Machine Learning Engineer Nanodegree

Capstone Project

Chester Fung
January 21st, 2016

I. Definition

Domain Background

The project of this capstone is on the entertainment film industry. The global film industry shows healthy projections for the coming years, as the global box office revenue is forecast to increase from about 38 billion U.S. dollars in 2016 to nearly 50 billion U.S. dollar in 2020. The U.S. is the third largest film market in the world in terms of tickets sold per year, only behind China and India. More than 1.2 billion movie tickets were sold in the U.S. in 2015. Many websites offer portals for users to give them feedback or reviews of the movies they watch. These reviews include both positive and negative. Being a big movie fan, I visit these review sites often to look for which movies I should watch.

Several papers have been published before on this topic, including
Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). "Learning Word Vectors for Sentiment Analysis." *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*. (link)

The paper and project is based on a Kaggle Competition.

Problem Statement

This capstone project is to classify the user reviews from IMDB.

Specifically, it is to classify the sentiment of sentences from the IMDB dataset. Label sentiment of the review; 1 for positive reviews and 0 for negative reviews

We'll then use feature extraction module from scikit-learn to create bag-of-words features.

Metrics

Evaluation Metrics

Evaluation metric that will be used in this capstone project is the area under the ROC curve.

In a ROC curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points of a parameter.. The formula for the ROC curve score is:

$$\mathrm{AUC} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} 1_{p_i > p_j} .$$

There are different evaluation metrics to evaluate the performance of a classifier, including F score and AUC. Though they are similar, but they're different. F score is for a fixed pair of precision and recall. In other words, it represents a particular point of the ROC curve.

One reason for choosing AUC over accuracy is that AUC will strong discourage for choosing models that are representative but not discriminative. AUC is especially good when distributions are skewed toward one class, and don't want to overfit a single class.

