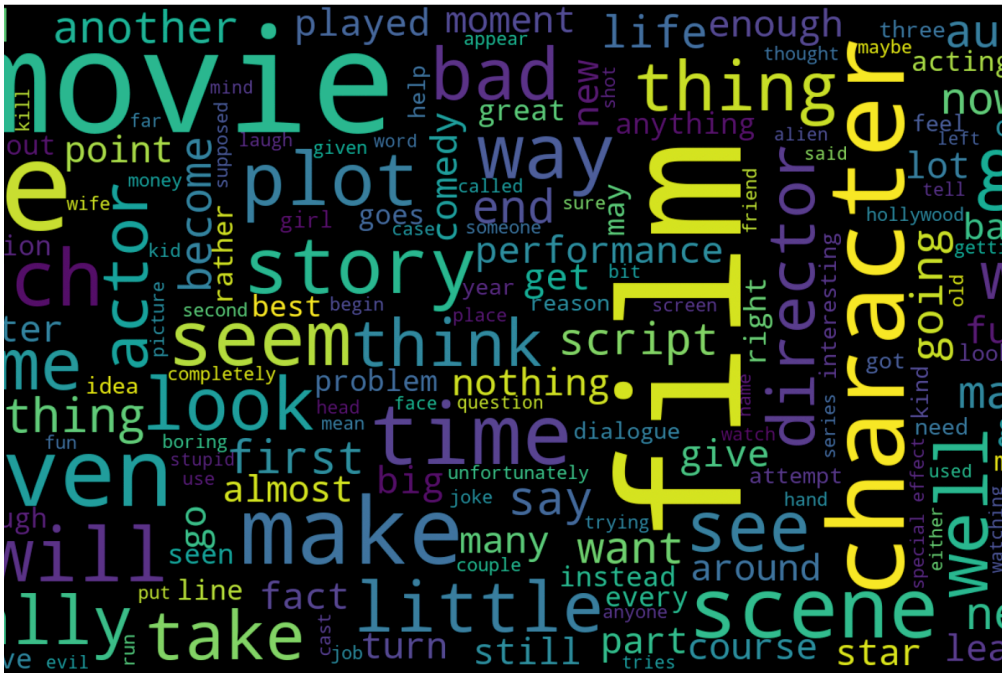
Sentiment Classification

A comparison of techniques



Ian Cross

November 2019

Introduction

Today there are millions of pieces of text uploaded to the internet daily. Take twitter for example, there are over 500 million tweets a day, how are they supposed to define what tweets are very negative and others that are positive. That is where sentiment classification can come into play, defining when a tweet is very negative to the point that it should no longer be put live is a huge task for Twitter.

Sentiment classification is effectively mining a piece of text to find the positive and negative words or phrases to categorize the subjective opinion of the writer. It can be applied to a very wide variety of tasks, from brand reception, to online chat monitoring, or even movie reviews.

This projects focus is around movie reviews, in an attempt to compare multiple combinations of feature selection, and classification techniques. More specifically this implementation focused on the use of one of chi-square statistic or ANOVA f-test in order to complete feature selection. After the features had been selected one of Bernoulli Naive Bayes, k Nearest Neighbors, or Nearest centroid were used as classification techniques to properly identify the parameters of the models.

In order to increase the strength of the models being created, a stratified k-fold algorithm was used. This algorithm is a generic implementation of cross validation. The implementation allows for the specification of the desired number of folds to test with but the default is 5.

A selection of 2000 movie reviews is to be used as our dataset for analysis. This dataset was provided by Cornell University (2004) which consists of 1000 positive reviews and 1000 negative reviews neatly sorted into their own categories. After comparing the 36 different combinations of feature selection methods, feature sizes, and classification techniques mentioned above, it was found that the ideal combination was an ANOVA f-test to identify the top 1500 features, and classified with Bernoulli Naive Bayes technique. This combination repeatedly produced models with an accuracy score above 80% during the validation steps, and above 90% during the training/testing phases.

