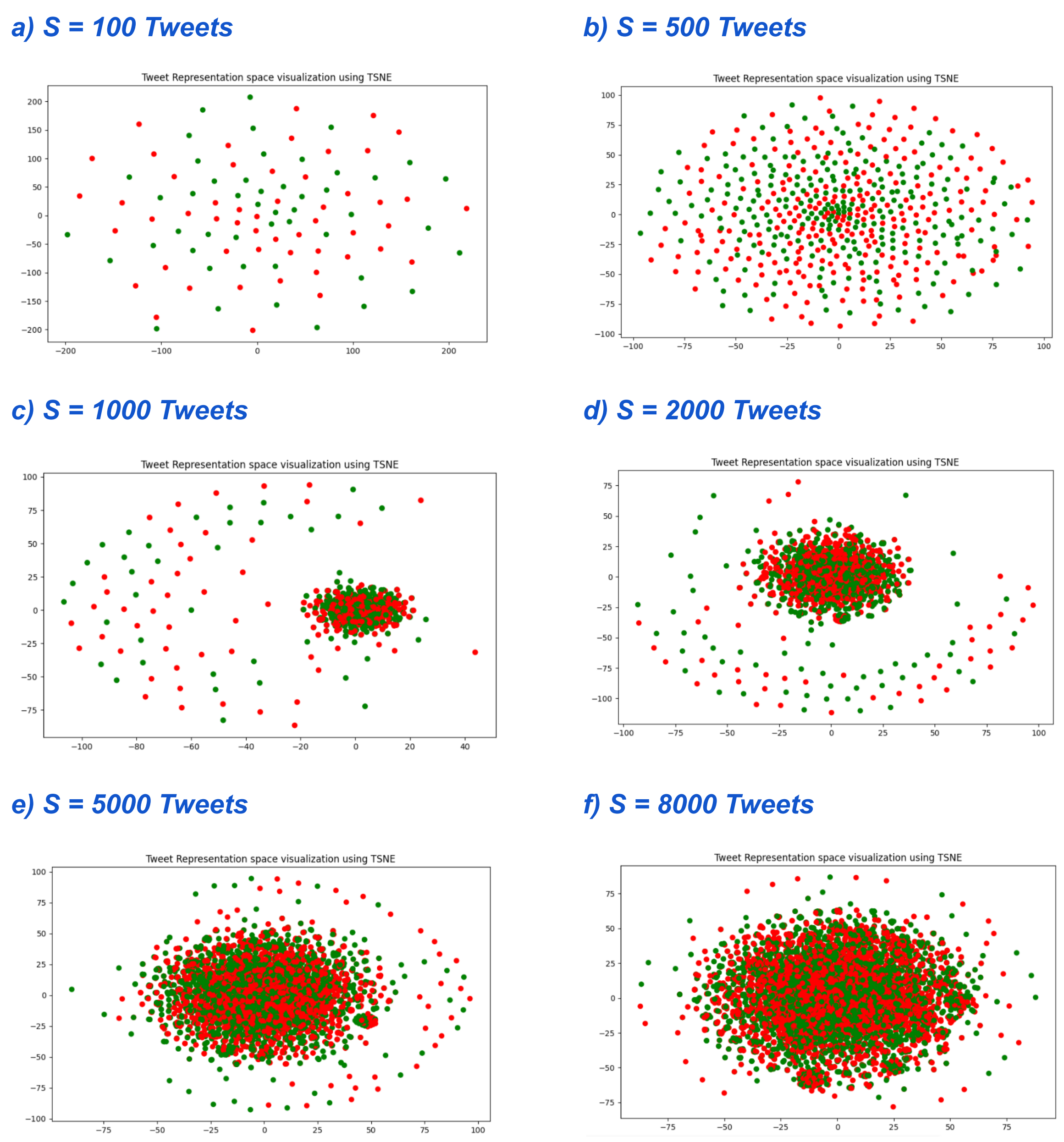


Fig. 5 | Tweet Representation in 2D Space as a function of subset size S

Using the conditional probability of term *i,* given class *y*:

$$P(x_i \cap y) = P(x_i|y)P(y) \qquad P(y|x_1, x_2, ...x_n) = \frac{P(y)\prod P(x_i|y)}{P(x_1,x_2,...x_n)}$$

As the denominator is a constant for a particular document, we can use:

$$P(y|x_1, x_2, ...x_n) \propto P(y)\prod P(x_i|y) \qquad \hat{y} = argmax_y P(y)\prod P(x_i|y)$$

1. **Multinomial Naive Bayes:**
   Since the true parameters are not known, MLE (Maximum Likelihood Estimates) of the probabilities are used in the calculations instead.

$$\hat{P}(y) = \frac{N_y}{N}$$

$$\hat{P}(x_i|y) = \frac{N_{yi}+\alpha}{N_y+(\alpha \times n)}$$

where

$N_{yi} =$ Number of times feature i appears in a sample of class y

$N_y = \sum N_{yi}$ i.e total count of features fo class y

α : Smoothing constant

When $\alpha \ = 1$, it is called Laplace smoothing and this is the default value

