

CIS 365 – Project 2 – Sentiment Analysis

The first goal with the project was to do some initial cleaning of the movie reviews. Unknown characters and HTML tags were littered all throughout the reviews and it was obvious that they would detract from determining whether a given review had positive or negative sentiment. From this, a separate file was created with all HTML tags and any unknown characters removed, which was saved to remove the overhead of having to perform the task on every iteration of training.

With this cleaned dataset, several different avenues were explored for further cleaning. The first was to use the Natural Language Toolkit (NLTK) package to remove all the meaningless words such as ‘the’, ‘a’, ‘is’, etc. These words are frequently used throughout both positive and negative reviews and because of this, they lack much, if any, meaning. The NLTK can also stem words with its PorterStemmer, which removes suffixes from words. This was done to make words like ‘running’ into the word ‘run’ because both really mean the same thing and now the model will treat them as the same.

This was the pretty much the extent of our preprocessing, but we also added in a simple regular expression to remove most punctuation and all digits. Throwing this data into a pipeline that had a term frequency – inverse document frequency vectorizer and used logistic regression as the classifier with only default parameters we got around an 89% accuracy. Hold-out cross-validation was used to determine the model’s accuracy and a split of 80% data to train on and 20% data to test on was used.

Instead of logistic regression, stochastic gradient descent was used as the classifier. Stochastic gradient descent (SGD) has a plethora of different loss functions to try and more parameters to tweak than logistic regression did. After utilizing Scikit-Learn’s GridSearchCV to hunt for the best parameters we found that ‘modified_huber’ was the best loss function – this by itself got us to around 90.2% accuracy. Some of the other parameters were changed, but only granted very marginal increases.

The next step was to try and optimize the TfidfVectorizer’s parameters and the same setup with GridSearchCV was used. It was found that only looking at features that weren’t in more than 5% of the documents was optimal, but they would have to be in at least two documents. This meant that we would ignore a lot of common words that did not get removed by NLTK’s stop words, but also that it would ignore things like specific names that were only mentioned once. The biggest change though was considering both single words as a feature along with bigrams of words. With these parameters, the model scored around a 91% accuracy on the test data.

The random number generator that shuffled the data before being trained was statically seeded throughout all of training. Originally, the idea was to be able to reproduce the results so that the effect of each change would be more obvious. The issue arose when it was seeded with a random number its accuracy fluctuated greatly, sometimes being close to 94% and other times

being in the low 60%. By choosing a static seed for the random number generator all the parameter tuning was essentially done to get the best performance on the initial test set. It was essentially over-fitting on the test data.

To solve this, cross-validation was used with 10 folds and from it an average score of 86% was achieved. With all the preprocessing of the data that was done 86% seemed extremely low, so it was decided to see how the model performed without removing stop words or doing any stemming. Consistently, the model would score an average accuracy of 91.5%, which was much better.

The theory of why it scores so much higher with less preprocessing is because of the inclusion of bigrams as features. Bigrams allowed for phrases such as ‘not funny’ or ‘not good’ to retain their negative meaning. Without bigrams, these phrases would be more likely to be classified as positive statements because the words ‘funny’ and ‘good’ could carry more weight than the word ‘not’ because it is commonly used. By preprocessing the data with the removal of stop words the actual meaning of the bigrams were being changed.

In conclusion, the main enhancements to the increasing the accuracy of the model came from removing HTML tags, going from logistic regression to stochastic gradient descent, and primarily from the addition of bigrams. Through a 10-fold cross-validation it was found that the 10 trained models from this averaged around a 91.5% accuracy over multiple training cycles. Based off this, it can be reasoned that the model now trained on all of the data provided should also perform close to metric for data that it has never seen before.