ECE681

Course Project Report

Shengxin Qian

Table of Contents

1. Problem Description

Today, automatic text categorization has become one of the most popular research topics in the natural language processing area. Categorization of the overall opinion towards the subject text which is also called sentiment classification is one of the most important parts of automatic text categorization. Sentiment classification could label texts with several attitudes such as “positive” or “negative” which would help the program to understand human attitude behind those texts. One of the most important implementations of sentiment analysis is to label rapidly growing online reviews on review sites which do not contain star rating indicators. Sentiment classification could also help companies to do preanalysis on free-form surveys in a natural language format.

In this project, I would use SVM and Naïve Bayes classifiers to label the reviews on IMDB website with “positive” and “negative” tags. The challenge of this project is to select proper linguistic features and to design a linguistic model to preprocess the datasets.

Sentiment classification was used in many business intelligence applications such as MindfulEye’s Lexant system and message filtering system (Spertus, 1997)[2]. Sentiment analysis is a very popular topic in machine learning. In 1994, Daniel Jurafsky finished “The berkeley restaurant project” [1] (a research on the sentiment of restaurant reviews) and built the corresponding database. In 2005, Pang and Lee [2] collected many linguistic structures from movie reviews. Many machine learning algorithms were used including SVM, Naïve Bayes and different types of neural network [3]. The accuracy of sentiment classification has been more than 80% [4].

2. Data Description

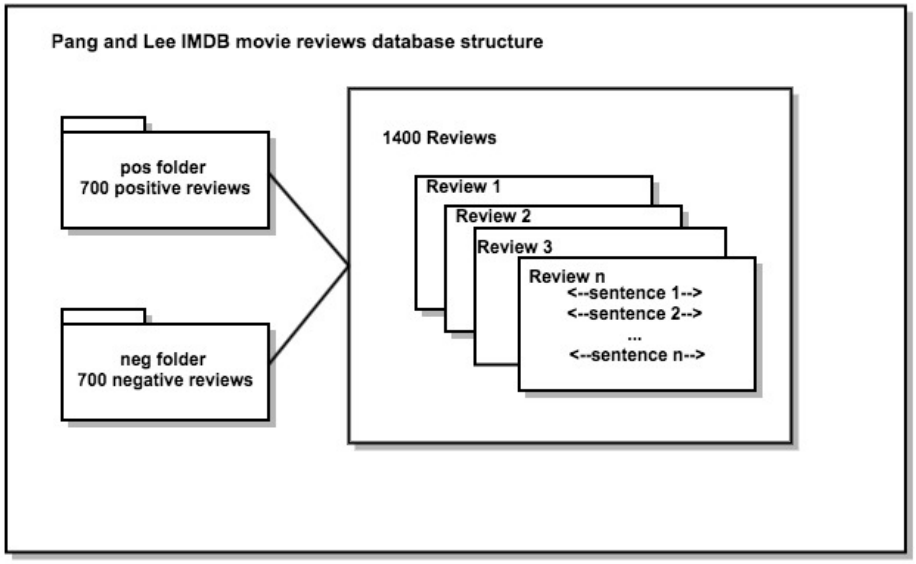

Figure 1: Structure of Pang and Lee Movie Reviews Database

This project used standard movie reviews database[1] as the datasets for training and testing. The reviews were gathered from IMDB website and preliminary steps were taken to remove rating information in HTML files. There are 1400 processed lowercase text files in standard database. Half of them were labeled positive in "pos" folder and the rest were labeled negative in "neg" folder [2]. Each text file only contains the content of reviews and each line corresponds to a single sentence, as determined by Adwait Ratnaparkhi's sentence boundary detector MXTERMINATOR [7].

Star rating	**0 - 2.5**	**3**	**3 - 5**
Tag	positive	neutral	negative

Table 1: Rule of tagging movie reviews in database

With a five-star system (or compatible number systems), three-and-a-half stars and up are considered positive, two-and-a-half stars and below are considered negative. Three-star reviews were not stored in database because three stars represent neutral.

[1] This data was first used in Bo Pang and Lillian Lee,``A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization. Based on Minimum Cuts'', Proceedings of the ACL, 2004.

