# Naive Bayes Sentiment Analysis

Brenda Izquierdo, Mireya Jurado, Susana Palacios

Florida International University

CAP 5771: Principles of Data Mining

## 1 Abstract

Sentiment analysis, as a prominent measurement tool of opinion mining, is experiencing significant interest in academic research due to its potential to unlock business intelligence insights at an unparalleled level. The increasing presence of opinion text on-line has made experiments feasible and increasingly recurrent. In this project, we focused on the implementation of the Multinomial Naïve Bayes Classifier as a classification model of book reviews in three categories: positive, negative or neutral. Data used in this experiment are on-line book reviews collected from the popular website GoodReads. The experiment was implemented both manually and using the popular Python library of Scikit-learn for data mining and data analysis. Lastly, we present our evaluation of both approaches, compare their performances, and provide insight into their limitations.

## Contents

## 2 Introduction

With the advent of on-line shopping, reviews have become a critical portion of the decision-making process of a buyer. Given such predominant role, on-line reviews provide customers and sellers alike abundant information on the product and an overall view of its perceived quality, as well as the level of customer satisfaction after the transaction took place. Such insights into consumer behavior and preferences are critical to gauge current marketing strategies and develop future product value.

Given this motivation, we constructed two classifiers that used a Naive Bayes algorithm to predict the positive, negative, or neutral orientation of book reviews. This report will firstly provide the background behind this research and situate the classifier within its academic context. It will then provide a brief overview of related work and past sentiment analysis research. Next, this report will describe the methodology of our Multinomial Naive Bayes Classifier. This section will address the data cleaning process of a body of over 350,000 instances of book reviews, the impact of the selection process of words to be evaluated by the classifier, and the decision to use k-fold cross-validation to split of train and test the datasets. Lastly, it will discuss our experimental results and provide some guidance for future work.

## 3 Background

Tan defines data mining as "the process of automatically discovering useful information in large data repositories" (Tan, Steinbach, and Kumar (2005), p.2). This process differs from traditional data analysis due to the large scale, high dimensional, and heterogeneous nature of the data source (Tan et al. (2005)). This process lies at the

intersection of statistics and artificial intelligence and be predictive or descriptive in nature. A predictive task predicts an attribute value based on the values of other attributes while a descriptive task derives patterns. Predictive tasks can be further subdivided into classification and regression in which the former evaluates discrete variables and the later evaluates continuous variables.

Classification is the task of assigning objects to predefined categories while a classifier is "a systematic approach to building classification models from an input data set" (Tan et al. (2005), p. 148). Examples of classifiers include neural networks, support vector machines, and Naive Bayes classifiers. These classifiers can be applied to a variety of tasks, including computer vision, credit scoring, or search engines. One area that incorporates classifiers is Natural Language Processing (NLP). NLP is a field concerned with how computers can understand or affect text or speech (Chowdhury (2003)). NLP can be applied to many problems including text summarization, speech recognition, and sentiment analysis. Sentiment analysis is the application of NLP that aims to determine opinion from text documents (Gamallo and Garcia (2014)).

There are challenges with respect to conducting data mining with the goal of sentiment analysis. Firstly, an accurate classifier requires a large set of test and train data to determine features and train the classifier (Prabowo and Thelwall (2009)). Secondly, sentiment analysis can be difficult since opinion is subjective and can vary between individuals (Villena Román, Lana Serrano, Martínez Cámara, and González Cristóbal (2013)). One person may interpret a text positively while another may not.

The Naive Bayes Classifier is a popular classifier that can used to perform sentiment analysis on text documents. A Naive Bayes classifier applies Bayesian inference to an

unordered set of words. Bayesian inference is the application of Bayes' rule in order to determine the probability of a particular element belonging to a class (Jurafsky and Martin (2014)).

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

Multinomial Naive Bayes was first proposed by Maron in 1961 in an attempt to automatically categorize journal abstracts (Maron (1961)). He argued that with this technique, a machine could read text and determine its category. Another notable application of this technique to text classification was performed by Mosteller and Wallace in 1964 (Jurafsky and Martin (2014) citing Mosteller and Wallace (1964)). These statisticians evaluated twelve disputed "The Federalist" papers in an attempt to determine the identity of their authors (Fienberg (1971)).

