Twitter Sentiment Classification Analysis: Group COP

Oliver Muellerklein, Chris Lin, Paul Land

April 29, 2016

1 Introduction

Why Twitter Sentiment?

A twitter sentiment classification model could be useful for determining the public's sentiment towards ideas, events, businesses, products, political figures, and many other possibilities. Interested parties can take tweets from Twitter and use a sentiment model to assess the outlook of their interests or the progress of their endeavors.

Our Goal (Model)

1.1 Overview Here we describe the purpose of analyzing this. Why is it important to understand this data? What do we gain from analyzing this data?

What do we gain by being able to predict with this data? Think big picture here.

Give overview of what we are trying to solve in one paragraph (e.g. we will perform THIS TYPE OF ALGORITHM as well as THIS TYPE OF METHOD to determine if the RESPONSE can be predicted as 0 or 1 - below we will give an overview of our data cleaning, feature engineering, and feature and model selection processes)

2 Data, Cleaning, Exploration

Introduction

The raw data contains 4 variables: `ItemID`, `Sentiment`, `SentimentSource`, and `SentimentText`. There are 1578627 total tweets. There are 765127 positive tweets, 763500 negative tweets, and 50000 unlabeled tweets.

Cleaning

We only had one issue loading the raw data (`MaskedDataRaw.csv`) into `R`. We figured out that some tweets contained unmatched double quotation marks, which caused the `read.csv` function to misorganize the `.csv` file's information as it was loaded into `R`. We used a `regex` pattern to find and remove the unmatched quotes. Once we solved this issue, we were able to move on to the data exploration and feature engineering parts of our analysis.

Exploration

We used numerous `regex` patterns to explore the tweets.

Feature Engineering

Username/Hashtag Positive/Negative Score Features After we loaded the raw data into `R` and explored the tweets, we began the feature engineering process. We focused on using the strings of tweets to extract useful information and construct features. We did not use the `ItemID` and `SentimentSource` variables in the raw data. The first set of features we constructed were based on the frequency of usernames and hashtags in the tweets. We wanted to explore the possibility that certain usernames/hashtags tend to appear in more positive or more negative tweets. To build those features, we first divided the data set into a set of positive tweets and negative tweets. For each set, we calculated the frequencies of each unique username and hashtag. For example, `@mileycyrus` appeared `2833` times in the positive tweets. With this information, we gave each username and hashtag a positive and negative score. The scores were calculated by dividing each username's/hashtag's frequency by the postive/negative set's highest username/hashtag frequency. Finally, if a particular username/hashtag appeared in a tweet, we assigned the tweet a corresponding username/hashtag score. Ultimately, this process created `4` features: positive username score, positive hashtag score, negative username score, and negative hashtag score.

The top 5 usernames found in the positive tweets:

Username	Frequency
@mileycyrus	2833
@tommcfly	2073
@ddlovato	2001
@DavidArchie	902
@Jonasbrothers	860

The top 5 usernames found in the negative tweets:

Username	Frequency
@tommcfly	1480
@mileycyrus	1339
@ddlovato	1255
@DonnieWahlberg	504
@mitchelmusso	416

Bag of Words Features We also created a large set of bag of words features. We constructed a positive corpus and a negative corpus. From these corpuses we determined which words appeared in the tweets most frequently. We took those words and created a large frequency matrix where each column in the matrix represented the frequency of a word.

Here we dive into our approach. First steps with data. Maybe we used method X and then method Y to explore and then clean data.

2.1 Method X Method X described above. Here we maybe show table of exploratory results and/or math equations etc.

2.2 Method Y Method Y described above. Here we maybe explain how we cleaned the data and what we found.

3 Cross Validation and Model Formation

3.1 Feature Selection Overview Through various feature importance and selection methods, we found that a majority of the loss reduction within random forest and boosting methods (via XGBoost) was captured by a small subset of our total features. For example, we approached the problem initial with 50 features from the respective word frequencies via Bag-of-Words and four normalized features corresponding to usernames and hashtags that occurred most frequently with both positive and negative tweets. Feature selection in both random forest and gradient boosting (XGBoost) dramatically selected for the four normalized username and hashtag predictors. Furthermore, the same pattern existed when increasing our Bag-of-Words term frequency usage from 50 to 118 words. We had also found that our initial models - including logistic regression with regularization, random forest, and gradient boosting tree ensemble (XGBoost) - generally obtained significantly greater accuracy and decreased misclassification for positive than negative tweets. For example, one method produced an accuracy above 80% for positive and just over 50% for negative tweets. As expected, when we compare the negative prediction accuracy and misclassification rate to random guessing we see minimal to no benefit.

