**Sentiment Analysis of Twitter Data**

Aatish Suman Abhiraj Singh

asuman@syr.edu asingh73@syr.edu

School of Information Studies

Syracuse University

Syracuse, NY 13210 USA

**Abstract**

The purpose of this report is to present the results of sentiment analysis performed on Twitter data using multiple algorithms and discuss the suitability of the algorithms for the task. The sentiment140 dataset containing 1.6 million tweets, each containing characters less than or equal to 140, was analysed using five traditional machine learning algorithms with different vectorization and feature selection methods, and their performances were evaluated. Logistic regression provides the best results amongst the algorithms used which indicates that the advanced deep neural network algorithms are likely to provide superior results. Addition of bigrams in almost all the cases, improves the performance of the algorithms.

**Introduction**

Sentiment Analysis (SA) or Opinion Mining (OM) is the computational study of people’s sentiment toward an entity. The entity can represent individuals, events or topics. One of the challenges of Sentiment Analysis is defining the meaning of “sentiment”. To underline the ambiguity of the concept, Pang and Lee (Pang and Lee, 2008) list the definitions of terms closely linked to the notion of sentiment –

* Opinion implies a conclusion thought out yet open to dispute (“each expert seemed to have a different opinion”).
* View suggests a subjective opinion (“very assertive in stating his views”).
* Belief implies often deliberate acceptance and intellectual assent (“a firm belief in her party’s platform”).
* Conviction applies to a party’s firmly and seriously held belief (“the conviction that animal life is as sacred as human”).
* Persuasion suggests a belief grounded on assurance (as by evidence) of its truth (“was of the persuasion that everything changes”).
* Sentiment suggests a settled opinion reflective of one’s feelings (“her feminist sentiments are well-known”).

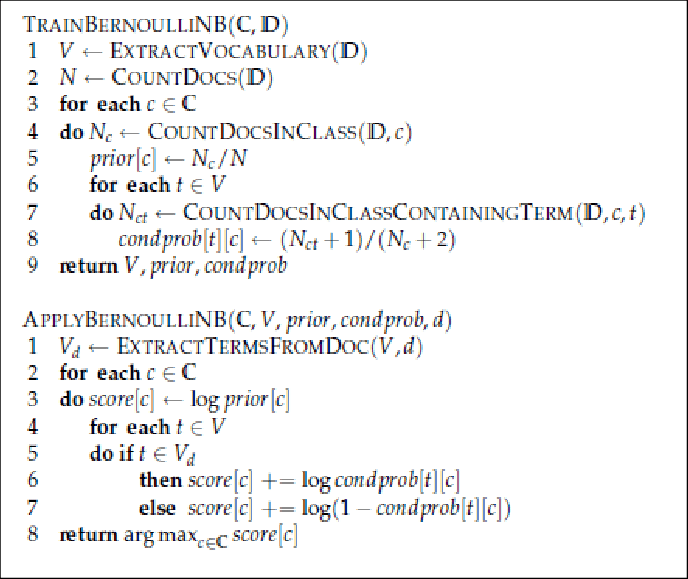
Sentiment Analysis can be considered a classification process. There are three main classification levels in SA: document-level, sentence-level, and aspect-level SA. Document-level SA aims to classify an opinion document as expressing a positive or negative opinion or sentiment. It considers the whole document a basic information unit (talking about one topic). Sentence-level SA aims to classify sentiment expressed in each sentence. The first step is to identify whether the sentence is subjective or objective. If the sentence is subjective, Sentence-level SA will determine whether the sentence expresses positive or negative opinions. Wilson et al. have pointed out that sentiment expressions are not necessarily subjective in nature.

Driven by the increasing amount of online communication sentiment analysis has become one of the fastest growing research areas in computer science with applications in several domains. Examples are the use of Google searches for early detection of influence epidemics or forecasting unemployment rates, the use of Twitter data to forecast elections or stock market prices and the use of Facebook in psychological experiments. In this report, we compare and demonstrate the performance of various machine learning algorithms for sentiment analysis using the sentiment140 dataset. This dataset has a total of 1.6 million tweets, each containing less than or equal to 140 characters, and the tweets are categorized into two classes – positive and negative. Twitter is a popular microblogging platform that enables users to post tweets/opinion with a limit of 140 characters of text. The popularity of this platform makes it a viable option for this model comparison report. The objective of this report is to implement relevant machine learning techniques and compare the models using appropriate evaluation metrics. We used 5 of the most commonly used non-neural network classification algorithms – Bernoulli Naïve Bayes, Multinomial Naïve Bayes, Support Vector Machines, Random Forest and Logistic Regression, and generated different models for each of the algorithms using different combinations of vectorization and feature extraction methods including Boolean representation, TF, TF-IDF, unigrams and bigrams. We used the SciKit-learn package in Python to implement the above-mentioned tasks.

