Jacob Dineen

Text Mining

9/2/2018

Homework 7

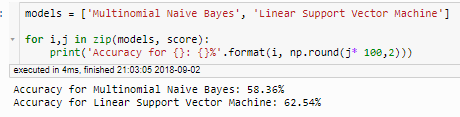
**Overview**

This week we are continuing our use of the popular Kaggle dataset referencing a collection of movie reviews – A precompiled/split dataset that contains ~156k observations in our training set, and ~66k in our test set. With an arbitrary cut point in place, we will be following a classic heuristic for model fitting by having a train set, a validation set and a test set. The test set contains unlabeled examples that we will be inferring based on our fitted models. We will be testing and comparing the algorithmic strength of two popular machine learning algorithms in the field of text mining/computational linguistics – Linear/Nonlinear max margin support vector machines and Multinomial Naïve Bayes – alongside different, specified vectorization parameter tuning. We are dealing with a multi class classification problem, and as such, our majority vote baseline changes. Random vote is set to 20% accuracy, while the majority vote benchmark is ~31% - This also suggests that our dataset may be inherently biased towards the neutral class. We’ll see what our errors look like later.

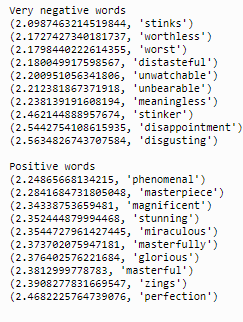
**Task 1**

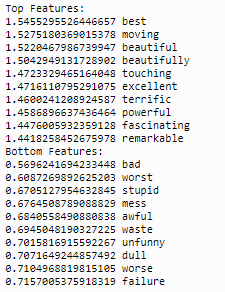
The prompt asks us to construct models with unigrams as a vectorizer parameter but doesn’t specify whether to use a Boolean/Count/TFidf vectorizer. I’ve chosen to go with a TFidf vectorizer because we are dealing with a massive corpus that should lead way to us being able to use a more advanced understanding of our representation – I say this because we’ll have such a large collection of text that we will likely have words occur in many documents, but just because they are occurring doesn’t add intrinsic value to a function mapping them to a respective class, and by computing and dividing by the inverse document frequency we’ll be able to weigh down the less important terms. We’ll start with standard vectorization parameters, removing stopwords and setting our mindf to 5 to try and reduce the dimensionality and resulting sparsity of our matrix.

These vectorization parameters are run through both algorithms, resulting in the following:

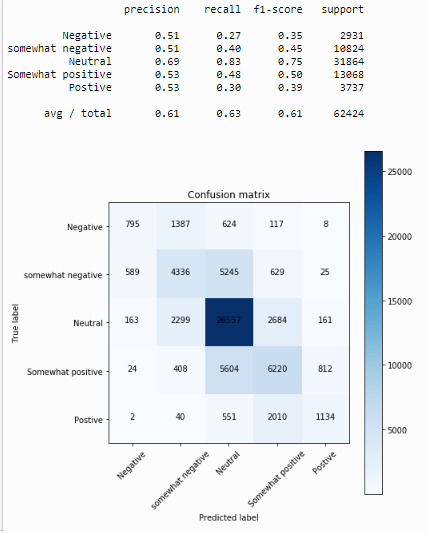
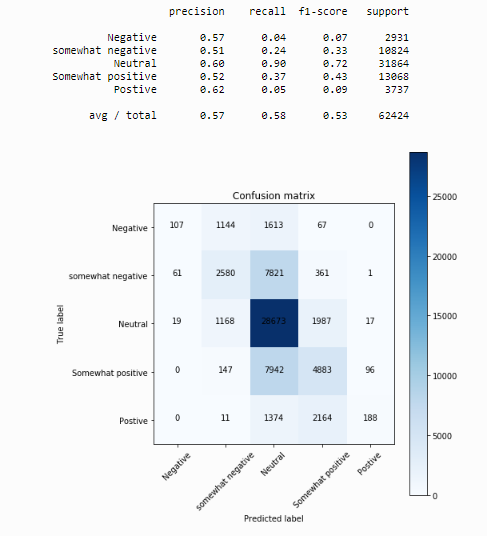


Our linear SVM, with no hyperparameter tuning, outperformed the multinomial naïve bayes model on inference. We now look at what our model learned – One of the most informational exercises is to express feature importance through our SVM coefficients and our MNB odds ratios. As the SVM requires a different approach to dealing with multiclass problems (one v one/one v all) we need to look specifically at the positive and negative classes. With the MNB we compute our odds ratios by taking dividing the log probabilities of words in each respective class. The below snippets show that our model appears to be able to distinguish the difference between positive and negative words – It’s interesting our model didn’t reflect a better ability to generalize, but again, we have a dataset that skews toward the neutral class.

