Classifying Blog Posts Using Natural Language Processing

By Johnny McGregor

Problem Statement

I approached this project as if I worked for a hypothetical website that aims to curate science related content. A lot of the content can be generated by users of the site. At the beginning, it is plausible that a human can check over each user submission and decide whether to display the content on the site. Once the traffic increases, however, there needs to be a way to automate the process. My goal here was to develop a classification model to determine whether a post is appropriate for a science website or not.

Data Gathering

In order to get content to train and test my model, I web scraped posts from Reddit. I gathered roughly 1500 posts each from science related subreddits and comedy related subreddits. I used the Beautiful Soup python library to accomplish this. I also imported the time library so that I could institute a one second delay after each iteration of the data-gathering for loop so that Reddit didn't think there was a security issue as my program repeatedly accessed the site's HTML. Finally, to ensure that my dataframe did not include any duplicate posts, I ran this line of code – unique_df = df[~df.duplicated()].

Once I had all my data, it needed to be cleaned before I ran any natural language processing techniques. I wrote a function that took care of this task in five steps: First, I used the get_text() method in Beautiful Soup to grab only the relevant text from the json file in which the post was nested. Second, I applied regex to the text to extract the letters and discard punctuation and other extraneous symbols that may have been included in the original post. Third, I made all the text lowercase with the lower() method and separated everything into single words with the split() method. Fourth, I removed "stop words" from the post. These are words that are so common that they would not have any predictive power in determining the subject matter of the post. I, you, me, we, and, or, etc. are just a few examples of stop words. The nltk.corpus python library has a stop words list.

Once I imported the stop words from that library, I was able to run a list comprehension to eliminate them from each post as follows:

words = [w for w in words if w not in stopwords.words('english')]

After completing all previous steps, I put the remaining individual words from the post back together using a join() method.

Modeling

Now that I had a dataframe of cleaned posts, I could begin modeling. I experimented with a count vectorizer as well as a term frequency – inverse document frequency (TF-IDF) algorithm to create features out of the words in each post (natural language processing). I fit quite a few classification models using both algorithms and compared the results. These models included LogisticRegression, MultinomialNaïveBayesClassifier, RandomForestClassifier, AdaBoostClassifier, and BaggingClassifier.

The best performing model was a Logistic Regression model using a count vectorizer. After utilizing grid search, I set the maximum document frequency for the count vectorizer at .1 and max features at 5,000. The C parameter of the logistic regression model was set at 3. It achieved an accuracy score of .976. Of the 717 posts in the test set, there were just 3 false positives and 14 false negatives.

Conclusions / Recommendations

This project was essentially my introduction to classification problems in machine learning. I know that in reality getting an accuracy score this high is unrealistic. If I were to go back and challenge myself a little more, I probably would not have chosen two subjects that are as easily separable as science and comedy. When contemplating classification metrics for which to optimize, I think it is appropriate to keep the false positives to a minimum. If a post or two does not make it onto the site because the classifier mistakenly determined it to be irrelevant, that is preferable to the classifier letting inappropriate posts onto the site. To account for this, I would raise the threshold for a science classification higher than the default probability of .5. I would look at all the posts misclassified as science and

see what the predicted probabilities were for each of those posts and adjust the threshold accordingly.