Classification of Movie reviews and Recommendations

CS-579 Online Social Network Analysis

Fall 2015

Final Project

Submitted By,

Praveen Balasubramanian – A20327251

Daniela Martinez - A202795721

I. Introduction:

The problem of sentiment classification has received a lot of attention of late. It finds applications in the fields of business intelligence, recommender systems (so as to be able to understand the sentiment in a user's feedback), in online surveys that have natural language sentences as responses, message filtering, assessing movie review polarity, among others. We wish to investigate ways in which these recent advances can be applied to the problem of sentiment classification for paragraphs of text (i.e., with multiple sentences, as is the case typically in movie reviews). The goal of the first phase of our project is to perform opinion mining or sentiment analysis on a large corpus of movie reviews by learning more about the features present in the reviews. We look forward to develop a classification model that automatically predicts the sentiment of a given movie review to be either positive or negative. We also would like to experiment deeper with respect to various machine learning techniques that would best perform given the circumstances and the objective of our project.

Recommendation systems are used in many of today's applications in order to enhance user experience or yield profit for commercial companies. Interaction with this type of system is part of everyone's daily life whether they are shopping on eBay, performing a search on Google, or simply watching a video or movie on their phone. Netflix, an on-demand streaming service, uses a recommendation system to suggest movies and TV shows to users based on movies or shows they've recently watched or rated. A user's account profile is customized based on such information so that only titles that are likely to be of interest to a user are displayed at login. Due to the popularity of Netflix in recent years, we decided to focus on movie recommendation systems: How do they determine suggestions? How can they be improved? The purpose of this second phase of our project is to understand the logic behind their implementation as well as analyze their level of accuracy by building our own recommender system using the user-user collaborative filtering system.

II. Data

We wrote a script to collect about 44,000 movie reviews from the metacritic website (www.metacritic.com) using their API. We stored this data into a text file with tab delimitation. This dataset consists of two fields especially, the first one is the "Reviews" field that contains movie reviews given by critics in a single sentence and the second one is the "Sentiment" field that indicates the sentiment score given by the critic on a scale of 100.

We wrote a separate script to fetch relevant data from the metacritic website for the sake of building a recommender system. This dataset consists of 25 movie critics along with the set of movies that each critic has watched and rated with scores on a scale of 100. Basically this dataset is a dictionary of dictionary values. We used pickle library file in python to store this data into a text file.

III. Methods

For the purpose of classification, we used the "Sentiment" field present in the dataset to categorize the data into positive and negative sentiment. We chose a cut-off score of 65 and thus we manually split all the reviews that had a sentiment score greater than 65 to be positive and all other reviews that had a score less than 65 to be negative. We used about 36,000 movie reviews as our training dataset that included 18,000 positive and negative reviews and we tested our classification model on 7,870 reviews which was our test data set.

We stored each of these movie reviews into separate text files by writing a script which basically reads each row of a python dataframe and stores them into an individual text file under the corresponding folders based on the sentiment of the review for the purpose of classification. Thus we created separate training and testing folders and then populated them with ample data.

While experimenting with the naïve bayes classifier using the chi-square feature selection strategy, we used about 20,000 movie reviews as our training set to train our model where half of it was positive and the other half was negative data. We tested our model on the testing set that contained 23,870 movie reviews in total. In this case we stored both the training and testing data in the form of a csv file. We also stored the list of positive and negative features into separate text files for later usage by our model during the classification process on testing data.

IV. Experiments

We attempted to build three different types of classifiers as listed below,

- Logistic Regression
- Multinomial Naïve Bayes
- Naïve Bayes model – using Chi-square test statistic measure for feature selection.