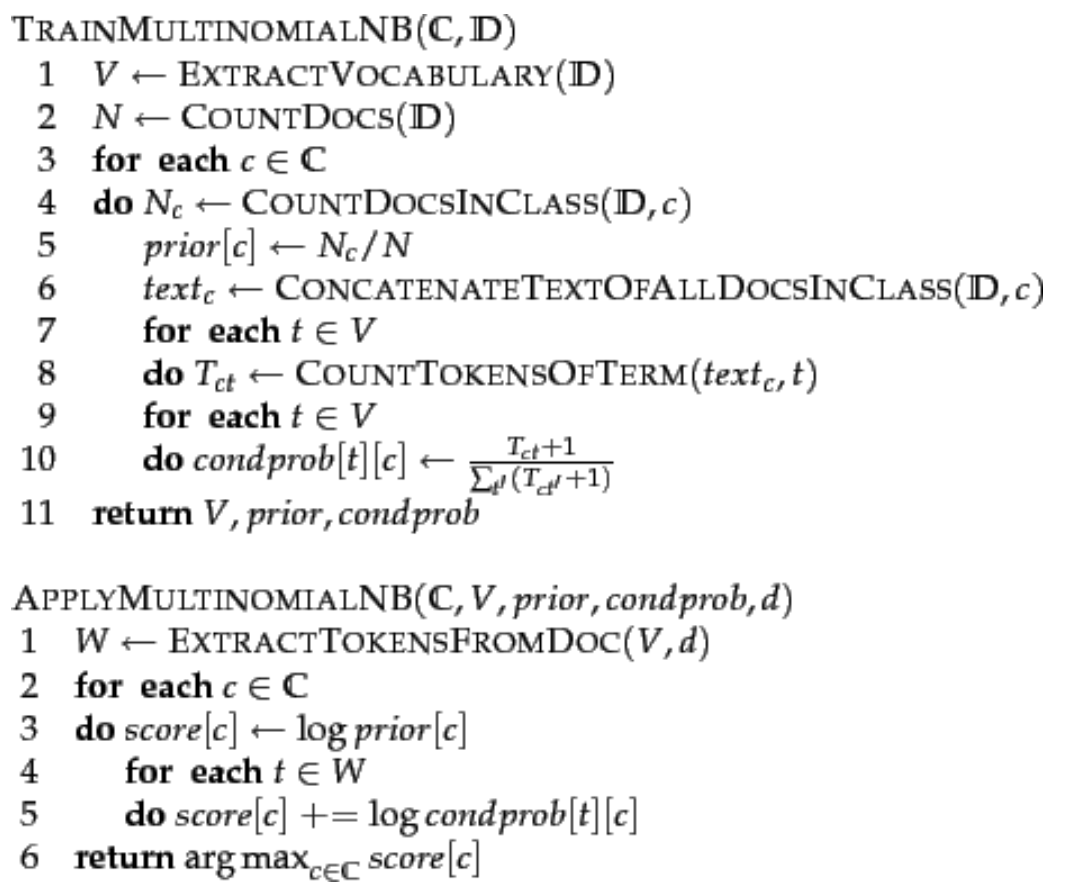
**Theoretical Background**

*Naïve Bayes*

In supervised machine learning, Naïve Bayes classifiers are a family of probabilistic classifiers based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. Bernoulli Naïve Bayes and Multinomial Naïve Bayes are two of the algorithms in this family. The complete algorithm for training a Bernoulli Naïve Bayes classifier is as follows –



The complete algorithm for training a Multinomial Naïve Bayes classifier is as follows (the 1 is added at step 10 to avoid zeros for words not found in the training dataset, a method known as Laplace smoothing) –



*Support Vector Machines (SVM)*

In machine learning, support-vector machines are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. For the purpose of text classification, a linear classifier is used more often and usually works better.