Techniques

Feature Extraction

In order to begin the reviews must be consolidated into a representation that can be understood by the computer. This initial step is accomplished by extracting the 'features' from the text. A feature, in this case any unigram, is taken out of the reviews and stored in a large spares matrix. The algorithm of choice is an implementation of `tf*Idf`. This algorithm learns the vocabulary, and calculates the inverse document frequency of every word, then represents every word in the matrix by its representative `tf*Idf` score.

For example, three documents are: "The quick brown fox jumped over the lazy dog.", "The dog", "The fox"

The vocabulary is represented by:

{'fox': 2, 'lazy': 4, 'dog': 1, 'quick': 6, 'the': 7, 'over': 5, 'brown': 0, 'jumped': 3}

The relative document frequencies can be shown as follows:

[1.6931, 1.2876, 1.2876, 1.6931, 1.6931, 1.6931, 1.6931, 1]

And then the encoded `tf*Idf` score for the first document is represented by.

[0.3638, 0.2767, 0.2767, 0.3638, 0.3638, 0.3638, 0.3638, 0.4298]

The resulting matrix would be all the encoded sentences sparely stacked together. Filling with 0's for missing words

The major advantage of this approach is that some words like "the" will appear many times and their large counts will not be very meaningful in the encoded vectors, so the words that occur in so many documents and in high frequencies will not be valued as highly as more important words.

Feature Selection

The first feature selection method is chi-square statistic. This method compute chi-squared statistics between each non-negative feature and class. This score calculated from a training set, which must contain only non-negative features like document frequencies. The chi-square test measures dependence between variables, so using this function "weeds out" the features that are the most likely to be independent of class and therefore irrelevant for classification.

The second feature selection method is an ANOVA f-test. This algorithm computes an ANOVA f-value for the provided samples. This is computed between X (the `tf*idf` matrix) and the target values.

Both feature selection methods are sent in as parameters to the `SelectKBest` method to select the top k features from the calculated scores.

Selecting k Best Features

Each feature selection method (vectorizer) is run through a `SelectKBest()` algorithm. This is a fairly simplistic function, when supplied with a feature selection method and a feature size. It will trim off n features with the highest scores resulting from the supplied method.

```
top_feat = SelectKBest(feat_select_meth, k=num_feat)
X = top_feat.fit_transform(X, y)
X_validate = top_feat.transform(X_validate)
```

Once the initial features have been selected, the training data must be fit to exclude the non selected features. The validation set is also fit along side so both have matching features.

Classification Techniques

K Nearest Neighbours

The Nearest neighbor classifier's goal is to find a predefined number (k) of training samples closest in distance to the new point, and predict the label from those neighbors. The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice. Neighbors-based methods are known as non-generalizing machine learning methods, since they simply "remember" all of its training data.

Nearest centroid

A Nearest Centroid classifier is a classification model that finds the means (centroid) of the classes in the training data. It then assigns the observations the label of the class of training samples whose mean (centroid) is closest to the observation. When used for text classification with tf-idf vectors, this classifier is also known as the Rocchio classifier.

Bernoulli Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. We can use Beyes Theorem to estimate P(y) and P(xi|y) the former is then the relative frequency of class y in the training set. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods.

Cross Validation

An implementation of cross validation is achieved by using the `StratifiedKFold(n_folds)` algorithm. This algorithm returns a cross validation object. This iterable object can be supplied the training set and target set. The function preserves an even percentage of each class for each fold.

Description of the techniques used, including the basic ideas (maybe with examples), all the steps and formulas, and major advantages and disadvantages.

Implementation

While many languages are capable of producing robust and dynamic machine learning solutions. Python was decided as a good choice to follow for this implementation.

> Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Pythons simplicity allows developers to write reliable system. Developers get to put all their effort into solving an ML problem instead of focusing [heavily] on the technical nuances of the language.
>
> - Angela Beklemysheva

When paired with powerful libraries such as scikit-learn (Scikit), NumPy, and Matplotlib (Matplot) the focus can be shifted on to proper implementation of a variety of methods and algorithms.