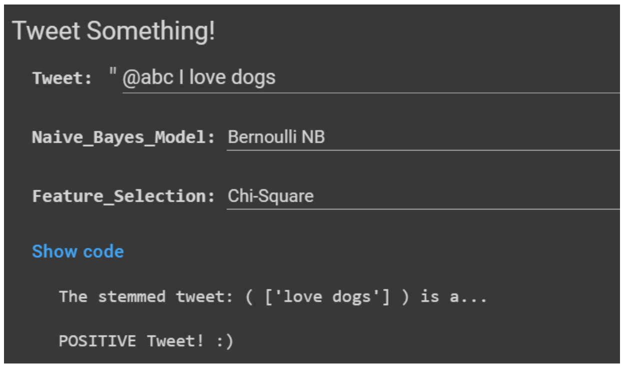
2. **Bernoulli Naive Bayes:**
   Here, the values for each feature are assumed to be binary i.e. 0 or 1 only, which is applied to our features matrix by binarizing the 2D array. Here, the probability of term *i* given class *y* is defined as:

$$P(x_i|y) = P(i|y) \times x_i + (1 - P(i|y)) \times (1 - x_i)$$

5.8 Prediction Tool:

A simple prediction tool was built that takes a tweet as user input, a choice of classifier (either Multinomial Naive Bayes or Bernoulli Naive Bayes), and a choice for feature selection (either further feature selection is not applied or Chi-Square feature selection is used). The tool then predicts the class of the tweet, based on the inputs. The GUI was built using python itself, and its functioning is shown in **Fig. 6.**

a) Positive tweet



Tweet Something!

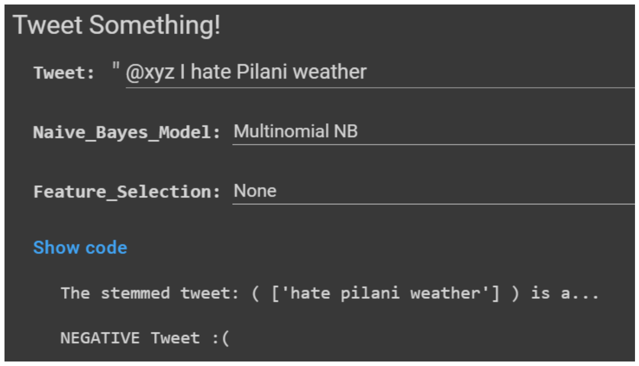
Tweet: " @abc I love dogs

Naive_Bayes_Model: Bernoulli NB

Feature_Selection: Chi-Square

```
The stemmed tweet: ( ['love dogs'] ) is a...
POSITIVE Tweet! :)
```

b) Negative tweet:



Tweet Something!

Tweet: " @xyz I hate Pilani weather

Naive_Bayes_Model: Multinomial NB

Feature_Selection: None

```
The stemmed tweet: ( ['hate pilani weather'] ) is a...
NEGATIVE Tweet :(
```

Fig. 6 | Prediction Tool tested on representative Positive and Negative Tweets

6. Evaluation Strategy

While accuracy can be a good metric for the evaluation of a classifier in the case of a balanced dataset, it is not an appropriate metric in the case of skewed datasets since it can be dominated by the most frequently occurring classes. In such cases, precision, recall, and f-score are the appropriate metrics.

Precision is the fraction of the correct predictions i.e. it tells us about the ability or confidence of the model to predict only the correct data points. The recall is the fraction of the class elements correctly identified i.e. it tells us about the ability of the model to find all the relevant points of the dataset. F-score is a metric that combines precision and recall, thus helping capture both the ability of the model to identify the relevant cases (recall) and the ability to correctly identify (precision).

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

$$F - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

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

Finally, we also plot the confusion matrix, which gives the distribution of the predicted sentiment against the actual sentiment (i.e. the number of true positives, true negatives, false positives, and false negatives).

		TP	FP	TN	FN	Precision	Recall	F-Score	Accuracy
a)	Multinomial	121,369	26,370	11,574	687	0.822	0.994	0.900	0.831
b)	Multinomial + Chi-Square	121,588	25,053	12,891	468	0.829	0.996	0.905	0.840
c)	Bernoulli	121,737	36,958	986	319	0.767	0.998	0.867	0.767
d)	Bernoulli + Chi-Square	121,780	36,888	1,056	276	0.768	0.998	0.868	0.768

Table 2 | Evaluation results for all 4 NB models

7. Experimental Results

Table 2 shows the Evaluation metrics for all 4 variations of Naive Bayes models described in section 5. The metrics used were: Precision, Recall F-Score, and Accuracy.

Table 2 rows **a)** and **c)** show that Multinomial NB outperforms Bernoulli NB in Precision, Accuracy, and F-score. However, Bernoulli NB however has slightly better Recall than Multinomial. For both models, the number of True Negatives (~ 12,000) is much greater than the number of False Negatives (~500). Multinomial NB has fewer false positives and false negatives than Bernoulli. The better performance of Multinomial could be attributed to the fact that it takes into account multiple occurrences of terms, a fact that is commonly observed in tweets.

