Jeffrey Levesque

jlevesqu@syr.edu | Text Classifier Evaluation and Human Annotation

IST-736: HW4

PRofessor gates

**Introduction**

Information science plays an ancillary role in communication, education, and general industry. The abundance of data has allowed technology to evolve in ways that pervade most global sectors. In 2015, 400 hours of video were uploaded to YouTube every minute[[1]](#footnote-1). Social media platforms such as Facebook produce 4 petabytes of data each day[[2]](#footnote-2), while search engines such as Google, collect information effectively at scale. Despite copious amount of data permeating across networks, deriving context has brought great value in areas such as data, and text mining. Many have stated that the Information Age is the New Gold Rush[[3]](#footnote-3). Organizations are challenged to apply varying techniques to study behavior and trends of individuals.

Though data is growing at an alarming rate, the ability to extract precise meaning is foundational to various fields. Some challenges include noisy data, misinterpretation, and deceiving language.

Noisy data is often depicted as unstructured, often requiring additional effort for preprocessing. Some examples include text scraped from one or many websites. While parsing content between syntax is often difficult, a greater challenge is often found deriving meaning, and confidence of the interpretation. In the case of reviews, a business would find the truthfulness of a statement to be important. A false statement against a product, would not effectively measure product success. Furthermore, sentiment provides another dimension of measurement. For example, a positive review generally indicates customer satisfaction, while the converse is similarly applicable.

**Analysis**

An analysis contrasted the effectiveness of sentiment and lie detection. Exploratory techniques such as sentiment measure (positive, negative, and neutral), and word clouds were created for lie modals (lying vs. not lying), and sentiment modals (positive vs. negative). Next, classification models were created for each of the earlier modal type. Unigram representation of text reviews were vectorized using TFIDF. Then, a Bernoulli Naïve Bayes, Multinomial Naïve Bayes, and Support Vector Machine were created for each modal category. Furthermore, each model was duplicated against the original unigram representation suffixed with a corresponding part of speech (POS). Specifically, each word was suffixed by a hyphen followed by its POS before applying TFIDF vectorization. Once all classifiers were created, an overall score ensembled related classifiers. Therefore, classifiers related to sentiment were pooled then averaged. Similarly, the ensembled score was conducted for the lie detection. Finally, a k-fold validation was performed for each classifier and modal type. The intent of the validation was to indicate whether the given dataset could be trained. High score variability from the k-fold validation, would indicate either the data was too variable, or the model cannot sufficiently characterize the data.

**Data Preparation**

The provided csv dataset contained a variable number of columns[[4]](#footnote-4). Specifically, some rows contained three columns, while others contained five or more. The original structure intended to consist of three columns - lie, sentiment, and review. Furthermore, matrix computations dictate an n x m dimension. Therefore, a utility dataframe[[5]](#footnote-5) function was created to merge the third column (c3) with any successive (c3+m) columns for the entire csv. The resulting dataframe was a balance n x 3 matrix.

Next, factors within the lie and sentiment columns were converted to integers[[6]](#footnote-6). Specifically, (f, t) 🡪 (0, 1) for the lie column, while (n, p) 🡪 (0, 1). These adjustments allow corresponding classifiers to perform prediction tasks. Additionally, each review column removed urls, string punctuations, and twitter structured tags[[7]](#footnote-7). Finally, the review column was converted to lowercase, then stemmed using PortStemmer[[8]](#footnote-8) before TFIDF was applied[[9]](#footnote-9).

**Results**

A wordcloud was generated on the lie column for both truthful and untruthful reviews. Word similarity between the two groups indicate additional stopwords can be appended to the default nltk english stopwords[[10]](#footnote-10). Namely, the words “nan”, “restaurant”, “food”, and “place” could potentially be added to a custom stopword list.

|  |  |
| --- | --- |
|  |  |
| **Figure 1:** wordcloud on truthful reviews | **Figure 2:** wordcloud on untruthful reviews |

Similar to earlier wordclouds on truthfulness (i.e. lie), the sentiment wordclouds indicate a familiar pattern. Specifically, the same custom stopword list could be created and utilized.

|  |  |
| --- | --- |
|  |  |
| **Figure 3:** wordcloud on negative sentiment | **Figure 4:** wordcloud on positive sentiment |

A series plot was created for lying and truthful sentiment. Though the neutral series between the two sentiments follow a similar pattern, later fluctuations indicate higher neutral scores for the lying category. Furthermore, lying sentiments generally have higher positive sentiment.

|  |  |
| --- | --- |
|  |  |
| **Figure 5:** series plot of lying sentiment | **Figure 6:** series plot of truthful sentiment |

Overall scores for the combined modals are provided for completeness.

|  |  |
| --- | --- |
|  |  |
| **Figure 7:** overall wordcloud. | **Figure 8:** overall sentiment plot |

The Bernoulli Naïve Bayes (BNB) has the best accuracy for lie detection, followed by the Support Vector Machine (SVM), then the Multinomial Naïve Bayes (MNB). In general the latter two models hovered at roughly 50% accuracy, indicating an inability to distinguish lies from reviews.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 9:** Bernoulli Naïve Bayes on lie detection. | **Figure 10:** Support Vector Machine on lie detection. | **Figure 11:** Multinomial Naïve Bayes on lie detection. |

The BNB model with POS suffix, outperformed the SVM and MNB, each also implementing the POS suffixes. The latter two models were roughly 50% accurate, not differing significantly than chance. Again, the SVM and MNB cannot distinguish lies from reviews, when each word is suffixed by the corresponding part of speech (POS).