Representing the dataset was the first major choice that was made during the project. By reading through the documentation of Scikit, their base implementation of a Bunch was chosen. A Bunch is a predefined container for datasets, it is a dictionary like object that exposes a few key attributes. These attributes can be then used to access the needed parts of the dataset easily and consistently.

An assumption was made during the data loading process, the data is to be formatted into individual files stored in positive and negative directories. These directories lead to the dataset retrieving it's labels for each of the documents. This is limiting as a corpus that is aggregated into a single file will need to be split by document and pre sorted into labeled folders.

Ideally a set would be used that could contain much more of each data type in order to train a better model. Allowing for more example of what is considered positive and negative directly allows our created models to perform more training, effectively creating strong models.

Once the data has been loaded and processed into a bunch, the features are extracted. Scikit provides a clean way of feature extraction, it includes a variety of 'vectorizers'. These vectorizers provide a way to transform the dataset represented by bunches into a NumPy spares matrix of unigrams. The specific vectorizer also applies a simple TfIdf calculation to each of the unigrams. Another vectorizer was considered, a counter vectorizer, that provides uses a one hot method to produce a spare matrix of document frequencies. This however, was discarded as it provided must worse accuracy across all models.

These spares matrices are used moving throughout the rest of the program and are sent through as training and testing data to the different feature selection methods and classification techniques.

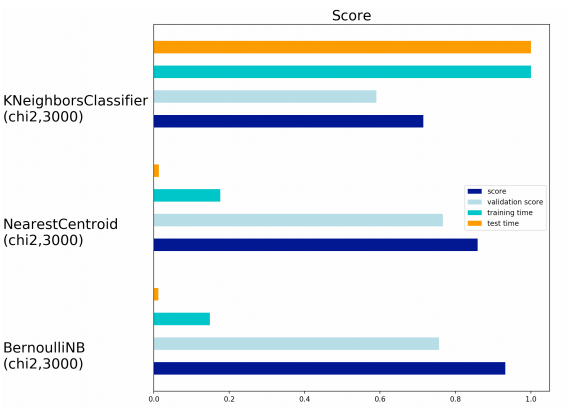
Results

Dataset

The dataset consists of 2000 movie reviews, 1000 positive and 1000 negative. Each movie review consists of a paragraph that is subjectively opinionated in nature towards positive or negative.

Comparison

There are multiple directions to compare over, first we will look at the effectiveness of each feature selection method. To keep the interactions simple, we will look at the largest feature set size individually and compare all three classification techniques together.



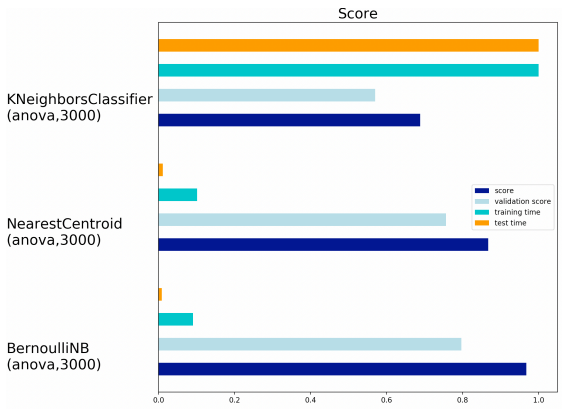


Figure 1 & 2: Scores, and timing tests for 3 classification techniques across 2 feature selection methods

	Training Accuracy	Validation Accuracy
KNeighbors with Anova and 3000 samples	0.688	0.570
Nearest Centroid with Anova and 3000 samples	0.868	0.757
Bernoulli Naive Bayes with Anova and 3000 samples	0.968	0.797
KNeighbors with chi-square and 3000 samples	0.715	0.590
Nearest Centroid with chi-square and 3000 samples	0.859	0.767
Bernoulli Naive Bayes with chi-square and 3000 samples	0.932	0.757

Table 1: Highest accuracy for models created with 3 classification techniques across 2 feature selection methods

From our initial results, we can see that between the two methods there isn't a significant difference. Some of the classifiers showed some increase but not to a drastic level. Lets compare the different feature set sizes over Naive Bayes, which shows the strongest accuracy as of now.