Chi-Square Feature Selection benefits both models, as shown in Table 2 rows **b)** and **d)**. Multinomial NB has a 1% increase in Precision and Recall and only a 0.6% increase in F-score. Bernoulli NB experiences only a 0.1% increase in Precision, F-score, and Accuracy, but no change in Recall. Due to the high dimensionality of the data, Feature selection is more helpful for Multinomial NB.

Finally, we must also discuss some limitations of this study. Firstly, the experiment is constrained by the run-time of the models and the size of the dataset. The bottleneck steps are performing Stemming, tf-idf Vectorizing, and t-SNE on the 1.6 million tweets. Secondly, Bernoulli NB might predict all labels as 1, thereby increasing the recall to 100% despite poor performance. Finally, the quality of predicted polarities will only be as good as the labeling of the dataset itself.

8. Conclusion and Future Work

In this study, we performed Twitter Sentiment Analysis (TSA) on the “Sentiment140” Kaggle dataset with 1.6 million tweets. First, we discussed the current state of Sentiment Analysis research, specifically some effective TSA approaches involving LSTMs. We also discussed the research gap and some TSA issues like Sarcasm detection and order detection.

We then proceeded to describe the architecture of our Naive Bayes (NB) system. As part of preprocessing, we performed Stemming and removal of URLs. The next step was Feature extraction where extracted the tf-idf vectors with (260,131 components) from the tweets. We used 2 models of NB, namely, Multinomial NB (tf-idf vectors) and Bernoulli NB (count vectors). Further, we also performed Chi-squared feature extraction and reduced the vector components from 260,131 to 3,000. Additionally, we visualized the tweet vector representation-space in 2D and 3D using t-SNE.

We evaluated the performance of the 4 models (Multinomial, Bernoulli with and without Chi-Squared feature extraction) on Precision, Recall, F-score, and Accuracy. Multinomial NB outperformed Bernoulli NB on all measures except recall. Feature Selection showed greater benefit for the Multinomial model.

Finally, we discussed the limitations of this study, which included the bottleneck steps for the system’s run-time, the 100% recall issue with Bernoulli NB, and the dataset quality. Improving performance on these criteria could be a viable direction for future research. Future work in TSA could also focus on dealing with more complex NLP tasks such as Sarcasm Detection, Subjectivity and Context, handling Emojis in tweets, etc. Last but not least, integrating deep learning approaches such as LSTMs could tackle the curse of dimensionality and significantly improve linear classifiers like Naive Bayes.

9. References

1 Carvalho, Jonnathan and Alexandre Plastino. “On the evaluation and combination of state-of-the-art features in Twitter sentiment analysis.” Artif. Intell. Rev. 54 (2021): 1887-1936.

2 Ma, Y., Peng, H. and Cambria, E. 2018. Targeted Aspect-Based Sentiment Analysis via Embedding Commonsense Knowledge into an Attentive LSTM. Proceedings of the AAAI Conference on Artificial Intelligence. 32, 1 (Apr. 2018).

3

David Zimbra, Ahmed Abbasi, Daniel Zeng, and Hsinchun Chen. 2018. The State-of-the-Art in Twitter Sentiment Analysis: A Review and Benchmark Evaluation. ACM Trans. Manage. Inf. Syst. 9, 2, Article 5 (June 2018), 29 pages. DOI:https://doi.org/10.1145/3185045

4 Kharde, Vishal, and Prof Sonawane. "Sentiment analysis of Twitter data: a survey of techniques." arXiv preprint arXiv:1601.06971 (2016).

5 Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), pages 79–86. Association for Computational Linguistics.

6 Li, Xiaodong et al. “Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong.” Inf. Process. Manag. 57 (2020): 102212.

7 Frank Z. Xing, Erik Cambria, and Roy E. Welsch. 2018. Intelligent Asset Allocation via Market Sentiment Views. <i>Comp. Intell. Mag.</i> 13, 4 (Nov. 2018), 25–34. DOI:https://doi.org/10.1109/MCI.2018.2866727

8 Araújo, Matheus and Matheus Lima Diniz. “A comparative study of machine translation for multilingual sentence-level sentiment analysis.” Inf. Sci. 512 (2020): 1078-1102.

9 Fu X, Wei Y, Xu F, Wang T, Lu Y, Li J, Huang JZ (2019) Semi-supervised aspect-level sentiment classifcation model based on variational autoencoder. Knowl Based Syst 171:81–92

10 [INGEOTEC at SemEval 2017 Task 4: A B4MSA Ensemble based on Genetic Programming for Twitter Sentiment Analysis](https://aclanthology.org/S17-2130) (Miranda-Jiménez et al., SemEval 2017)

11 Ameeta Agrawal, Aijun An, and Manos Papagelis. 2018. Learning Emotion-enriched Word Representations. In Proceedings of the 27th International Conference on Computational Linguistics, pages 950–961, Santa Fe, New Mexico, USA. Association for Computational Linguistics.