## 4 Related Work

Sentiment analysis, also called opinion mining, "deals with the computational treatment of opinion, sentiment, and subjectivity in text" (Pang, Lee, et al. (2008)). Sentiment analysis can be applied to large available data sources in order to determine what people think, what they like, and what they buy. This task is considered a precursor and building block of recommendation systems and corporate on-line presences. During recent years, there has been an undeniable burgeoning in research in sentiment analysis where "99% percent of the papers have appeared after 2004 making sentiment analysis one of the fastest growing research areas" (Leszek (2016)). This increase is closely related to the proliferation of data available for experimentation.

Sentiment analysis techniques have included Naive Bayes, maximum entropy, and

support vector machines (SVM) (Vinodhini and Chandrasekaran (2012)). It is challenging to directly compare the success of Multinomial Naive Bayes with other classifying approaches such as with respect to sentiment analysis due to the variety of implementations and the size and nature of datasets employed by researchers (Prabowo and Thelwall (2009)).

Zhang, Ye, Zhang and Li employed naive Bayes and SVM to online restaurant reviews written in Cantonese (Zhang, Ye, Zhang, and Li (2011)). Through the use of a crawler, they collected 1,151 positive reviews and 913 negative reviews and sub-sampled them for 900 reviews of each category. They then used three-fold cross-validation and achieved an average accuracy of 88.83%.

Melville, Gryc and Laurence evaluated 20,488 technology blogs containing 1.7 million posts, 16,741 political blogs, and 2000 movie reviews (Melville, Gryc, and Lawrence (2009)). For each of these domains, they implemented 10-fold cross-validation on five different models: a lexical classifier, a future supervision model, a naive Bayes model, a linear pooling model, and a log pooling model. Although the model with the best accuracy was linear pooling, they argued that more data could improve the naive Bayes model.

Hasan, Sabuj, and Afrin used a Naive Bayes classifier on 16,000 Amazon reviews (Hasan, Sabuj, and Afrin (2015)). They also translated the reviews into Bengali to test the accuracy of the classifier before and after translation. With their English data, they obtained an 86% accuracy rate and on their Bengali data, they obtained a close 85% accuracy rate.

Sentiment analysis classification can be performed using Java and Weka (Egidi (2016)). The Java class StringToWordVector is a feature extractor which converts string

attributes into a set of numeric attributes representing word occurrence information from the text contained in the strings. The dictionary is determined from the first batch of data filtered (typically training data). The classification is unsupervised; several Java class objects can be chosen to perform the prediction: SupportVectorMachine, Naive Bayes, Naive Bayes Multinomial, and J48 Decision Tree. The data filter can be attribute or instance based, and K-fold cross validation can be applied to test the classifier accuracy (Egidi (2016)).

## 5 Methodology

The experiment consisted in implementing a natural language processing classifier with the objective of automatically predicting new book reviews from a preselected dataset into positive, negative or neutral. The classifier would identify the tone of the review by analyzing the text and would label such review with the aforementioned types. We decided to use the Naive Bayes Classifier to estimate the class-conditional probability of the reviews. Since each review is represented as a vector of words, for purposes of the experiment, each word is assumed to be probabilistically independent and their order immaterial.

The reviews are then converted into feature sets where the attributes are possible words and the values are the number of times a word occurs in the given review. Based on the word frequency and its associated label, the word is incorporated into a lexicon of positive, negative or neutral words. These lexicons are then used to evaluate which type of words are predominant in a review to predict the review's label as positive, negative, or neutral.

The experiment was divided in the following three steps. Firstly, the dataset was selected and pre-processed for analysis. This step involved removing duplicates, empty values, and non-English words. Secondly, k-fold cross-validation was chosen to minimize data bias and variance. Lastly, two classifiers were created to evaluate the dataset. While both classifiers used a Naive Bayes classification model, one used the Python libraries developed by SKlearn and the implemented the classifier manually.

### 5.1 Data Pre-Processing

A dataset of books reviews found on https://data.world/ssaudz/goodreads-review-of-350-k-books was selected for the experiment. This dataset (br.csv) contained 350,000 reviews from the website “Good Reads”, a popular on-line social network known for its book reviews and recommendations. The file contained the following variables: “bookID”, “title”, “author”, “rating”, “ratingsCount”, “reviewsCount”, “reviewerName”, “reviewerRating” and “review”. Moreover, the dataset presented missing and corrupted values as well as reviews written in languages other than English. Therefore, the parser first verified that the columns "review" and "reviewerRating" contained a score and text respectively since the value of the latter will be used as a reference point to further classify the review.