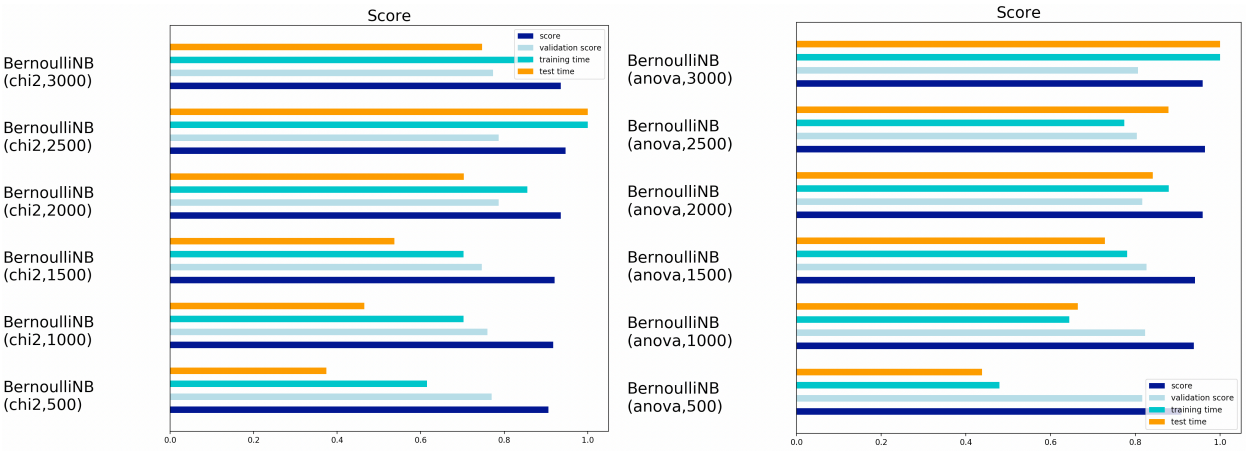


Figure 3 & 4 Scores, and timing tests; Naive Bayes classification across 2 feature selection methods and 6 feature sizes

Bernoulli Naive Bayes	Training Accuracy	Validation Accuracy
chi-square and 500 samples	0.906	0.770
chi-square and 1000 samples	0.918	0.760
chi-square and 1500 samples	0.921	0.747
chi-square and 2000 samples	0.935	0.787
chi-square and 2500 samples	0.947	0.787
chi-square and 3000 samples	0.935	0.773

Bernoulli Naive Bayes	Training Accuracy	Validation Accuracy
Anova and 500 samples	0.909	0.817
Anova and 1000 samples	0.938	0.823
Anova and 1500 samples	0.941	0.827
Anova and 2000 samples	0.959	0.817
Anova and 2500 samples	0.965	0.803
Anova and 3000 samples	0.959	0.807

Table 2 Highest accuracy for Naive Bayes classification across 2 feature selection methods and 6 feature sizes

From this we can gain two insights, one we see that the two highest scores for the feature set sizes are on the 2000, and 1500 level sets. 2000 for a chi-square and 1500 for an anova test. We can also show that through this the ANOVA f-test method of feature selection is outperforming the chi-square stat. These results can be backed up as well when looking at a chart comparing all 36 combinations. Keeping in mind this chart looks chaotic but it can portray what we are seeing in the individual interactions.