II. Analysis

Data Exploration

Datasets and Inputs

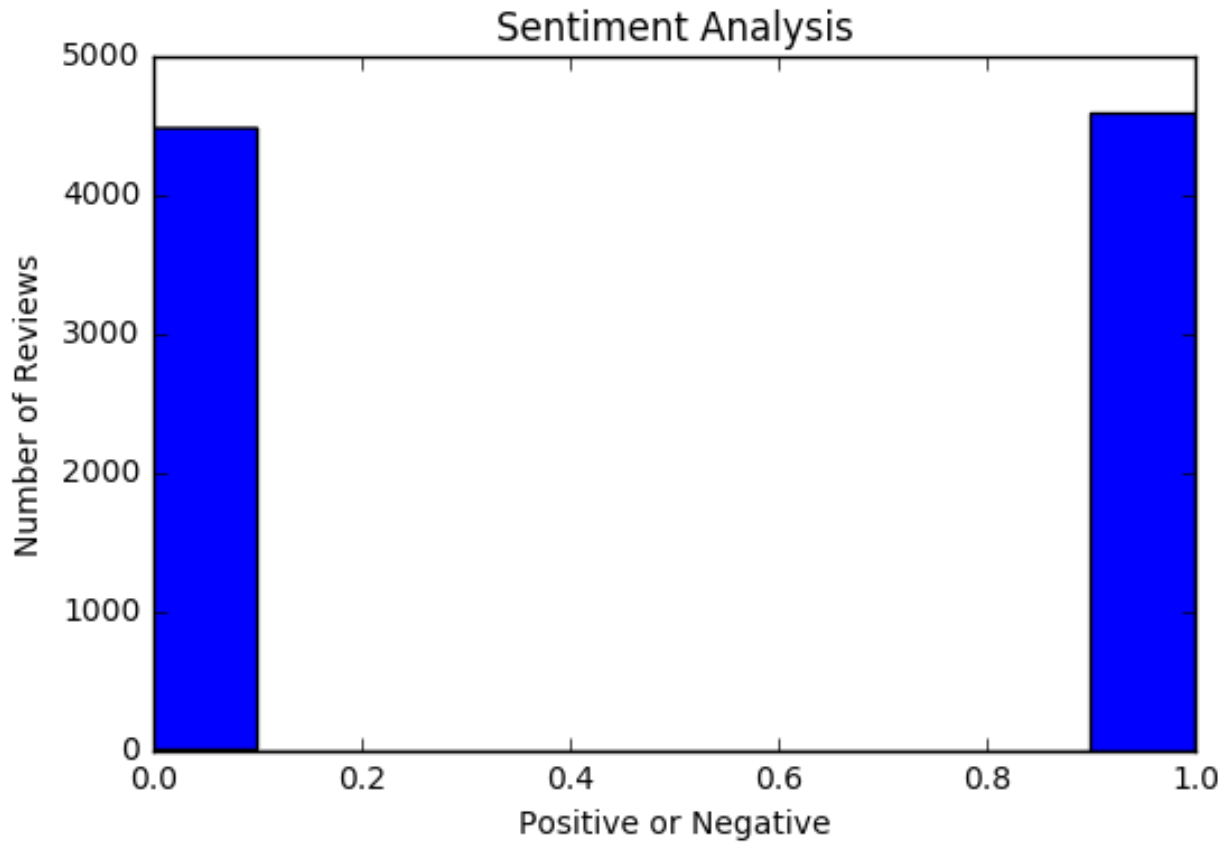
Data set is from IMDB

Size of the data set has 50,000 movie reviews that are labeled. And another 50,000 that are unlabeled.

The labels in the dataset include id, sentiment and review. ID is a numeric value assigned to the review. Sentiment (score of 0 to 10) is an integer representing the sentiment score. Review is a string written by users.

25000 reviews from the labeled test data set will be used for training. Another 25000 will be used for testing.

Exploratory Visualization



"Figure 1 – Number of Positive reviews vs Negative

The figure above plots the number of positive reviews and negative reviews. This plot gives us an idea of the distribution of positive and negative reviews. The bars show that the numbers of positive and negative reviews are close. Each is around ~4500.

Algorithms and Techniques

Sentiment of the review should be labeled as 1 for positive and 0 for negative. IMDB rating < 5 results in a sentiment score of 0, and rating >=7 have a sentiment score of 1

We'll use an approach called "Bag of Words". The Bag of Words model learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. Scikit-learn will then be used to create features from "Bag of Words" Support Vector Machines and Random Forest will then be used to tackle this classification problem. We'll then try different options in CountVectorizer to tune and optimize the model.

The following algorithms will be used and results will then be compared:

Logistic regression - is a regression model where the dependent variable (DV) is categorical, where it can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick

advantages: Good if problem (target variable) is linearly separable. Robust to noise. No distribution requirement

Reason for choosing: trains must faster than a random forest for sparse high-dimensional data such as text

Random Forest Boosting - A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

Advantage (Random Forest): fits well with uneven data sets with missing variables. Lower classification error rate compared to decision tree. Faster training time compared to SVM

Naive Bayes classifier - It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Advantages: fast to train, not sensitive to irrelevant features

weaknesses: assumes independence of features

Reason for choosing: Naive Bayes works well for small training set sizes and it's fast, and our dataset is small

SVM (Support Vector Machine) - discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples

Disadvantage: Inefficient to train. Therefore, it's not good for problems with many training points

Advantages: high accuracy, works great for linear problems

Reason for choosing: high accuracy

Benchmark

Benchmark Model

This is a Kaggle competition. Winner of the competition achieved an ROC curve score of **0.99259**. The goal of this capstone project is to obtain **0.8 or above**

III. Methodology

Data Preprocessing

We'll first preprocess /clean the data by doing the following steps:

- **Use BeatifulSoup to parse the data**

- **Remove numbers and punctuations, convert all letters to lower case and separate the phrases into individual words.**

We'll also remove "stop" words. These are words that do not have much meaning, i.e. 'and', 'is', 'a'.

Number of features extracted from input text:

Feature_extraction.text was used from CountVectorizer to count the number of train data features. The length of train_data_features in the jupyter notebook was 9060

Implementation

After removing the 'stop' words, next step is to convert the review into words.

Create an empty list, clean_train_reviews, and add all the "cleaned" train reviews.

We'll split the data into train and test data. Since we've already created a "clean_train_reviews" list and appended all the "clean" reviews to the list, we'll perform the same step against the test dataset, "test_train_reviews".