For the first two classifiers we used the following experiments,

1. **Tokenization**
   We created three different tokenization functions. One that tokenizes reviews by considering punctuations as separate tokens, another that looks for the presence of "not" in the given review and whenever this term occurs it appends the word "not" along with the feature that appears next to the original word. By doing so we can normalize the impact of negatively associated features and thus give a chance for our classifier to perform without any external bias. The third way of tokenization is quite general to text classification problems wherein we remove all punctuations that occur anywhere within the review, we are only concerned about the feature sets here. We try to measure the performance of our classifier by estimating its accuracy on each of the three tokenization functions. We observed very minimal fluctuations in both the classifier's accuracy.

2. **K-fold cross validation:**

We used k-fold cross validation method to evaluate the performance of our classification model. Here, the data set is divided into k subsets, and the holdout method is repeated k times. We used different values for k and thus measured our model's accuracy. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.

3. **Comparison of model's performance on varying the n_folds, min_df and max_df parameters:**

We also estimated our model's performance by changing the values of min_df, max_df and n_fold parameters. We observed that by altering the n_fold values and then calling the cross validation method with this parameter, our model's accuracy seemed to increase initially and then flattens out for all n_fold values greater than 10. We learned that this is because of the fact that as the number of folds increases for a fixed amount of data, number of data within each fold decreases. Hence, less data is available for testing. Also, more data is available for training purposes. Therefore it trains the model more and tests the model with less data per fold. Hence, the increase in accuracy.

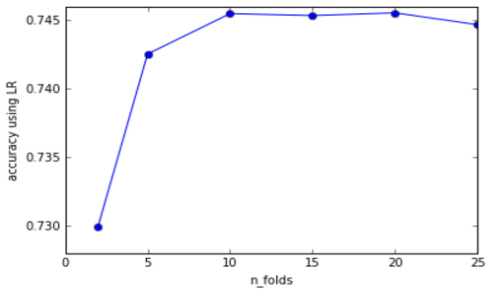


Figure-1

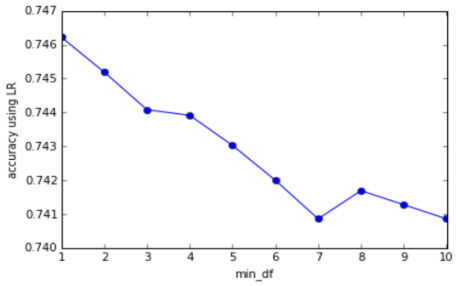


Figure-2

By altering the min_df parameter, we inferred that by setting a too low value the model ignored very rare terms which didn't count towards sentiment calculation. A bit higher value seemed to ignore sufficient rare terms which might have improved the accuracy. Thus we learned that if the value was high, the model ignored necessary terms as well and hence we observed a decline in accuracy in the right hand side of the graph.

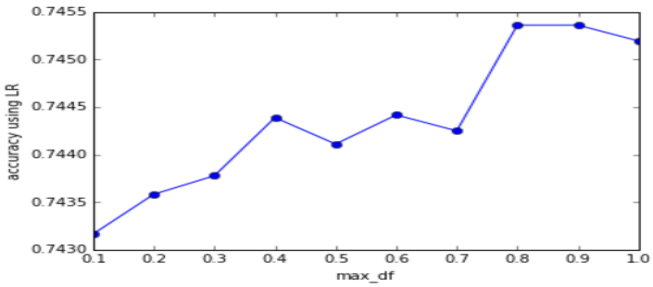


Figure-3

4. **Error analysis on misclassified data:**

We wrote a function to list out the top 10 misclassified documents and we took a brief look at the predicted probability of those documents. We noticed that for the top 25 sorted out misclassified errors, the variance between the actual probability and the predicted probability was nearly about 90%. Which is nothing but our classifier had predicted these to be either positive or negative with about 90% or above probability wherein the actual true label was the opposite to the predicted output. We then analyzed the reason for such misclassification and figured out that our classifier found difficulty to understand the reviews that contained dual polarity based words. For example let's consider a sample review to be "this should have been a good movie, but it wasn't". Words like "but", "not", etc. act to negate the sentiment of words. Our classifier also went clueless when it faced reviews that had multiple word dependencies. We may state another example here such that, "this movie is not remarkable, touching, or superb in any way" just because the review contains typically positive words like remarkable/touching/superb", in these cases our classifier gives it a high probability meaning it to be positive wherein the actual nature of the review turns out to be negative.