*Random Forest*

Random Forest is an ensemble learning technique for classification, regression and other tasks that operate by constructing a collection of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. They use the “bagging” technique and random selection of features to create the collection of decision trees. Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

1. Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
2. Train a classification or regression tree fb on Xb, Yb.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

or by taking the majority vote in the case of classification trees. This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias.

*Logistic Regression*

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labelled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labelled "1" is a linear combination of one or more independent variables ("predictors"). The independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labelled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labelling. The function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a *logit*, from logistic unit, hence the alternative names. Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the *probit* model. The defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter. For a binary dependent variable this generalizes the odds ratio.

In a binary logistic regression model, the dependent variable has two levels (categorical). Outputs with more than two values are modelled by multinomial logistic regression and, if the multiple categories are ordered, by ordinal logistic regression (for example the proportional odds ordinal logistic model). The logistic regression model itself simply models probability of output in terms of input and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cut-off value and classifying inputs with probability greater than the cut-off as one class, below the cut-off as the other. This is a common way to make a binary classifier.

**Method**

*Data munging*

We followed the steps mentioned in the paper by the creators of the dataset (Go, A. et al 2009). We removed all the user mentions, URLs, HTML tags and non-letter characters including the emoticons. The paper suggests that although the emoticons have sentiment value, their meaning is ambiguous in a lot of the cases, and their presence deteriorates the performance of the models. We expanded all the negation contractions, for example, “don’t” was converted to “do not”. This makes sure that the tokens are complete words. We converted consecutive occurrences of letters more than 2 times to 2 times, for example, “huuuungry” was converted to “huungry”. The dataset contained the following 6 fields –

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Definition** | **Example** |
| target | the polarity of the tweet | 0 = negative, 4 = positive |
| ids | the id of the tweet | 2087 |
| date | the date of the tweet | Sat May 16 23:58:44 UTC 2009 |
| flag | the query (if there is no query, then this value is NO\_QUERY) | lyx |
| user | the user that tweeted | robotickilldozr |
| text | the text of the tweet | Lyx is cool |

We selected the “target” and “text” columns which are class labels and tweets respectively.

*Vectorization and feature selection*

We generated different models for each of the algorithms using different combinations of vectorization and feature extraction methods including Boolean representation, TF, TF-IDF, unigrams and bigrams. The Bernoulli Naïve Bayes algorithm expects the data to have Boolean representations of the features for each of the documents whereas for the Multinomial Naïve Bayes algorithm, the data needs to contain term frequencies of the features. The Boolean representation was only used for Bernoulli Naïve Bayes and for Multinomial Naïve Bayes, only TF was used. For all the other algorithms, both TF and TF-IDF were used, and unigrams and bigrams were used for all the algorithms. Instead of using a predefined list of stopwords, maximum and minimum document frequencies were used to create a dynamic list. This produces better results most of the times as suggested by Saif, Hassan et al. 2014.

*Training and tuning*

The dataset was split into training and test sets with a 99/1 % split. 3-fold cross-validation was used on the training set for training and tuning the models, and the performance was evaluated using the accuracy on the test set. The data was perfectly balanced and therefore, other metrics like precision, recall and F1 score are not required.

1. Bernoulli Naïve Bayes – The cross-validation method was used to find the best values for the parameters – minimum document frequency, maximum document frequency and alpha, which is the smoothing parameter. The model was trained on the training dataset using these values and the accuracy was evaluated on the test set.
2. Multinomial Naïve Bayes – The cross-validation method was used to find the best values for the parameters – minimum document frequency, maximum document frequency and alpha, which is the smoothing parameter. The model was trained on the training dataset using these values and the accuracy was evaluated on the test set.
3. Linear SVM – The cross-validation method was used to find the best values for the parameters – minimum document frequency, maximum document frequency and C, which is used to adjust the margin of the hyperplane of the linear SVM. The model was trained on the training dataset using these values and the accuracy was evaluated on the test set.
4. Random Forest – The cross-validation method was used to find the best values for the parameters – minimum document frequency, maximum document frequency and maximum features, which is the number of features to select for each decision tree. The model was trained on the training dataset using these values and the accuracy was evaluated on the test set.
5. Logistic Regression – The best performance was obtained for the default parameters.

**Results**

The table below shows the test accuracy values for each of the algorithms and various vectorization and feature selection methods, along with the time taken for training the models.