Tag	Positive	Negative	Whole
Number of words	2006	4783	6789

Table 2: Statistics of opinions lexicon

This project also used English opinions lexicon[2] to extract positive and negative opinion words from reviews. This English opinions lexicon was compiled over many years starting from Hu and Liu's research [8]. This lexicon contains 6789 opinion words. There are 2006 positive opinion words and 4783 negative opinion words. The opinions lexicon was tagged by providers based on the dictionary.

3. Approach

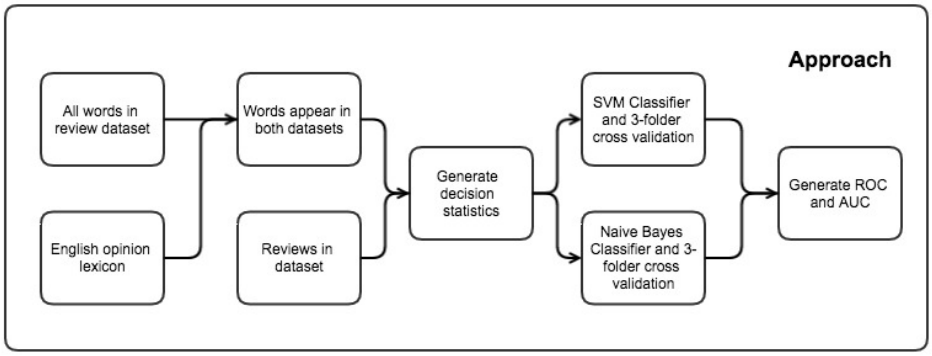

Figure 2: Flowchart of general approach

3.1. Pre-processing

Intuitively, we might suspect that there are certain words people tend to use to express strong sentiment. So, it might be helpful if we simply select a group of sentiment words based on the meaning of the words and rely on the selected words to classify.

Sentiment	Proposed word lists
positive	love, wonderful, best, great, superb, still, beautiful
negative	bad, worst, stupid, waste, boring, ?, !

Table 3: Opinion words in baseline approach

The baseline approach used only 7 positive words and 7 negative words to represent

[2] This data was first used in Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA,

positive and negative sentiment (unigrams). The selected words were shown in Table 3. The selection of those words was based on Pang and Lee 's research [2]. I use the following bag-of-features framework with those selected words. Let $\{f_1, \ldots, f_{14}\}$ be the set of features and f_i represent one pre-defined sentiment word. Let $n_i(d)$ be the number of times feature f_i occurs in review d. Then, the program would produce a vector $(n_1(d), \ldots, n_{14}(d))$ for each review in the database as decision statistics. So, 1400 reviews could be represented as a 1400 x 14 matrix.

After finishing baseline approach, I expanded the scale of selected sentiment words. I considered all sentiment words appearing in opinion lexicons and the reviews dataset at the same time as the features. The number of all considered features was 4198. Same as baseline approach, let $\{f_1, \ldots, f_{4198}\}$ be the set of features and f_i represent one opinion word. Let $n_i(d)$ be the number of times feature f_i occurs in review d, the program would produce a vector $(n_1(d), \ldots, n_{4198}(d))$ for each review in the database as decision statistics. So, with 1400 reviews, the decision statistics could be represented as a 1400 x 4198 matrix after expanding the scale of sentiment words.