Chi-Square Feature selection Strategy:

In this experiment, we first read through each review present in the training dataset and separated positive and negative features along with their term and document frequency and also the total vocabulary of unique words. Now, we are aware that chi-square test statistic is a measure widely used in statistics to measure the independence of two events. In feature selection we use it to test whether the occurrence of a specific term and the occurrence of a specific class are independent. Below is the formal representation of the chi-square test statistic formula,

$$X^2(\mathbb{D}, t, c) = \frac{(N_{11} + N_{10} + N_{01} + N_{00}) \times (N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01}) \times (N_{11} + N_{10}) \times (N_{10} + N_{00}) \times (N_{01} + N_{00})}$$

Figure-4

Below are the stats to be estimated in order to apply them in the formula,

- n11 - The total number of reviews where the word occurs and provided the review was labelled positive.
- n10 - The total number of reviews where the word occurs and the review was labelled negative.
- n01 - The total number of reviews where the word does not appear and the review was labelled positive.
- n00 - The total number of reviews where the word does not appear and the review was labelled to be negative.

For each word present in the vocabulary we estimated this measure for that particular feature and then stored it in a separate dictionary along with its estimated chi-squared value. Now while performing classification on testing data, we only pick out the top 'n' features from this dictionary and use it to estimate prediction. During prediction our classifier makes use of the naïve bayes formula to estimate the prior and posterior probabilities of those features that occur in each review and hence based on the likelihood it makes a decision on the class label. This approach of feature selection has proved to be highly efficient given the size of the training dataset. We measured the accuracy of our classifier by choosing different set of 'n' features and then tried to measure the performance of our classifier on testing data based on these features.

Recommendation system:

While building the recommender system we tried to experiment two different similarity measures namely the Euclidean distance measure that estimates the similarity between two users and then followed by the Pearson correlation co-efficient. The problem with euclidean distance is that it measures the dissimilarity too, in our case, the people who like the same movies are less important than the ones that prefer different movies or that they have different tastes. To be more precise, instead of just relying on the most similar person, a prediction is normally based on the weighted average of the recommendations of several people. The weight given to a person's ratings is determined by the correlation between that person and the person for whom to make a prediction.

The Pearson correlation computes the statistical correlation (Pearson's r) between two user's common ratings to determine their similarity. This measures how well two critics are linearly related in our case. A formal representation of the Pearson-co-efficient is given below, where the variables used in the formula refer to the ratings given by two users A and B.

Figure-5

$$\text{Pcc}\ (U_a, U_b) = \frac{\sum_{i=1}^{n} (R(a)_i - \overline{R(a)})(R(b)_i - \overline{R(b)})}{\sqrt{\sum_{i=1}^{n} (R(a)_i - \overline{R(a)})^2 \sum_{i=1}^{n} (R(b)_i - \overline{R(b)})^2}}$$

V. Related work

We built three different types of classifier's and then estimated their performance on the testing data.

1. Logistic Regression

It means to obtain a best-fit logistic function in terms of machine learning. We try to come up with a function that can predict the future inputs based on the experience it has gained through the past inputs and their outputs (training set). Logistic Regression is - coming up with a probability function that can give us 'the chance, for an input to belong to any one of the various classes' we have (classification). In our case we look forward to come up with a probability function that basically takes in an input X (the feature vectors of a review) and return 'the probability of this review to be either positive or negative'. This probability function or also known as the sigmoid function can be represented as below,

Figure-6

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

Now in our case we used the Scikit-learn logistic regression package to estimate this probability function of classifying a review. Below is the accuracy measurements of this model on various data.