One possible explanation for the discrepancy in sentiment prediction may be due to feature engineering bias. The total number of positive and negative tweets was nearly equal in the original dataset. However, our Bag-of-Words and Term Frequency - Inverse Document Frequency methods may not have captured the vast array of word choices for negative tweets as we simply took words that occurred in at least 1% of all tweet words. This rests on the assumption that single word use and frequency either 1) give a general, accurate account for tweet sentiment differences and 2) the underlining distributions of positive and negative word counts and frequencies are equal. The first point could be explained by claiming that some n-gains or set of words would give stronger features. The second point comes from the statistical conceptualization of finding signal within noise of data. To elaborate, in order to use the word frequencies and differences between positive and negative tweets from the words that appeared in at least 1% of tweet words, we must assume that such words are a valid subsample. Thus, we are assuming that the 118 words that occurred in at least 1% of the total tweet words are an accurate representation of both the actual word frequencies (for all words) in all tweets and the differences in usage between positive and negative sentiment.

On the other hand, we used normalized username and hashtag occurrence scores for positive and negative features. These four features were far less sparse than the Bag-of-Words and TF-IDF generated features - i.e. far less observations with 0. This can bias the importance of those features in the decrease in misclassification (loss, similar principle can apply for other forms of optimization) for each node split.

3.2 Overview of TF-IDF Method We can apply the TF-IDF principle to the top occurrences of both usernames and hashtags to get the corresponding sparse features in the same format as the 118 word frequencies. We determined the number of important usernames and hashtags by analyzing the total number of unique values, e.g. unique usernames in all tweets, and finding the total number of occurrences per element. From this data exploration method we found that the username @mileycirus appeared as the second most frequented tweeted username in all negative tweets. The top three negatively tweeted usernames appeared more than 1,000 times but dropped significantly by the fourth or fifth most population negative username. We were able to pick the top 50 usernames based on occurrences in positive, negative, or all tweets. Similarly, we found the top 50 hashtags. These top users and hashtags were fed into the TF-IDF algorithms we created as terms and their corresponding sparse predictors were augmented to the data as features.

From the 118 word frequencies, 50 usernames, 50 hashtags, and an additional 45 *special terms* - such as special characters that do not necessarily appear often but are consistent with either positive or negative tweets - we created a sparse matrix of 263 (or 264 or 265??) predictors. This was fed into Random Choice, Logistic Regression, Support Vector Machines (SVM), and Gradient Boosted Tree Ensembles (via XGBoost).

4 Ensemble

?

5 Model Analysis and Fine Tuning

5.1 Feature Selection and Tuning We performed several optimization and parameter tuning tricks, including:

1. various train / test splits of the labeled (sentiment of 0 or 1) data from 50% / 50% to 90% / 10%
2. early stopping (XGBoost) when loss is not improved for 10-50 iterations of boosting - this is an optimal method to extracting optimal iteration number while preventing overfitting
3. K-means clustering on relative feature-based gain (or loss) - this is particularly useful for *post facto* feature selection, augmentation, and further ensemble-level engineering
4. K-folds Cross-Validation for feature and model selection - this is performed by randomly extracting the training dataset into K number of subsets, training the model on that subset, and then testing on the remaining training dataset

6 Future Ideas

There are several steps and ideas we had for future direction with this project. The two most obvious are to i) redefine our term use in the TF-IDF algorithm and ii) further our feature and model selection and tuning methods towards an exhaustive search of both. The former idea could be done by adjusting our idea of *words* and redefining the terms we use to be some set of characters or relative word groupings, perhaps through an n-gram approach. The latter idea could be done through the use of the *Caret* library in *R* to perform an exhaustive grid search. Another approach could be to perform adaptive solvers either *post facto* or during iteration boosting - such as particle swarm optimization or other forms of evolutionary algorithms. These approaches may be able to adaptively solve for the optimization problem of misclassification or accuracy gain / loss in a non-parametric fashion. I.e. such adaptive solvers in optimization problems do not assume distributions or relationships between parameters and can be used to search through the feature dimensional space for non-linear solutions. This could of course be performed in a different route with kernel based approaches of dimensional reduction, such as kernel-PCA and kernel-SVM (NOTE: some of these approaches were touched on or implemented fully in our report above).

END