Diego Valdes

IST 736

Home Work 7

11/19/2019

**Introduction**

“Why would anyone do that?” is an often uttered reaction to a surprising human action. It is a sentiment that everyone can relate to and provide a personal story. The reason for these types of reaction is that human beings are—human. As much as logic and the desire to make sense of things is part of our nature, irrationality is also a core component of our existence. We cannot simply do things because they make sense. Often (and there is no data on this to reference), the actions humans take are based in emotion, regardless of how fleeting it can be, or how little sense it makes.

If humans make most of their decisions based on gut instincts, then for any algorithm to be effective in dealing with humans, it must somehow take this irrational part of mankind into account. It may be easy to graph and plot numbers, but how does one plot happiness? Can fear be a variable on a heat map? Is love something that can be explained with a normal distribution? Can an emotion be ranked and be part of a linear model? Well, while the answer to most of these questions is no, a better answer may be, not yet.

Sentiment analysis is a crucial part of the predictive analysis driving decisions today. Most of the data being generated today is not numbers that can easily be broken up into third normal form. It’s unstructured, ugly, and filled with text that can range in subject and feeling. Data is used to make sense of the world, but if that trend is to continue, and greater insights and capabilities obtained, deriving the meaning behind our words is the next step in understanding.

**Analysis**

**About the Data**

The datasets are a collection of sentences. The training set has a sentiment value between 0 and 4.

The test set does not have the sentiment value. To verify the results of the test set, a submission of results must be submitted to Kaggle.com. The sentences vary in length, from complex sentences to instances of a single word. There are also instances of punctuation in the dataset, all of which were given a value of 2. Slightly more than 50% of the data falls in the neutral category.

|  |  |  |
| --- | --- | --- |
| Value | Sentiment | Distribution |
| 0 | Very Negative | 4141 |
| 1 | Negative | 16449 |
| 2 | Neutral | 47718 |
| 3 | Positive | 19859 |
| 4 | Very Positive | 5468 |

The datasets were vectorized in preparation for analysis in SVM and MNB.

**Models**

Multinomial Naïve Bayes and Support Vector Machines was used to model the data and make sentiment predictions. A 60/40 split of the training test set was initially used to build and test the models. After examining the results and tuning parameters, a model was built with 100% of the testing data and tested on the testing set.

**Results**

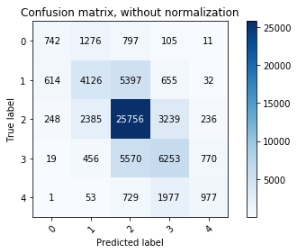
**Multinomial Naïve Bayes**

Multiple variations of count vectorizer were implemented for training. Each model was tested against the test set, a list of the top very positive/negative words, and metrics were examined. There was no significant difference in the performance of the model based on the type of count vectorizer used. As an example, the top 10 negative words were common with each model, but the conditional probabilities were different. In some cases, this difference was enough to put some words out of order, but not knock words off the top 10.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Negative** |  |  | **Positive** |  |
| **time** | -5.9416 |  | **performance** | -5.81757 |
| **characters** | -5.93102 |  | **comedy** | -5.76994 |
| **minutes** | -5.92054 |  | **great** | -5.72448 |
| **story** | -5.92054 |  | **story** | -5.69528 |
| **comedy** | -5.91018 |  | **performances** | -5.64615 |
| **just** | -5.6891 |  | **good** | -5.37226 |
| **like** | -5.13779 |  | **funny** | -5.23159 |
| **bad** | -4.9755 |  | **best** | -5.13669 |
| **film** | -4.8324 |  | **movie** | -4.76046 |
| **movie** | -4.32158 |  | **film** | -4.25303 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **0** | 0.46 | 0.25 | 0.33 | 2931 |
| **1** | 0.5 | 0.38 | 0.43 | 10824 |
| **2** | 0.67 | 0.81 | 0.73 | 31864 |
| **3** | 0.51 | 0.48 | 0.49 | 13068 |
| **4** | 0.48 | 0.26 | 0.34 | 3737 |
|  |  |  |  |  |
| **micro** | 0.61 | 0.61 | 0.61 | 62424 |
| **macro** | 0.52 | 0.44 | 0.47 | 62424 |
| **weighted** | 0.59 | 0.61 | 0.59 | 62424 |

Accuracy for the model was slightly over 60%, considering that most of the data is neutral (~50%) these results are just slightly better than random guessing.

