Machine Learning For Natural Language Processing And Movie Review Classification

Tyler Green
0977966
University Of Guelph

November 2019

Contents

1 Introduction

What's the problem and its major applications? What major techniques have been used for your project? What data sets are used for your experiments and what are the major results?

The basic application of this program is to classify movie reviews to be a positive review or negative review. The major applications for this project is to help determine which type of model combined with a feature selector on a number of features works best for determining the classification of movie reviews. The other major application is to determine the classification of movie reviews in order to be able to automatically generate statistics based on the number of positive and negative reviews.

The techniques used in this project are three machine learning models, 2 types of feature selection. The three types of machine learning models used were the SVM model, the K Nearest Neighbours model and, the Gaussian Naive Bayes model. These were implemented using scikit-learn. The feature selection methods chosen were chi squared and the mutual information statistics. These were implemented through scikit-learn as well.

The data set used for this project was the movie review data set released at Cornell University in 2004 for sentiment analysis. This data was obtained from the course instructor via courselink. The results from this project showed that SVM was the best model to classify movie reviews. The project also showed that 3000 features works best but 1500 are also acceptable if you are worried about time and space.

2 Techniques

.The first technique for feature selection was chi squared statistic taking the k best values. A major benefit to this test is the test can only be used on non-negative features such as word frequency in documents. The time of the implementation of this algorithm is linear. Another major benefit to using the chi squared statistic is that the algorithm in the scikit-learn package turns sparse data dense. This is a benefit as most of the array hopefully has useful information in it. No big downsides of the chi squared test could be found. The chi squared value for a feature can be calculated with:
$X^2 = (O - E)^2/E$.

The second technique for feature selection used was mutual information

statistic. the mutual information statistic measures the dependency between variables. This was chosen as a feature selection method in an attempt to help reduce the redundancy in the dependency between the chosen features. A major benefit to this statistic like the other one is it turns sparse data dense. No other benefits could be found and no disadvantages could be found. The mutual information test statistic is calculated by:
$I(X;Y) = XYp(x,y)log(p(x,y)/(p(x)p(y)))dxdy$
where where $p(x,y)$ is the joint probability density function of X and Y, and where $p(x)$ and $p(y$ are the marginal density functions.

The first model used was the SVM. The SVM attempts to generate a line to split the data where one side is positive and the other is negative. A major benefit to this method is that when there is a clear split between the data the results can be rather well. A downside to this method is that if there is not a clear split it can have trouble creating an effective line.

The second model used was the K Nearest Neighbours model. This model has an advantage with its learning time as teaching it is as simple as giving it a series of neighbours to compare to. A disadvantage to this method is when sparse data is used there may not be many neighbours. This models big disadvantage is choosing how many neighbours to be the nearest neighbours. Too large of a k value and you can end up with ambiguity on which classification to include. To small of a value and the document could be close to an outlier in the opposite class and get a wrong classification. For this model a value of 5 was chosen.

The third model used Gaussian Naive Bayes model. This model is similar to naive bayes. Where naive bayes only allows for categorical variables, Gaussian naive bayes allows for continuous variables. A benefit to this model is that is requires less training data than other methods making it potentially able to perform better on the smaller data set. The biggest disadvantage is that if a category does not appear in the training data that is in the test data is has a probability of zero.

All three models chosen have the disadvantage of only using tuples of size 1. This is a disadvantage as some words mean more together than separate such as hot dog. Together they are a food, however separate it could be interpreted as a dog that is hot. This can be fixed by updating the way feautures are calculated by adding tuples into the mix and using up to n-tuples. N-tuples were left out in order to help with performance and space.

3 Implementation Details

The implementation was done in python using virtualenv as the environment manager and pip3 as the package manager. The primary data type used was the numpy array. This was used since the scikit-learn models required numpy arrays as input. The program relies on pythons loose data types to be able to save a random model to a set variable and not worry about types. The analysis relies on the data type counter in order to easily count sentences and tokens in a file. This is loosely based on a hashmap where the key is the document name.

There is a limitation on the number of files currently set at 2000 with a dev set at 15 percent. There are assumptions that the movie reviews are strictly all about movie reviews and there is no extra information in them. There is another assumption that the files are in separate folder in the directories they are in. There is another assumption that all negative reviews have neg in the file path and all positive reviews do not contain neg in the file path.

4 Data And ML Models

The data set contained two thousand reviews with one thousand "positive" reviews and one thousand "negative" reviews. The data was split up into a 15 percent dev set consisting of 150 positive reviews and 150 negative reviews. These reviews were selected by taking the first 15 reviews of every hundred. The rest of the 1700 reviews were shuffled and then split into 5 different folds. Each fold is tested on 500 to 3000 features in increments of 500 on each of the models. The best model is chosen as the model for the test and that is taken. After all folds have run the best model is then decided based on the results from the test fold.