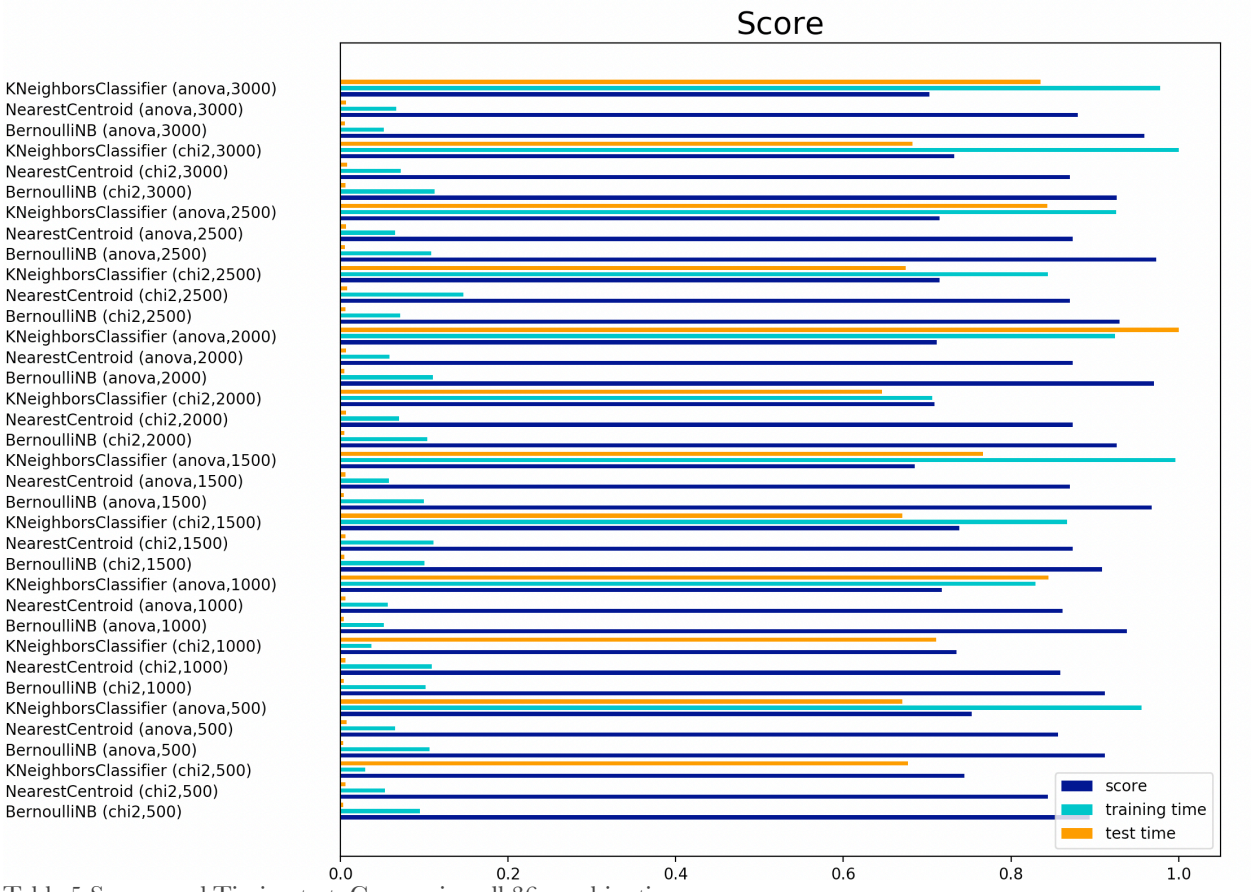


Table 5 Scores and Timing test. Comparing all 36 combinations

Conclusion

One of the most interesting things that was discovered during this testing was that the higher feature sizes didn't always produce the most accurate models. Along with that it was somewhere in the middle that showed the highest results.

It was also very interesting to see the difference between the 3 classification methods, especially in their training time. K Nearest Neighbors constantly showed the highest training and testing times but that did not show anything in the results as it was constantly lower than the others which had a nearly negligible training and testing time.

Finally, the most "Naive" classification method seems to constantly out perform the other two. This was not the initial thinking when choosing the three methods. With the best model overall being - an ANOVA f-test to identify the top 1500 features, and classified with Bernoulli Naive Bayes technique.