Create an empty list, test_train_reviews, add all the "cleaned" reviews to it by using convert_review_to_words function.

Now, apply the "Bag-of-words" model. The bag-of-words model is a simplifying prepresentation used in natural language processing and information retrieval. In this model, a text is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. For example, we have the following two documents,

1. John likes to watch movies. Mary likes movies too.

2. John also like to watch football games.

Based on these two text documents, a list is constructed as follows:

[
"John",
"likes",
"to",

"watch",
"movies",
"also",
"football",
"games",
"Mary",
"too"
]

We can then use the Bag-of-words model as a tool of feature generation. After transforming the text into a "bag of words", we can calculate the term frequency.

In our case, the following code demonstrates to application of using the "bag-of-words" model.

After the train_data_features has been converted to an array, let's look at 1 entry,

```
train_data_features[:1]
```

```
array([[0, 0, 0, ..., 0, 0, 0]])
```

As expected, it is an array containing 0s and 1s.

Now, we'll use the features and use an algorithm to train and fit the training set. Let's start with Random Forest first

After the Random Forest model is trained, use it to predict on the test dataset

Refer to the "Results" section for score generated by Random Forest.

Next, let's try on Logistic Regression

Pls refer to the "Results" section for score generated by Logistic Regression

Next, we'll try using Naïve Bayes

After training with the train data using Naïve Bayes, let's predict with the test data. Pls refer to the score generated by Naïve Bayes in the "Results" section

Finally, we'll use Support Vector Machine (SVM) model.

Once the model finishes using the train data set to fit, we'll predict the result using the test data set. Pls refer to the score generated by SVM in the "Results" section

Refinement

Let's play with different parameters in RandomForest and see if it improves the performance.

Random_state :

Set random_state to 42. After changing this parameter, Random forest result has increased to 0.836 from 0.833.

For logistic regression, GrdSearchCV is used to find the optimal values for different parameters. There are a number of parameteirs for Logistic Regression, including C. C is the inverse of regularization strength. The lower the value of C, the strong the regularization is.

With the GridSearchCV implemented, logistic regression now obtains a result of **0.8628**, which is higher than the version without GridSearchCV imeplemented

IV. Results

The final score is judged on area under the ROC curve. A Receiver Operating Characteristic curve (ROC curve), is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The results for different models are the following:

Random Forest: ~~0.83384~~ 0.836
Logistic Regression: ~~0.83384~~ **0.846**
Naïve Bayes: ~~0.83384~~ 0.691
SVM: ~~0.83384~~ 0.803

Previously I had the same result (0.83384) was due to a careless mistake. When I was creating the dataframe to submit, I forgot to change the variable name for "sentiment". Therefore, all the results for different algorithms were still identical. After fixing this mistake, the numbers for different results are now correct, with Logistic Regression achieving the highest score (0.846), and Naïve Bayes got the lowest score(0.691).

The benchmark model mentioned that the winner of the kaggle obtained 0.99259 and my goal was to surpass 0.8. The logistic regression with gridsearchCV implemented was 0.86280. And this result has met the objective goal.