Fold 1
Chi Squared

Model	500	1000	1500	2000	2500	3000
SVM	0.7367	0.7400	0.7567	0.7400	0.7433	0.7367
Neighbours	0.5933	0.6400	0.6133	0.6233	0.6467	0.6500
GNB	0.6800	0.6733	0.6833	0.6800	0.6800	0.6800

Mutual Information

Model	500	1000	1500	2000	2500	3000
SVM	0.7400	0.7367	0.7433	0.7400	0.7533	0.7600
Neighbours	0.6233	0.6300	0.6367	0.6633	0.6733	0.6533
GNB	0.7367	0.7133	0.6867	0.6967	0.6733	0.6567

Fold 2
Chi Squared

Model	500	1000	1500	2000	2500	3000
SVM	0.7333	0.7400	0.7466	0.7433	0.7433	0.7533
Neighbours	0.5833	0.6000	0.5766	0.5833	0.6033	0.6033
GNB	0.6533	0.6533	0.6700	0.6867	0.6967	0.6800

Mutual Information

Model	500	1000	1500	2000	2500	3000
SVM	0.7467	0.7500	0.7533	0.7433	0.7467	0.7533
Neighbours	0.5767	0.5933	0.5900	0.5833	0.6100	0.6167
GNB	0.7433	0.6933	0.7067	0.6867	0.7067	0.6900

Fold 3
Chi Squared

Model	500	1000	1500	2000	2500	3000
SVM	0.7200	0.7367	0.7333	0.7333	0.7367	0.7367
Neighbours	0.6133	0.6000	0.6267	0.6267	0.6100	0.6133
GNB	0.6667	0.6833	0.7100	0.7133	0.7333	0.7067

Mutual Information

Model	500	1000	1500	2000	2500	3000
SVM	0.7200	0.7533	0.7500	0.7533	0.7667	0.7733
Neighbours	0.6167	0.5967	0.6133	0.6133	0.6200	0.6367
GNB	0.7400	0.7233	0.7100	0.7067	0.7167	0.7167

Fold 4
Chi Squared

Model	500	1000	1500	2000	2500	3000
SVM	0.7433	0.7433	0.7367	0.7533	0.7500	0.7466
Neighbours	0.6033	0.6100	0.5900	0.6067	0.6033	0.6000
GNB	0.6733	0.6800	0.7033	0.7067	0.7000	0.6900

Mutual Information

Model	500	1000	1500	2000	2500	3000
SVM	0.7300	0.7333	0.7500	0.7500	0.7433	0.7567
Neighbours	0.6033	0.5767	0.6067	0.6267	0.6267	0.6467
GNB	0.7267	0.7167	0.7033	0.7133	0.6900	0.6800

Fold 5
Chi Squared

Model	500	1000	1500	2000	2500	3000
SVM	0.7300	0.7367	0.7500	0.7467	0.7533	0.7500
Neighbours	0.6033	0.6600	0.6333	0.6300	0.6333	0.6567
GNB	0.6633	0.6733	0.6767	0.7000	0.7033	0.6933

Mutual Information

Model	500	1000	1500	2000	2500	3000
SVM	0.7567	0.7533	0.7600	0.7600	0.7533	0.7600
Neighbours	0.5967	0.5800	0.5800	0.6000	0.6067	0.6300
GNB	0.7333	0.6933	0.7033	0.7200	0.7100	0.6967

From looking at the f values calculated from predicting the dev set we can see that SVM is producing a higher F value than GNB and GNB higher than Neighbours. This is shown when the best for each fold is a SVM model and specifically on mutual information.

Fold	SVM
1 (3000 features)	0.7567
2 (1500 features)	0.7323
3 (3000 features)	0.7794
4 (3000 features)	0.7382
5 (1500 features)	0.7647

The SVM gives f values all in the mid seventies. This shows that the model is working rather well for the test data. Overall the best chosen model was SVM model with 3000 features on fold 3. The fact that all the folds produce roughly the same value shows that the SVM is a good model for predicting reviews as when randomized it works on uneven data.

5 Conclusion

The conclusion for this project is for the basic classification problem of movie classification the SVM model is the better model over K nearest neighbours and gaussian naive bayes. The number of features is less important however the mutual information is the feature selection method prefered over chi squared. One possible future improvement is optimizing the numpy array modifications as this was a major slowdown in the code. Another improvement would be to add support for more than 2000 files. Another improvement would be to save the best model that way it can be used in future projects.

6 Build And Installation Guide

In order to build the and run the program there are a few things that need to be done. First download and extract the project

- download and extract the project
- navigate to the project directory
- create a new virtual env using the command: py -m venv env or python3 -m venv env
- Install the required packages outlined in requirements.txt
- type python ml.py ¡-a¿ ¡-f n¿
  where -a will only conduct the analysis
  -f n will run with n folds

References

https://stackoverflow.com/questions/46752650/information-gain-calculation-with-scikit-learn
https://scikit-learn.org/stable/modules/generated/sklearn.feature$_extraction.text.TfidfVectorizer.html$
$https://kavita-ganesan.com/tfidftransformer-tfidfvectorizer-usage-differences/.XeB5r-hKjb1$
$https://levelup.gitconnected.com/the-easy-guide-to-python-command-line-arguments-96b4607baea1$
$https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/26/python-virtual-env/$