|  |
| --- |
| CIS 4500 |
| Movie Review Classification |
| Language Processing |

|  |
| --- |
| Nolan Mullins  11-13-2019 |

Introduction

Over the course of this paper we will be looking at the text-based classification and sentiment analysis problems. The goal of text-based classification is to assign predetermined categories to a given document. Companies use this strategy to organize data, analyze content and provide high level information about a given body of text. The other part of the problem we’ll be looking at is sentiment analysis where the goal will be to understand the emotion being conveyed in a given document. Some of the challenges with sentiment analysis are determining how the emotion is being conveyed towards a given topic. For example, we could see the sentence “I loved the popcorn but didn’t enjoy the movie”, here sentiment analysis will need to recognize the difference in context. The problem we will be dealing with is movie review sentiment analysis. We will be analyzing 2000 pre classified movie reviews to build a model that can predict whether a given review about a movie is either positive or negative. The reviews we will be analyzing consist of 1000 positive and 1000 negative movie reviews. The even ratio will keep our data balanced which will aid our machine learning algorithm approach. Some other interesting information about our review data is we have 39659 unique terms over the 2000 documents. Out of those terms we have 26127 unique stems which refers to the root of the word, Ex: “Likes” “liking” and “Liked” all share the same stem “Like”. Our largest review is 2294 words and our smallest is on 17 words long with the average document being 629 words. When looking in to the reviews many of the reviews contain information related to the plot which will test our sentiment analysis as we could come across “The murder scene was horrific” and “The movie was horrible”. We will be approaching this problem using machine learning to build a model that will be able to recognize these patterns with a high degree of accuracy.

Method

Our main goal is to build a model that is able to predict whether a movie review is positive or negative, to do this we will compare various different methods and how they work together. Before we get into selecting these methods we must first break down the problem in to 3 major steps.

**First**, we will need to apply preprocessing in order to get the data into a form that we can work with. This requires tokenization, removing unnecessary data such as stop words and punctuation, converting the text to lower case, and stemming each word in the document. According to (Song, Liu, & Yang, 2005) the effectiveness of stemming and stop word removal are minimal but do reduce the model size and increase efficiency. Another paper by (Toman, Tesar, & Jezek, 2006) stated that stop word removal benefited the text-based classification accuracy but the stemming actually had a negative effect on the accuracy. For this reason, I will not be stemming in the preprocessing step. Lastly in preprocessing we could generate bigrams and trigrams which will combine pairs / trios of words together to form new terms. For the sake of efficiency and compute power I will only be using unigrams for the comparisons (Single words).

**Second**, feature selection. The goal of feature selection is to determine which terms in a given dataset are the most significant and will contribute the most to each label. In our case we will be looking for terms that are largely associated with either positive reviews or negative reviews. Terms that frequent both categories will not help up much in the classification step coming up. For feature selection we will be comparing 2 different techniques selecting the K best features and selecting the p-values corresponding an alpha cutoff (SelectFpr). In both cases they require a score function and we will be using a chi square matrix.

**Lastly**, text classification which will analyze the given features of a document and compare them to known patterns. These patterns will be learned from our test data set of 2000 movie reviews. This area will have the largest impact on our results and for that reason we will be comparing 3 different classification techniques. The first one being KNearestNeighbours, this approach is a simple and intuitive machine learning technique. It also requires no assumptions about the data which simplifies the setup of the model. KNearestNeighbours will be compared against the increasingly complex methods, multinomial Naïve Bayes which analyzes word counts to assign probabilities to terms which allows it to predict whether a collection of terms belongs to a given class with a probability p. The most complex method is the multi-layer perception classifier which is a type of neural net. The major advantages of neural networks are that they are incredibly flexible and work well with nonlinear data. On the flip side neural networks depend heavily on the training data and require large amounts of it. Another downside of neural networks is we can not see inside the model and analyze how different terms effect the classification. This reduces our understanding of the results and in turn we may not be able to properly judge the reasoning of our model. For instance the word “helping” could only appear in our sample data positive reviews and the neural net could use that to predict perfectly on our sample data and we wouldn’t have an idea why our model wasn’t performing well on real world data.

Implementation

My implementation for the machine learning strategy talked about above revolves around using Scikit-learn’s built in machine learning pipelines. These pipelines consist of the above 3 steps in order to transform the data and understand its patterns so that we can classify a given document in to either a positive or negative movie review. The first that must occur is the data loading, here we call loadFiles() which returns an 3 by n matrix where each row represents a single review and the 3 columns represent the review contents, whether it is positive or negative, and the third column is the file name it was loaded from. After loading in the data it is split it into 2 separate chunks which will form our validation set (15% of the data) and our training set (85%) of the data. Now we can begin testing our various methods.