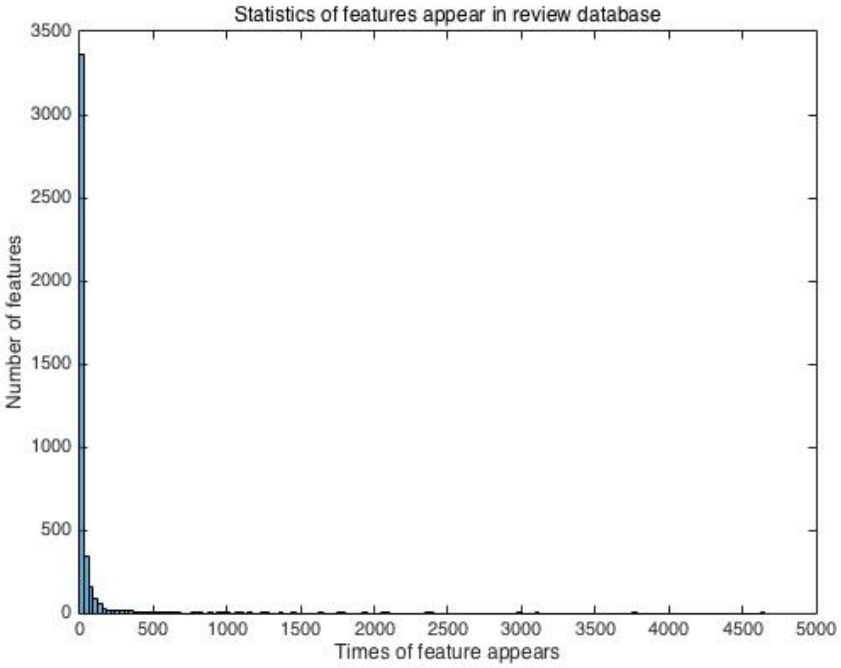

Figure 3: Statistics of 4198 features appear in review database

As we can see in Figure 3, all feature words appear at least once and most feature words only appear several times. So, it makes the decision statistics matrix very sparse.

3.2. Classifier selection

I use the Support vector machines (SVM) classifier [3] with linear kernel inside scikit-learn package [10] and the Naive Bayes [5][6] written by myself to classify reviews. SVM and Naïve Bayes are two of the most popular classifiers in text classification.

3.2.1. Naïve Bayes

Naïve Bayes classifier is based on Bayes' rule. Given a class variable and a dependent feature vector x_1 through x_n, Bayes' rule could be stated as follows:

$$P(y|x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)}$$

If we assume feature words are independent and ignore the constant $P(x_1, \dots, x_n)$, we could simplify the Bayes' rule to:

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

That could also be stated as:

$$\hat{y} = \arg max\, P(y) \prod_{i=1}^{n} P(x_i|y)$$

So, we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i|y)$. The former is the relative frequency of class y in the training set. The key part of Naïve Bayes classifier is choosing the assumed distribution of $P(x_i|y)$ and there are three popular assumptions that worked well in document classification and spam filtering [11]:

- **Gaussian Naive Bayes**

$$P(x_i|y) = \frac{1}{\sqrt{2\pi{\sigma_y}^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2{\sigma_y}^2}\right)$$

The parameter σ_y and μ_y are estimated using maximum likelihood.[5][6]

- **Multinomial Naive Bayes**

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

The parameter N_{yi} is the number of times feature i appears in a sample of class y in the

training set. N_y is the total count of all features in class y. The parameter α is the smoothing prior which accounts for the feature not appearing in the learning sample. In this case, we choose $\alpha = 1$ called Laplace smoothing. [5][6]

- **Bernoulli Naive Bayes**

$$P(x_i|y) = \frac{N_{yi} + \alpha}{N_y + \alpha n} x_i + (1 - \frac{N_{yi} + \alpha}{N_y + \alpha n})(1 - x_i)$$

The parameter is similar to Multinomial Naïve Bayes classifier. $x_i = 1$ if feature i appears and $x_i = 0$ if feature i does not appear. The difference between Bernoulli Naïve Bayes and Multinomial Naïve Bayes is the latter would explicitly penalize the non-occurrence of a feature. [5][6]

3.2.2. Using statistics of the data -- Baseline algorithm

Without using machine learning techniques, we can simply count the number of proposed positive and negative words in feature set. The decision statistics could be generated by the following rule (Pang and Lee 2004) [2]:

$$decision\ statistic = The\ number\ of\ times\ all\ positive\ words\ appear - The\ number\ of\ times\ all\ negative\ words\ appear$$

3.2.3. Feature binarized function

With $n_i(d)$ (the number of times feature f_i occurs in review d), we can binarized frequency statistics of features with the following equation (Pang and Lee 2004) [2]:

$$\mathrm{F}(n_i(d)) = \begin{cases} 1, n_i(d) > 0 \\ 0, n_i(d) = 0 \end{cases}$$

3.3. Result

3.3.1. Result of using 14-word feature framework -- Baseline approach

In baseline approach, I use my program to produce the baseline result for using statistics of data and SVM classifier. As we know, for the normal binary classification problem, the expected accuracy should be larger than 50%. When using statistics of data, the program would calculate the number of occurrence of positive and negative words. If the times of positive words occur is higher, the classification decision is “positive”. If the times of negative words occur is higher, the classification decision is “negative” and the rest is “tie”.