The “reviewerRating” column originally contained book reviews from 1 to 5. In order to evaluate on only three classes, the reviews were divided into three groups. Those rated 4 and 5 were designated as “positive”, those rated 1 and 2 were designated as “negative” and all 3 reviews were designated as “neutral.” An additional column named “label” was added to the dataset to associate the reviews with their positive, neutral or negative categories,

denoted with 1, 0 and -1 respectively.

Subsequently, non-latin characters and symbols were removed and English contractions were separated. Removal of replicated reviews took place as well as those reviews that were not in English. The resulting dataset was stored in the file dataset.csv. The code for this task can be found in the file `Dataset Parser.ipynb` Once this process was finished, the statistics of the parsed dataset were evaluated.

After parsing, the data remained highly skewed towards positive reviews, presenting 3,434 negative reviews, 16,384 neutral reviews and 48,622 positive reviews. As demonstrated in Figure 1, over 71% of the dataset contained positive reviews. With the purpose of balancing the data, 3,300 reviews of each class were randomly selected to form the final dataset (balanced_dataset.csv). The results from this sub-sampling can be seen in Figure 2. The code for this task can be found in `Balance.ipynb`.

*Figure 1*. Percentage of positive, negative and neutral reviews before subsampling

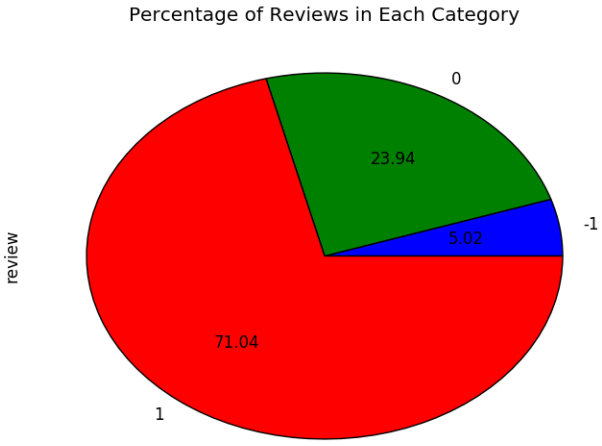
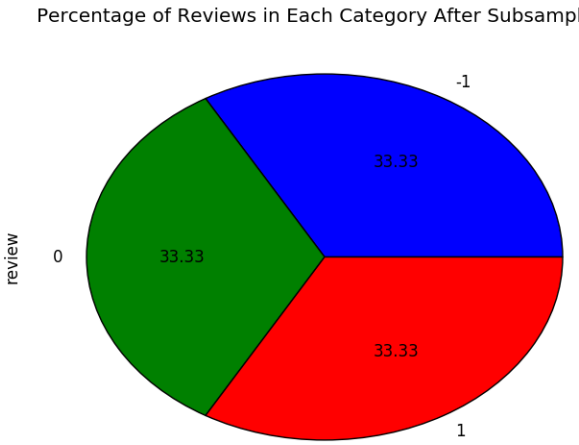


*Figure 2*. Percentage of positive, negative and neutral reviews before subsampling




### 5.2 K-fold cross-validation

Since our original dataset had significantly reduced the original number of instances, following our instructor's recommendation, we decided to utilize k-fold cross-validation. As a method of cross-validation, k-fold cross-validation allowed us to minimize bias and variance by dividing the dataset in $k$ bins of equal size, running $k$ different learning experiments by picking one subset as a testing set and the remaining $k-1$ together as the training set. For every run of the model, one sample is used as the test data while the remaining $k-1$ samples are used to train. The model therefore runs a total of $k$ times and every sample of the data is used for training $k-1$ times and testing 1 time. The accuracy is then calculated for each run and the final accuracy is an average of these runs.

The SKlearn model ran k-fold cross-validation for three values of $k = \{3, 4, 5\}$. The motivation behind this decision was compare the accuracy of the models with different $k$ values and different percentages of the data belonging to the training and testing data.

When $k = 3$, the training set contained 66% of the data with the remaining 33% of the data belonging to the testing set. When $k = 4$, the training set for all 4 runs contained 75% of the data with 25% of the data in the testing set. Lastly, when $k = 5$, 80% of the data was used for training and 20% was used for testing.

For each run of the model, the area under the curve was calculated. An average accuracy for each $k$ was calculated from these individual runs. Ultimately, all data points were used both for training and testing the classifier. The hard coded model has a considerably longer run time (20-25 hours) and therefore only used one $k$ value where $k = 3$.