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 12:** Bernoulli Naïve Bayes on (POS) lie detection. | **Figure 13:** Support Vector Machine (POS) on lie detection. | **Figure 14:** Multinomial Naïve Bayes (POS) on lie detection. |

A Bernoulli Naïve Bayes outperforms the SVM and Multinomial Naïve Bayes for sentiment analysis. However, unlike the lie detection results, all models performed relatively well. Specifically, the BNB performed best, followed by SVM then MNB.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 9:** Bernoulli Naïve Bayes on sentiment analysis. | **Figure 10:** Support Vector Machine on sentiment analysis. | **Figure 11:** Multinomial Naïve Bayes on sentiment analysis. |

Both BNB and SVM equally exceeded prior results for sentiment analysis using POS suffixes. However, the MNB still performed well, and comparable to the equivalent non-POS classifiers.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 12:** Bernoulli Naïve Bayes on (POS) sentiment analysis. | **Figure 13:** Support Vector Machine (POS) on sentiment analysis. | **Figure 14:** Multinomial Naïve Bayes (POS) on sentiment analysis. |

Overall, the below bar graphs reinterpret earlier classifier scores. Specifically, both categories witnessed the BNB being the top performer, followed by the BNB pos. However, for the sentiment analysis, the SVM and SVM pos scored with same relative strength to the BNB. Moreover, the sentiment analysis classifiers outperformed each corresponding lie detection classifier. This is an indication that sentiment analysis may be an easier task to train. Though more data, or another topic of categories can further prove this, in general there is less ambiguity is sentiment analysis. Specifically, there is a greater challenge with labeling the truthfulness of a review compared to determining a sentiment. Therefore, it is expected that corresponding trained classifier will represent these shortcomings.

|  |  |
| --- | --- |
|  |  |
| **Figure 7:** lie detection classifier accuracy. | **Figure 8:** sentiment classifier accuracy. |

Given the earlier results, it is appropriate to perform the k-fold validation to indicate the general performance and variability of a model. However, a different argument can also be made to perform the validation prior to model generation. Overall the MNB contained scores higher than BNB and SVM. Furthermore, the SVM contained the lowest scores, while maintaining the least variability between each k-fold. The BNB scores were almost comparable to the MNB, with greater variability. Overall the MNB was the best performing model, since the lowest score was roughly equal to the highest SVM score.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 9:** Bernoulli Naïve Bayes on lie detection. | **Figure 10:** Support Vector Machine on lie detection. | **Figure 11:** Multinomial Naïve Bayes on sentiment analysis. |

The MNB seemed to outperform the BNB and SVM. Specifically, each score was roughly 80% or higher, with a small exception on k-fold=5. Next, the SVM contained scores comparable to the BNB, while indicating less variability. Though the variability of the MNB was higher than the other two models, the lowest scoring fold was higher than most of the higher scoring folds for the other models respectively.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 9:** Bernoulli Naïve Bayes on sentiment analysis. | **Figure 10:** Support Vector Machine on sentiment analysis. | **Figure 11:** Multinomial Naïve Bayes on sentiment analysis. |

Overall, the k-fold validation generally agreed with earlier predictive results. Specifically, sentiment analysis is an easier task to perform. However, the resulting validation over the five folds, indicate that MNB is generally preferred for both modal. An interesting follow-up study would include a higher k-fold (possibly 10), while also including a custom stop word list. Specially, visual findings from earlier word clouds show common words across the different categories. This is an indication of noisy data, providing no additional benefit for the model to characterize between labels. By removing these words, less apparent words have more weight, providing better predictive capabilities.

**Conclusions**

Sentiment analysis is a task often utilized with text mining. The ability of predict sentiment from a given text is often invaluable for organizations. By integrating a feedback system from customers, an organization can determine preferences for various products. Moreover, having this knowledge can better prepare a supply chain by ensuring products are properly distributed across warehouses.

Though lie detection is often more difficult to ascertain than sentiment, the added information can supplement decision factors. For example, if a lie detection classifier has high predictive qualities, then noisy data (untruthful reviews) can be omitted from successive analysis. This allows descriptive analysis as well as classifier(s) to generalize and increase awareness. Overall, utilizing both lie detection, and sentiment analysis for reviews, can provide functional information needed for decision making.

1. <https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/> [↑](#footnote-ref-1)
2. <https://www.brandwatch.com/blog/facebook-statistics/> [↑](#footnote-ref-2)
3. <https://www.itchronicles.com/technology/data-the-new-gold-rush-for-businesses/> [↑](#footnote-ref-3)
4. <https://github.com/jeff1evesque/ist-736-hw/blob/master/data/deception_data_converted_final.csv> [↑](#footnote-ref-4)
5. <https://github.com/jeff1evesque/ist-736-hw/blob/master/hw4/utility/dataframe.py> [↑](#footnote-ref-5)
6. <https://github.com/jeff1evesque/ist-736-hw/blob/master/hw4/assignment.py#L44-L48> [↑](#footnote-ref-6)
7. <https://github.com/jeff1evesque/ist-736-hw/blob/master/hw4/algorithm/text_classifier.py#L70-L81> [↑](#footnote-ref-7)
8. <http://www.nltk.org/howto/stem.html> [↑](#footnote-ref-8)
9. <https://github.com/jeff1evesque/ist-736-hw/blob/master/hw4/algorithm/text_classifier.py#L83-L88> [↑](#footnote-ref-9)
10. <https://pythonprogramming.net/stop-words-nltk-tutorial/> [↑](#footnote-ref-10)