If the decision is “tie”, it would be categorized as false alarm. After using this algorithm, the program could achieve 63% accuracy with 14-word feature framework. At the same time the percentage of “tie” is 14% which limits the accuracy of this simple approach. It means the accuracy should be higher than 63% with machine learning method and 14-word feature framework. With SVM classifier and three-fold cross-validation, the average accuracy of classification is 72% which is 9% more than our baseline.

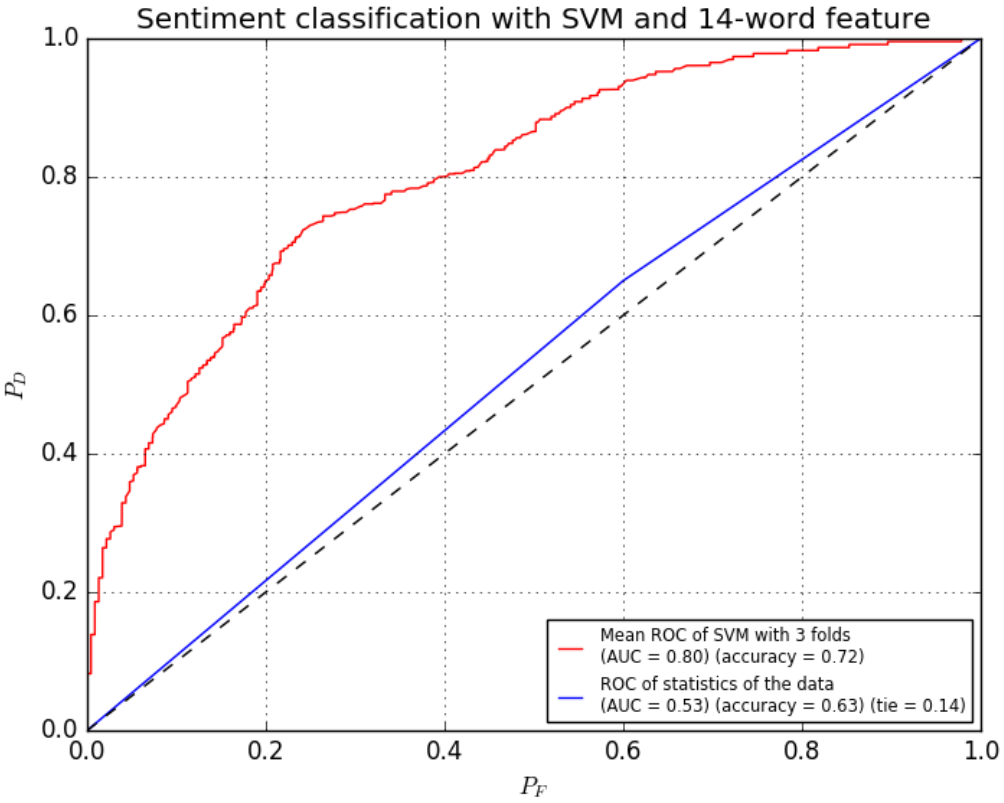

Figure 4: Mean ROC with 14-word feature framework

3.3.2. Result of using 4198-word feature framework

When using 4198 proposed words as feature set, we can see a significant performance improvement. With SVM classifier and three-fold cross-validation, the average accuracy of classification increased from 72% to 77% and the AUC increased from 0.8 to 0.85. This result does make sense. Because the 14-word feature framework could not cover all sentiment words in each review, we could simply increase the number of features to improve the performance. When comparing different Naïve Bayes Classifiers, we can clearly see the difference of performance. Multinomial Naive Bayes Classifier has the best performance and the Bernoulli Naive Bayes Classifier’s performance is similar. The area

under the ROC curve (AUC) of the Multinomial Naïve Bayes Classifier is 0.89 and the accuracy is 81% which is consistent with Pang and lee's research [2]. The area under the ROC curve (AUC) of the Bernoulli Naïve Bayes Classifier is 0.88 which is less than that of Multinomial Naïve Bayes Classifier. This is different from the result of Andrew and Nigam's research [5]. In Andrew and Nigam's research of text classification, Bernoulli Naïve Bayes was supposed to be better than Multinomial Naïve Bayes when dealing with short texts. One possible explanation is that the performance could be different when dealing with different datasets. In IMDB movie review dataset, we may not explicitly penalize the non-occurrence of a feature. On the other hand, Gaussian Naïve Bayes Classifier has the worst performance which is close to the result without classifier. So, we can conclude that the distribution of each word is not Gaussian.