### 5.3 Classifier Implementation

Both the manual and automatic Naive Bayes Text Classifiers were implemented in the same way. Firstly, sentences contained in the reviews were split into single words by tokenizing them. Secondly, "stopwords" or words that are to be ignored or filtered out by the classifier were removed since they do not present any content significance. The third step was to train and test the model. The manual classifier iterated through all the reviews in the training set to form a dictionary with the relative frequency of each word per class (dictionary.csv).

There are three ways to create a Naive Bayes classifier with Scikit Learn libraries. The first is `GaussianNB` which implements the Gaussian Naive Bayes algorithm. The Gaussian approach is best for continuous variables that can be fit to a Gaussian distribution. The second approach is `BernoulliNB` which address discrete data but makes classifying decisions based on binary or boolean features. The last approach is `MultinomialNB` which implements the Naive Bayes algorithm on discrete data and is often

used for text classification (Pedregosa et al. (2011)). Due to its ability to handle multiple classes and its prevalence in text classification, our classifier used the `MultinomialNB` package. The first step was to designate the X and y values from the processed dataset. In the model, X is set to the column containing the text reviews and y is assigned to the column containing the labels $\{-1, 0, 1\}$.

The second step was to create a Count Vectorizer to convert the review text into a matrix and extract the features. The vectorizer accepts the review text, creates a dictionary of the words, and extracts the count of each word with its associated label. In our implementation, the vectorizer accepts a dictionary of English "stop words." Stop words are frequently occurring words such as "for", "do", and "by." By accepting this parameter, the vectorizer completely ignores these words when it builds the dictionary. Stop words were used for both the SKlearn and manual classifiers in order to accommodate the computationally intensive manual model.

Within the model itself, the `Kfold` object returns two sets of indices that identify the position of the train and test data for that run. The features, or review text is then tokenized by the vectorizer and a dictionary is formed from the train data. The model then fits the features established from the train data to their respective labels. Next, the model predicts the label of the test features and calculates accuracy.

After all $k$ runs have completed for a specific $k$ value, an average accuracy is computed. As stated above, the classifier runs a total of 12 times with three separate $k$ values. When $k = 3$, the model runs 3 times, when $k = 4$, it runs 4 times, and when $k = 5$, it runs 5 times. The code of this model can be found in `Multinomial Naive Bayes with K Fold Cross Validation.ipynb`.

In the manual model, after obtaining the frequency table of each word per class, we computed the prior probability for each class: P(c) = $\frac{R(c)}{R}$ where R(c) is the number of reviews of class c and R the total number of reviews. Next, we computed the conditional probability of each word using the conditional probability formula:

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

where *count*$(x|c)$ is the number of times that a word x appears in reviews of class c; + 1, Laplace smoothing; count(c), the total number of words that appear in reviews of class c and |V| the total number of words in the dictionary.

The final step was to compute the posterior probability:

$$P(X|c) = P(x_1|c) * P(x_2|c) * P(x_3|c) * \cdots * P(x_n|c) * P(c)$$

The text of the new review is represented by X, and each word in it by $x_1$, $x_2$, $x_3$ up to $x_n$; c is the class. Thus, we ended up with three posterior probability values for the new review, one for each class, and finally, we labeled the new review with the class that had the highest posterior probability (prediction.csv). The code of this model can be found in `Multinomial Naive Bayes with K Fold Cross Validation [Hard Coded].ipynb`.

## 6 Results

The following section will outline the results from both the manual model and the SKLearn model. The accuracy for $k = 1$ for the manual classifier was 56.9% and the average accuracies for the SKlearn model where $k = \{3, 4, 5\}$ was 54%. These low accuracy scores are likely due to nature of the train and test datasets.