The approach to method testing is to first build a scikit Pipeline object containing the necessary components, a preprocessor, a feature selector, and a classifier. With the model constructed we pass that to runTest along with the split data where it will be cross validated on the training data using a 5 fold strategy where it fits itself to 4 of the folds and tests on the remaining fold then rotating the test fold around. After the cross validation is run the mean of the 5 accuracy results is printed and then the verify function is called and the model Is tested on our held back test data.

With this setup it is now easy to define different pipelines tweaking one component at a time and comparing them. For this reason the runTest() function was created where it builds 42 different pipelines and compares; 2 different feature selection methods (select k best and select p values), 7 different cut off values for the feature selection methods, and finally 3 different classification methods (k nearest neighbours, multinomial naïve bayes, and a multilayer perceptron.

For each of the experiments each classification method will be using its default settings. This can be seen below. Some heights the the k nearest neighbour method will use 5 neighbours, a uniform weighting for prediction, it will automatically select the most appropriate algorithm to compute the nearest neighbours, and a leaf size of 30. The multinomial naïve bayes will use an alpha value of 1.0 for its lidstone smoothing. For the Multilayer perceptron neural net it will use 100 hidden layers, the rectified linear unit function for its activation function, and the adam stochastic gradient based optimizer for its weight optimization. Finally the count vectorizer will be set to remove English stop words from the data set.

class sklearn.neighbors.**KNeighborsClassifier**(n\_neighbors=5, weights=’uniform’, algorithm=’auto’, leaf\_size=30, p=2, metric=’minkowski’, metric\_params=None, n\_jobs=None)

class sklearn.naive\_bayes.**MultinomialNB**(alpha=1.0, fit\_prior=True, class\_prior=None)

class sklearn.neural\_network.**MLPClassifier**(hidden\_layer\_sizes=(100, ), activation=’relu’, solver=’adam’, alpha=0.0001, batch\_size=’auto’, learning\_rate=’constant’, learning\_rate\_init=0.001, power\_t=0.5, max\_iter=200, shuffle=True, random\_state=None, tol=0.0001, verbose=False, warm\_start=False, momentum=0.9, nesterovs\_momentum=True, early\_stopping=False, validation\_fraction=0.1, beta\_1=0.9, beta\_2=0.999, epsilon=1e-08, n\_iter\_no\_change=10)

Experiments

Each experiment will consist of a feature selection method and a classification method and then test 7 different feature selection cut off values. Each trial will be cross validated on the training data and then assigned an F Score based on its predictive capability on the held-out data. The training data and held-out data will be balanced meaning that each split will contain an equal number of positive and negative reviews.

Here the K nearest neighbours method performs extremely poorly with an average accuracy slightly better than a coin toss. This can be attributed to not being able to quantify a distance between terms which leads to no way to accurately group terms.

Next, we will analyze the multinomial naïve bayes classifier. Here we are now getting much better results with an average accuracy in the 80%s. The method is much simpler than others where it assigns probabilities to each term. So clearly there is a relationship between certain terms in the reviews that the method can distinguish. As for the feature selection method, there appears to be little if no difference between the 2 options but both methods do improve their accuracy with a greater number of terms available to analyze.

The Neural Net we will be testing is a multiplayer perceptron which is considered more of a black box method. The major downside is we don’t know what the correlation is between the features and the prediction are. On the flip side this method is giving us the strongest results with instances of the model hitting high 80% accuracy predictions. Again here we don’t see a big difference between feature selection methods but we do see one interesting piece of information related to the feature counts. The neural net unlike the naïve bayes doesn’t seem to need anymore than 1000 features as anymore doesn’t improve the accuracy. This is interesting as it may indicate that the neural net has identified patterns between certain terms that are more important than just the raw feature counts.

Model tuning

The results above indicate our best model is the neural net so we will now see if we can improve our results by tuning the parameters of the method.

Model tuning results - TODO

Build Guide

Required:

Python3

Setup:

Pip3 install -r requirements.txt

Chmod +x processor.py

Running:

./processor.py

References

Effectiveness of stop word removal and stemming

<https://link.springer.com/article/10.1007/s10044-005-0256-3>

<https://www.sciencedirect.com/science/article/pii/S0306457313000964#b0140>