This project consisted of testing 2 feature selection methods across 6 feature sizes with 3 different classification techniques. An easiest transition to improve this would to be include more of either of the 3 categories, to add more combinations. This would theoretically allow the program to find a better solution than was found. Another easy improvement would to be increase the size of the training dataset. This would allow for the models to find more connections between words and the positive and negative targets. With stronger connections, during validation a strong accuracy would be expected.

More analysis needs to be completed to compare and contrast all 36 combinations of results and all their interactions. This short paper however wasn't the correct avenue. This should be expanded upon to truly derive the aspects of a sentiment classification system that can provide the strongest models.

Build and Installation

Code base

This project was written in Python 3.7 using a virtual environment and package manager called Pipenv. As long as the machine has a working version of those, this project should execute properly. As Python doesn't need to be compiled this section will just explain the different execution methods for the project.

Execution

Pipenv can be installed using pip3 - a package manager that is installed alongside python. Once pipenv is installed, navigate into the root directory and use pipenv install & pipenv shell to activate the virtual environment. Once in the environment the program can be executed as a normal python program.

There are two main parts to the project. The first, 'analysis.py', which gives insight on the dataset, can be run separately with " python3 analysis.py ". The program will look for a 'dataset' folder by default or can be directed to another directory using the " -ds " or " --dataset " flags. The included directory should contain two sub directories 'pos' and 'neg' containing example positive and negative reviews respectively. Inclusion of the " -v " or " --verbose " flag will allow individual review statistics along with final averages.

The second part of the program can be run similarly with a variety of flags to choose the desired comparison. Those flags can be seen in the chart below, for proper operation at least one feature selection method and at least one classification technique needs to be chosen. Normal execution with one feature selection method, one classification technique and all feature sizes with no desired output would look like this: " python3 compare.py -f2 -cb ". The program will compare the chi-square feature selection method and a Naive Bayes classification technique over 6 feature sizes (500,1000,1500,2000,2500, and 3000) and provide a short output showing the best model (after 5 fold cross validation) for each feature size.

Command line Argument	Default	Actions
"--feat-anova" or "-fa"	Not Active	Creates a selection of k features using the calculated f-value using an ANOVA f-test between sample and label
"--feat-chi2" or "-f2"	Not Active	Creates a selection of k features using the calculated chi-square statistic between non negative features
"--clf-bayes" or "-cb"	Not Active	Uses the Bernoulli naive Bayes classification technique to identify the best model
"--clf-neighbors" or "-cb"	Not Active	Uses the k nearest neighbors classification technique to identify the best model
"--clf-centroid" or "-cb"	Not Active	Uses the nearest centroid classification technique to identify the best model
"—max-features x" or "-mf x"	Uses all feature counts	Defines the desired feature size x, will only run using that size. x must be a integer

Command line Argument	Default	Actions
“--verbose“ or “-v”	Silent	Prints out helpful logs, and a classification report of each model created during cross-validation
“--plot-results“ or “-pr“	Not Active	Creates a large chart comparing the best models from each combination
“--file-results" or “-fr”	Not Active	Prints out the same data used to make the chart, can be used to perform deeper and more specific analysis
“--num-splits" or “-ns”	5	Defined the number of folds to use in the cross validation step

Table 1: Command line arguments for the “compare” program

References

Welcome to Python.org. Python.org. https://www.python.org/. Published May 29, 2019.

Kennethreitz.org. (2018). Pipenv: Python Dev Workflow for Humans — pipenv 2018.11.27.dev0 documentation. [online] Available at: https://pipenv.kennethreitz.org/en/latest/

Scikit-learn.org. (2017). API Reference — scikit-learn 0.20.3 documentation. [online] Available at: https://scikit-learn.org/stable/modules/classes.html.

Cornell.edu. (2012). Data. [online] Available at: http://www.cs.cornell.edu/people/pabo/movie-review-data/

Beklemysheva, A. (2019). *Why Use Python for AI and Machine Learning?*. [online] Steelkiwi.com. Available at: https://steelkiwi.com/blog/python-for-ai-and-machine-learning/.