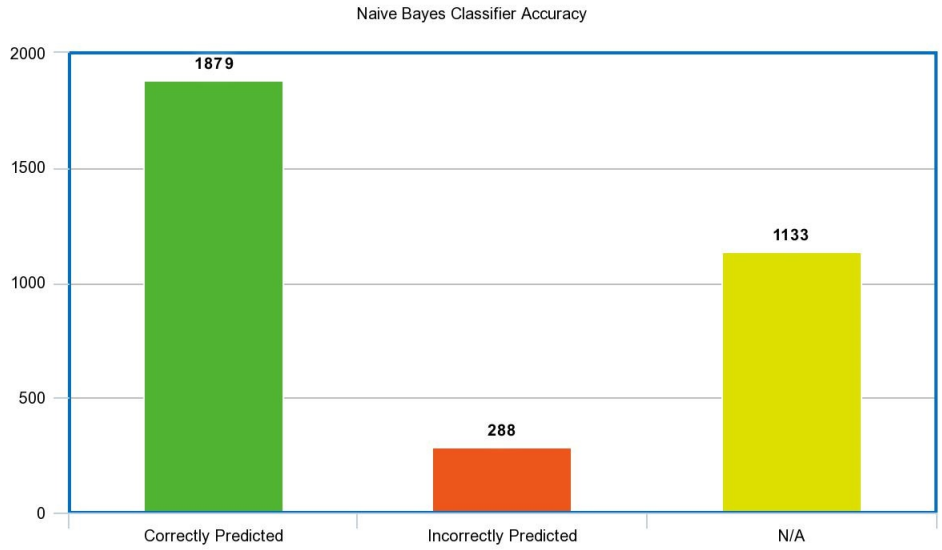
### 6.1 Manual Model

The classifier accuracy was of 56.93% with 1,879 instances correctly classified. In total, 1,133 instances could not be classified and 288 were labeled wrongly. A review could not be classified when it had the same posterior probability for more than one class; this occurred when a review contained in its majority (or entirety) words that could not be matched to a dictionary entry. In those cases, the conditional probability became:

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

Since we had a balanced number of positive, negative, and neutral entries in test.csv, $\text{count}(c) + |V|$ was exactly the same for the three classes, thus affecting the classification process.

*Figure 3*. Naive Bayes Classifier Accuracy for K = 1



### 6.2 SKlearn Model

Multinomial Naive Bayes fits a unique model for each class in the dataset and so for every $k$, $k$ unique models are fitted. The accuracy of this model is then measured by the internal scoring strategy associated with the `cross_val_score` tool. For every $k$, this tool calculates the accuracy of every run. The results from these tests can be seen in the table below.

| K | Accuracy | Standard Deviation |
|---|---|---|
| 3 | 54% | + 0.01 |
| 4 | 54% | + 0.05 |
| 5 | 54% | + 0.05 |

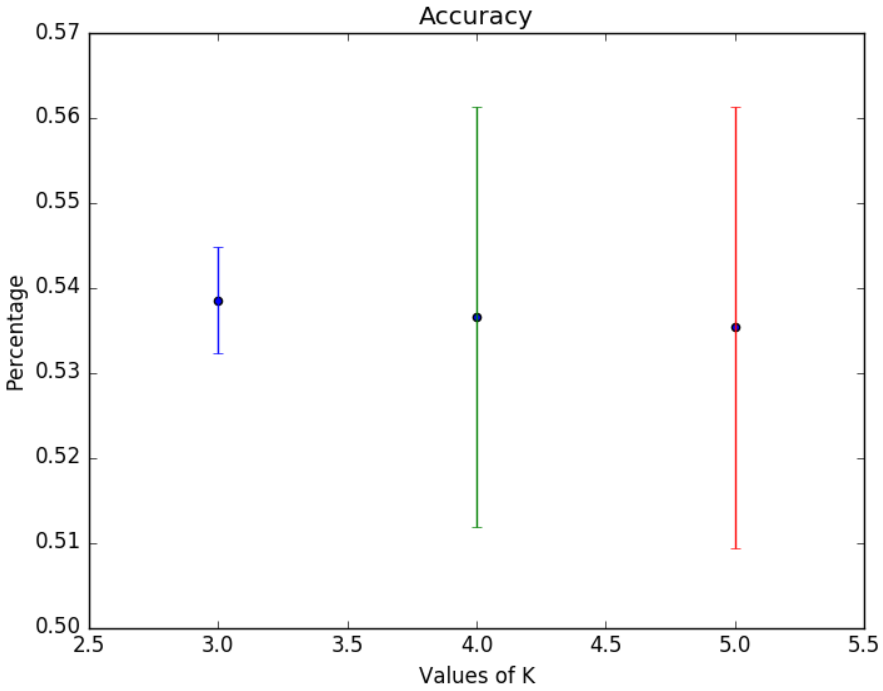
The low accuracy of this model can be explained by the over-fitting tendencies of the Naive Bayes algorithm. The model is accurate at classifying training data that it already contains in its dictionary but it is not generalized and cannot accurately predict new data. If a review contains words that the model has not seen before, it is unlikely to be classified properly.