Model Evaluation and Validation

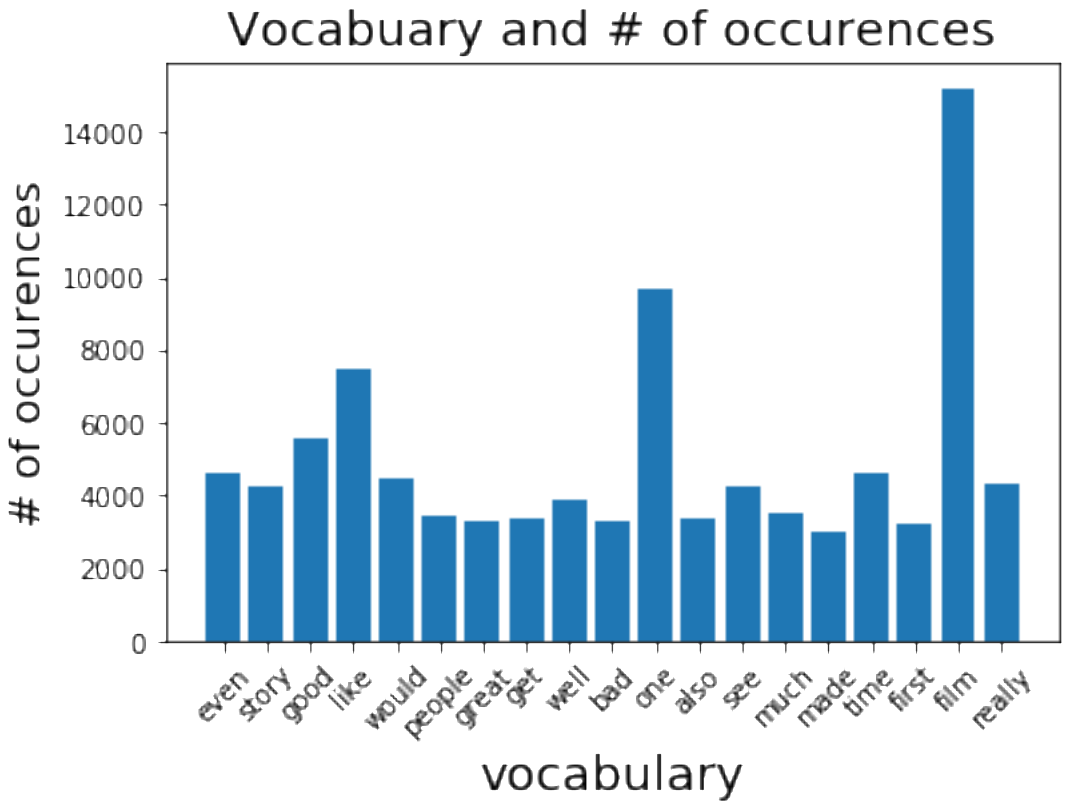
Justification

Based on the result, the scores for using different models are still the same, that leads me to believe that there could be something wrong with my code, could be related to training set or test data set parsing.

V. Conclusion

Free-Form Visualization

The chart below plots the vocabulary and the number of occurrences of those vocabs. This plot shows a distribution of the vocabs.



"Figure 2 – Vocabulary and its count

The figure above lists the words used in review and their count. Certain words such as ‘great’, ‘good’ are more likely to appear in positive reviews. And other words such as ‘bad’ are more likely to appear in ‘negative’ reviews. There are also neutral words such as ‘film’, ‘really’, ‘story’ that can appear in either positive or negative review. More context used with those words need to be examined. A figure displaying the count distribution of words gives readers an idea of how certain words might be distributed and how it might affect positive and negative reviews.

Reflection

One of the difficult/frustrating part of the project is training and predicting results with Random Forest is significantly longer the other algorithms. This could be due to RF using more memory, leading to occasional freeze of my macbook pro.

One of the most interesting aspect of the project was to learn sentiment analysis and different approaches to solving this problem. Typically text classification can be performed using supervised learning, or recursive neural networks. I've implemented the supervised version in this project. However, there are limitations to supervised learning approach. In many languages, the definition of a word can be affected by other words preceding or following it. Therefore, we cannot fully understand the sentiment of a word without its context. This is where recursive Neural Networks can be a powerful way to solve this issue as it goes thru each word, store the meaning and start again. I have not implemented the neural network version in this project but it certainly looks to be an interesting approach.

Recap:

The first step of the project is to define a problem statement. Afterward, we need to determine the evaluation metric to evaluate the effectiveness of the model. In our case, we chose to use AUC (Area Under Curve) to evaluate the performance of models used to predict results. Once we have the evaluation metric determined, we'll start working on the data. We'll need to clean the data by removing numbers and punctuations, convert all letters to lower case and separate the phrases into individual words. We will also remove "stop" words, words which have no impact towards the sentiment of the review.

Once the preprocessing and cleaning step is completed, we can move to the next step of choosing different algorithm models and obtain results. Four different algorithms were used, including Logistic Regression, Random Forest, Naïve Bayes and SVM, with Logistic Regression obtaining the highest AUC score. Refinement by changing different parameters was also performed.

Improvement

One method that can be tried to improve the score is tf-idf, term frequency –inverse document frequency.

Tf-idf, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining.

Another method to try is to use Neural Networks.