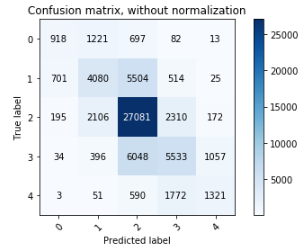


**Support Vector Machines**

As with Naïve Bayes, many different forms of count vectorizer were used to train a model. Each model was tested against the test set (60/40), a list of the top very positive/negative words, and metrics were examined. Unlike with Naïve Bayes, the list of words changed more so with SVM. Different implementations of count vectorizer produced fairly different very positive/negative top 10 word lists. One example is provided. Metrics also had slight range differences, but generally maintained in the 60% range with the highest accuracy being 65%.

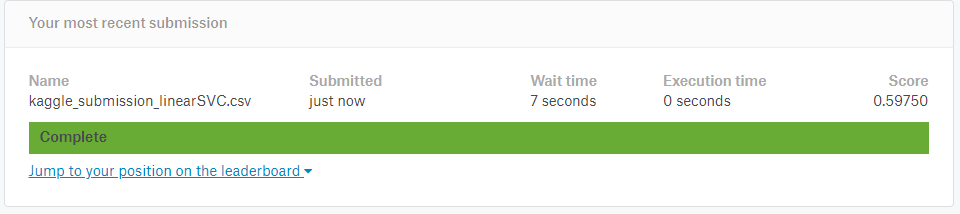
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Negative** |  |  | **Positive** |  |
| **cesspool** | 1.62161 |  | **stunning** | 1.563529 |
| **disappointment** | 1.648488 |  | **astonish** | 1.60058 |
| **pompous** | 1.659249 |  | **refreshes** | 1.610813 |
| **stinks** | 1.668369 |  | **flawless** | 1.61489 |
| **distasteful** | 1.692774 |  | **phenomenal** | 1.647465 |
| **unwatchable** | 1.695591 |  | **masterful** | 1.650642 |
| **unbearable** | 1.75264 |  | **masterfully** | 1.677616 |
| **stinker** | 1.787357 |  | **glorious** | 1.878142 |
| **disgusting** | 1.82287 |  | **miraculous** | 1.980188 |
| **worthless** | 1.823306 |  | **perfection** | 2.014325 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **0** | 0.5 | 0.31 | 0.38 | 2931 |
| **1** | 0.52 | 0.38 | 0.44 | 10824 |
| **2** | 0.68 | 0.85 | 0.75 | 31864 |
| **3** | 0.54 | 0.42 | 0.48 | 13068 |
| **4** | 0.51 | 0.35 | 0.42 | 3737 |
|  |  |  |  |  |
| **micro** | 0.62 | 0.62 | 0.62 | 62424 |
| **macro** | 0.55 | 0.46 | 0.49 | 62424 |
| **weighted** | 0.6 | 0.62 | 0.6 | 62424 |



Naïve Bayes and SVM both performed similarly in accuracy, but the very positive/negative words vary within both models. 50% of the training data was neutral (2), and simply always selecting 2 would provide an accuracy of 50%, which means both models are performing slightly better than guessing.

A final model was tuned and trained with the entirety of the training data and tested on the test set for Kaggle submission. The results from this submission matched the general accuracy of models trained with a 60/40 split of the training data.



**Conclusion**

The nature of language makes it difficult to analyze and derive meaning. No better example can be found that examining text messages, tweets, and Facebook posts where the literal reading of the sentence is not the true meaning. A complicated example would be the use of the most notorious ‘N’ word in the English language. How would a computer determine when the usage is friendly vs offensive when Americans have been struggling with it for over a hundred years.

Both models performed slightly better than random guessing. In previous analysis, SVN performed better than Naïve Bayes and had better accuracy. Those results were generated on data with binary results. The measure of how negative or positive a sentence may be a cause for the less than spectacular results in this analysis.

While these results may not seem promising, better than guessing is a good start for a computer. The issue may not be in the model, but in the amount of data. While adjusting parameters may increase accuracy, nothing will improve a model better than more data to learn from.