## 7 Discussion

As mentioned in the Results section, the low accuracy from the Naive Bayes approach may be due to a high percentage of novel words in the testing set. If the testing data contains words that are not found in the dictionary created by the training data, the classifier may have difficultly correctly interpreting the tone of the review. This issue could be addressed by using a larger dataset or by increasing the number of k-folds.

The low accuracy could also be due to fact that the data was split into three classes

*Figure 4*. Average accuracies and standard deviations for the respective k values




instead of two positive and negative sets. In order to address this possibility, future work could use existing SciKit Learn libraries that are expressly created to address multi-classification problems. While `MultinomialNB` is capable of evaluating data with multiple classes, there are other libraries that may result with a higher accuracy rate. In addition to using a different classifier technique, a combination of multiple classifiers may produce a better accuracy rate (Prabowo and Thelwall (2009)).

One additional area to improve would be in the implementation of stopwords during the lexicon building process. According to Saif, Fernandez, He, and Alani (2012) "Words that do not provide any discriminative power in one context may carry some semantic information in another context". Furthermore, the authors state that "the use of pre-compiled (classic) stoplist has a negative impact on the classification performance."

Naive Bayes classifiers are highly sensitive to stopword removal, thus they need an ideal stoplist modeled with the input text in mind in order to achieve a high classification performance. Future work could create a unique stoplist or not employ any stopwords list.

## 8 Conclusion

To conclude this paper, when working on building two Naive Bayes classifiers, one using the SKlearn module from Python, and the other one manually, both presented similar error rates since the same principles of calculation were implemented in both models. A limited feature set of words that was used as a dictionary proved to be the biggest obstacle to produce a definite result on categorizing certain reviews. If there is a lack of occurrence of a class label and an attribute value together, the Naive Bayes frequency based probability will be zero, affecting the posterior probability estimate which is ultimately our data point of interest. We consider important to conduct further experiments with additional methods such as Support Vector Machine and continue comparing results to increase the understanding of their ideal usability as well as their overall limitations such as in the case of the Naive Bayes Classifier in our experiment.

# 9 References

Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, *37*(1), 51–89.

Egidi, S. (2016). Java application for text classification in a social context.

Fienberg, S. E. (1971). *Inference and disputed authorship: The federalist.* JSTOR.

Gamallo, P., & Garcia, M. (2014). Citius: A naive-bayes strategy for sentiment analysis on english tweets. In *Proceedings of the 8th international workshop on semantic evaluation (semeval 2014)* (pp. 171–175).

Hasan, K. A., Sabuj, M. S., & Afrin, Z. (2015). Opinion mining using naive bayes. In *Electrical and computer engineering (wiecon-ece), 2015 ieee international wie conference on* (pp. 511–514).

Jurafsky, D., & Martin, J. H. (2014). *Speech and language processing* (Vol. 3). Pearson London:.

Leszek, Z. (2016). The sentiment analysis as a tool of business analytics in contemporary organizations. *Studia Ekonomiczne*, *281*, 234–241.

Maron, M. E. (1961). Automatic indexing: an experimental inquiry. *Journal of the ACM (JACM)*, *8*(3), 404–417.

Melville, P., Gryc, W., & Lawrence, R. D. (2009). Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1275–1284).

Mosteller, F., & Wallace, D. (1964). Inference and disputed authorship: The federalist.

Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and*

*Trends® in Information Retrieval*, *2*(1–2), 1–135.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Prabowo, R., & Thelwall, M. (2009). Sentiment analysis: A combined approach. *Journal of Informetrics*, *3*(2), 143–157.

Saif, H., Fernandez, M. F., He, Y., & Alani, H. (2012). On stopwords, filtering and data sparsity for sentiment analysis of twitter. , 810-817.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to data mining. 1st.* Boston: Pearson Addison Wesley. xxi.

Villena Román, J., Lana Serrano, S., Martínez Cámara, E., & González Cristóbal, J. C. (2013). Tass-workshop on sentiment analysis at sepln.

Vinodhini, G., & Chandrasekaran, R. (2012). Sentiment analysis and opinion mining: a survey. *International Journal*, *2*(6), 282–292.

Zhang, Z., Ye, Q., Zhang, Z., & Li, Y. (2011). Sentiment classification of internet restaurant reviews written in cantonese. *Expert Systems with Applications*, *38*(6), 7674–7682.