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 format, 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.

**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 kfold 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 Support Vector Machine (SVM) appears to have the best result for lie predictions. However, results are heavily unbalanced. Therefore, the Bernoulli Naïve Bayes seems the best generalized model.

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

Multinomial Naïve Bayes (POS) outperforms the Binomial and SVM equivalent. Moreover, the Multinomial Naïve Bayes (POS) performs better than the above Bernoulli Naïve Bayes.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **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.

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

The best performing POS classifier on sentiment analysis indicate less than a 50% accuracy. Therefore, the POS variants do not perform well for classifying sentiment.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **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 ensembled scores, which did not exceed 50% accuracy. Moreover, the best performing classifiers generally do not exceed 65% accuracy. It appears that the overall scores are roughly equal between sentiment and lie detection. However, lie detection scores are more variable between implemented classifiers, whereas the sentiment classifiers indicate less score variability. In general, sentiment analysis is an easier task than lie detection, since characterizing sentiment has less ambiguity.

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

The Bernoulli Naïve Bayes and Multinomial Naïve Bayes have higher k-fold validation scores compared to SVM for lie detection. However, SVM exhibits less accuracy variability.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **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 Bernoulli Naïve Bayes and Multinomial Naïve Bayes k-fold validation outperforms SVM for sentiment analysis. However, the Bernoulli Naïve Bayes is slightly more variable.

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

**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)