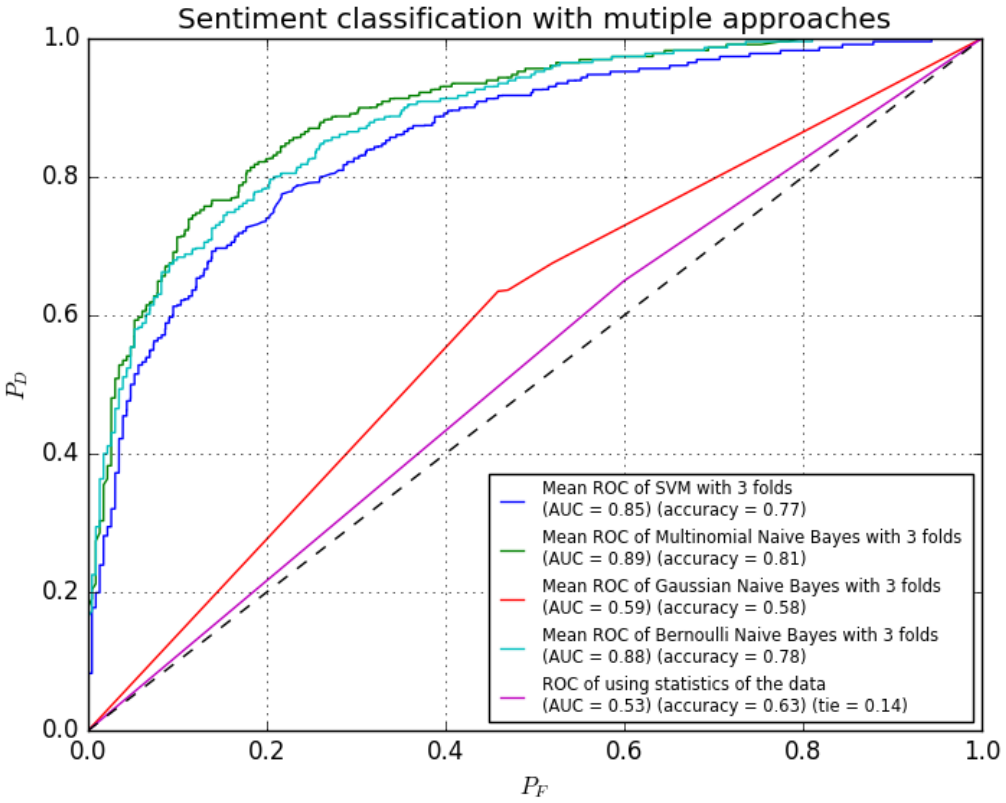

Figure 5: Mean ROC with 4198 features and multiple classifiers

3.3.3. Result of using 4198-word feature framework with feature presence value

Recall that we represent each review with vector ($n_1(d)$,…, $n_{4198}(d)$). If we change the vector into ($F(n_1(d))$, … , $F(n_{4198}(d))$) with feature binarized function, we can make

performance better.