**SVM results MNB Results**



**SVM results MNB Results**

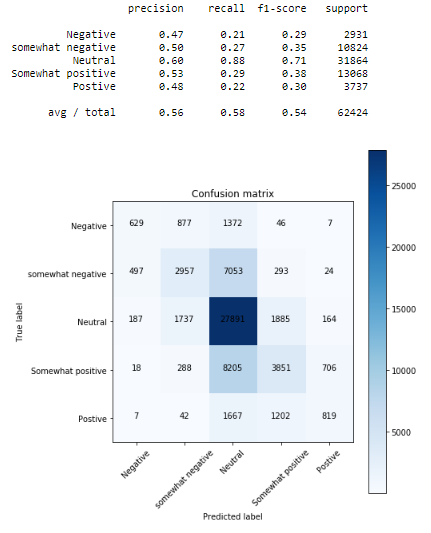
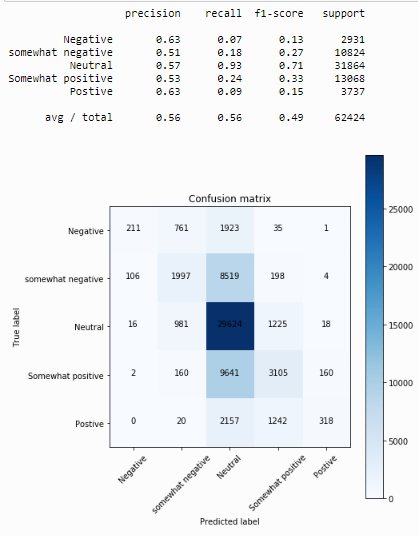


The confusion matrices above help us to evaluate our error terms. The MNB model was able to minimize false positive (precision) to an extent, but mightily struggled when it came to rejecting instances that belonged to a specific class (exceptionally low recall). As hypothesized above, it appears that the neutral skew may be influencing our results – Many of the errors come on the model’s inability to decipher a ‘somewhat’ class from a neutral class. This could also come down to subjective labelling. The linear SVM, on the other hand, was able to curtail those false negatives to an extent, suggesting we were able to linear separate the high dimensional representations. We note that the f1 scores were the lowest on the classes furthest from the center, but also notice that those were the least supported class in terms of class representation.

**Task 2**

Running the above experiment with the modification that we allow for bigrams proves to be a futile effort. What was noticeable here was the fact that our feature ranking failed to include many bigrams, suggesting that those vectors were too sparse too lend a hand to gained information. Curious, I decided to eliminate unigrams entirely. MNB performance was stagnant across all three experiments, but removing unigrams severely impacted the SVMs ability to generalize.

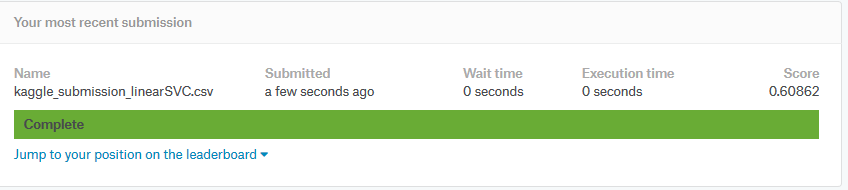
**SVM results MNB Results**



Again, we see the one number that stands out is the Neutral class’s recall.

**Task 3 Bonus**

I built a linear SVM model with 5fold cross validation that achieved 70% accuracy on the train set, but couldn’t generalize well to the Kaggle test set:



**References**

[**http://www.tfidf.com/**](http://www.tfidf.com/)

**Appendix**