A screenshot of a cell phone

Description automatically generated

The following observations can be made from the results –

* The results for Multinomial Naïve Bayes and SVM are similar to that reported by Go, A. et al 2009 and Pang and Lee 2009. Both reported accuracies in the range of 78-82 %.
* Naïve Bayes algorithms are the quickest whereas SVM is the slowest of the algorithms.
* Both Naïve Bayes and SVM provide comparable results.
* Random Forest does not perform well (lowest scores amongst the algorithms)
* Logistic regression provides the best results for all the cases.

Looking at the incorrectly classified documents for the best performing model (logistic regression with bigrams), some of the common types of errors occurred in cases where there were no sentiment words (e.g. “where the are my pinking shears rarararrarararr babyproofing while cutting stuff makes me stick shears random places forget them”), contained a mix of both positive and negative words (e.g. “umm its getting betterr than before but its still pretty bad lol”) or the dataset seems to have incorrect labels (e.g. “watching the last leno so glad got to go once” is positive but labelled as negative, “dropped your books off in the library” is neutral but labelled as positive).

**Conclusions**

For large text-based datasets –

* Naïve Bayes algorithms provide sufficiently good results in a short period and can be used to quickly create baseline models. They perform particularly well for datasets containing short text documents as in this case.
* SVM also provides similar results but is computationally very expensive. The accuracy to time ratio is too low and other advanced models which take equivalent amount of time would more likely provide better results.
* Random Forest seems unsuitable for this purpose. In general, Random Forest does not perform well with high dimension datasets and in the case of text analysis, the sparse matrix generated deteriorates the performance of the algorithm.
* Logistic regression also provides sufficiently good results and can also be used to create baseline models in short periods. In this case, the high accuracies obtained indicate that neural network-based algorithms are also likely to perform well.

**Additional Information**

Data URL – <https://www.kaggle.com/kazanova/sentiment140>

Code links –

* Bernoulli NB – <https://github.com/aatishsuman/ist736/blob/master/bernoulliNB.ipynb>
* Multinomial NB – <https://github.com/aatishsuman/ist736/blob/master/multinomialNB.ipynb>
* SVM – <https://github.com/AbhirajSingh88/IST-736-Text-Mining/blob/master/clean%2BlinearSVC%2BRandomForest%20(1).ipynb>
* Random Forest – <https://github.com/aatishsuman/ist736/blob/master/randomForest.ipynb>
* Logistic Regression – <https://github.com/aatishsuman/ist736/blob/master/randomForest.ipynb>

**References**

Go, A.; Bhayani R.; Huang, L. 2009. Twitter Sentiment Classification using Distant Supervision. *CS224N Project Report, Stanford*, pages 1-12.

Yelena Mejova, 2009. Sentiment Analysis: An Overview.

Medhat, W.; Hassan, A.; Korashy, H. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal* (2014) 5, 1093–1113.

Pang, B. and Lee, L. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1-135, 2008.

Saif, Hassan; Fernández, Miriam; He, Yulan and Alani, Harith (2014). On stopwords, filtering and data sparsity for sentiment analysis of Twitter. In: *LREC 2014, Ninth International Conference on Language Resources and Evaluation. Proceedings.*, pp. 810–817.

Ginsberg, J.; Mohebbi, M.H.; Patel, R.S.; Brammer, L.; Smolinski, M.S.; Brilliant, L. Detecting influenza epidemics using search engine query data. *Nature* 2009, 457, 102–104.

D’Amuri, F.; Marcucci, J. The predictive power of Google searches in forecasting US unemployment. *Int. J. Forecast.* 2017, 33, 801–816.

Gayo-Avello, D.A. Meta-Analysis of State-of-the-Art Electoral Prediction From Twitter Data. *Soc. Sci. Comput. Rev.* 2013, 31, 649–679.

Huberty, M. *Can we vote with our tweet.* On the perennial difficulty of election forecasting with social media. Int. J. Forecast. 2015, 31, 992–1007.

Bollen, J.; Mao, H.; Zeng, X.J. Twitter mood predicts the stock market. *J. Comput. Sci.* 2011, 2, 1–8.

Kramer, A.D.I.; Guillory, J.E.; Hancock, J.T. Experimental evidence of massive-scale emotional contagion through social networks. *Proc. Natl. Acad. Sci.* USA 2014, 111, 8788–8790.