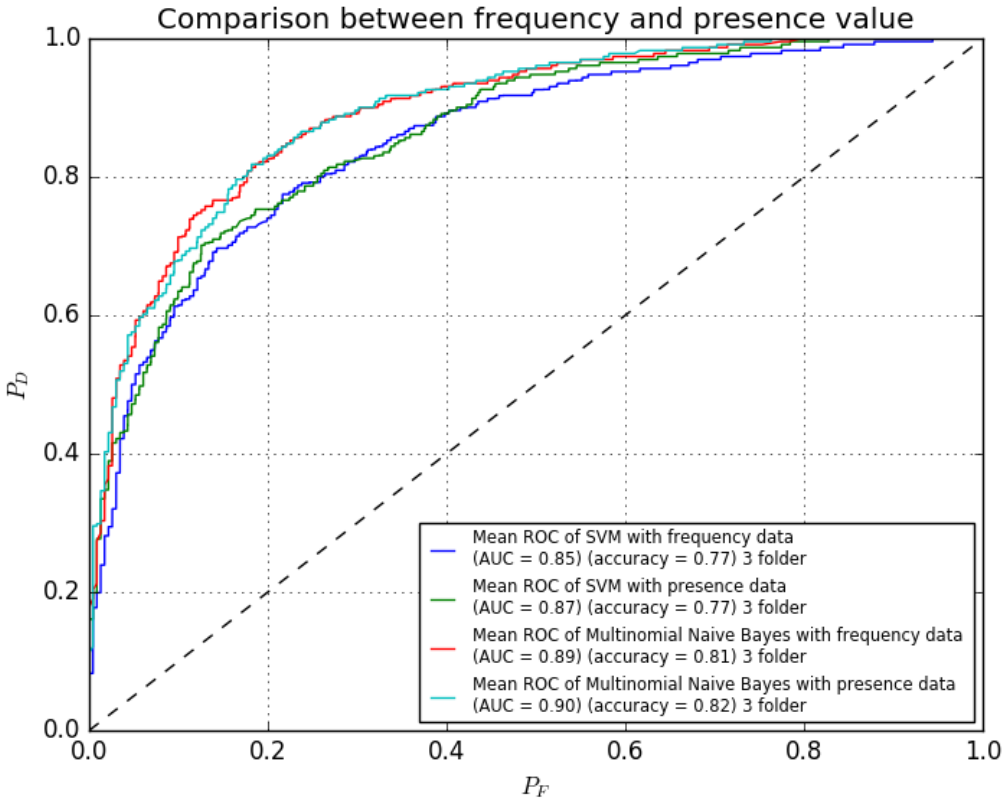

Figure 6: Mean ROC with 4198 features and presence data

As we can see in Figure 6, when we binarized the decision statistics, we can make AUC a little better by accounting only for feature presence. When using SVM Classifier, we improve AUC from 0.85 to 0.87 with binarized function. When using Multinomial Naïve Bayes Classifier, we improve AUC from 0.89 to 0.90 with presence data. This may because sentiment being conveyed mostly by particular content words that tend to be repeated [2]. So using frequency data instead of presence data may strengthen the importance of repeated features.

3.4. Conclusion

As we can see, with increasing the number of feature words, we can increase the area under ROC curve from 0.8 to 0.90 which is close to the upper limit of unigram model [2]. If I want to make a step further, adjusting the weight of each word could be one solution. Because different words have different strength of sentiment, the weight of each word should be different when calculating the decision statistics. Another possible solution is using word embedding but not unigram model to represent features because word embedding may have better performance when dealing with ambiguity problems.

4. Reference

[1] Jurafsky, Daniel, Chuck Wooters, Gary Tajchman, Jonathan Segal, Andreas Stolcke, Eric Fosler, and Nelson Morgan. "The berkeley restaurant project." In *ICSLP*, vol. 94, pp. 2139-2142. 1994.

[2] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques." In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79-86. Association for Computational Linguistics, 2002.

[3] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer, Berlin: Springer series in statistics, 2001.

[4] Socher, Richard, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. "Recursive deep models for semantic compositionality over a sentiment treebank." In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631, p. 1642. 2013.

[5] McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." In *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41-48. 1998.

[6] Metsis, Vangelis, Ion Androutsopoulos, and Georgios Paliouras. "Spam filtering with naive bayes-which naive bayes?." In *CEAS*, pp. 27-28. 2006.

[7] Reynar, Jeffrey C., and Adwait Ratnaparkhi. "A maximum entropy approach to identifying sentence boundaries." In *Proceedings of the fifth conference on Applied natural language processing*, pp. 16-19. Association for Computational Linguistics, 1997.

[8] Hu, Minqing, and Bing Liu. "Mining and summarizing customer reviews." In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168-177. ACM, 2004.

[9] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." *Machine learning* 20, no. 3 (1995): 273-297.

[10] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[11] Zhang, Harry. "The optimality of naive Bayes." *AA* 1, no. 2 (2004): 3.