Table-1

Data	Accuracy
Train – within training	93.08%
Test – within training	74%
Overall Training data	74.39%
Overall Testing data	72.27%

2. **Multinomial Naïve Bayes**

This technique is used when the multiple occurrences of the words matter a lot in the classification problem. It is a specialized version of Naive Bayes that is designed more for text documents. This model explicitly models the word counts and adjusts the underlying calculations to deal with in. Basically our goal here is to determine whether a particular movie review is positive or negative. But in general reviews are just some bunch of words, so the first step in order to train a multinomial Naive Bayes model to classify reviews to either to be positive or negative is to collect all the movie reviews that are pre-classified to be either positive or negative. Now we make two separate lists. The first list consists of all the words that have appeared in our positive reviews along with the number of times they have appeared and the second list consists of all the words that have appeared in our negative reviews along with the number of times they have appeared. Now using these list we can estimate the probability of a particular word appearing in a positive review or in a negative review. This can be well explained by the formula given below, we used the Scikit learn's multinomialnb () method to accomplish this task.

Figure-7

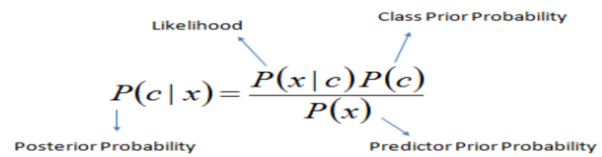
$$\hat{P}(t|c) = \frac{T_{ct}+1}{\sum_{t' \in V}(T_{ct'}+1)} = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'}) + B'}$$

Data	Accuracy
Train – within training	87.1%
Test – within training	75.3%
Overall Training data	75.68%
Overall Testing data	90.86%

Table-2

3. **Naïve Bayes approach – using chi-square feature selection:**

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods. Bayes theorem provides a way of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c). Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence. In this approach we used the chi-square feature selection strategy as explained above.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- $P(c|x)$ is the posterior probability of *class* (*target*) given *predictor* (*attribute*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

Figure-8

Data	Accuracy
Testing data - 5000 features	83.78%
Testing data – 6000 features	84.55%
Testing data – 7000 features	85.15%
Testing data – 8000 features	85.72%
Testing data – 9000 features	86.22%

Table-3

We can infer from the above table that, this approach yielded comparatively better results and we also observed the increase of model performance on a linear fashion. Which is quite obvious because, as we increase the number of influential features to be looked upon during the classification process, our classifier performs even more well thereby avoiding slight errors that occurred earlier. Now using this approach might be problematic as whenever we tend to use this statistical test multiple times, then it's said that the probability of getting at least one error increases. However, in text classification it rarely matters whether a few additional terms are added to the feature set or removed from it. Rather, the relative importance of features is important. As long as x2 feature selection only ranks features with respect to their usefulness and is not used to make statements about statistical dependence or independence of variables, we need not be overly concerned that it does not adhere strictly to statistical theory.

VI. Conclusion and Future work

In conclusion we would like to say that, after experimenting with three types of machine learning algorithms and trying to build a predictive model that would perform well in extreme circumstances. We observed that all the three approaches were quite successive in achieving its goals. The major positives for our project was our dataset. Since we were able to train our classification model with ample data, our model was able to perform reasonable well on testing data. In our case, we used the **bag of words** representation of features and hence we lost track of **positional relationships between words** and inter dependencies. Since we realized this late, we were not able to properly back track and figure an ideal alternative to address these issues. We also identified several other factors that would have led to the misclassification errors. One major factor would be to train our classification model to understand the **semantics behind word dependencies**, which we will be working in the near **future**. With respect to the third classification approach, some might argue that the feature selection strategy might prove to be in-efficient but in our case it has worked in a satisfactory level. With respect to recommendation system, our goal was to try one simple approach and we feel that we've carried out our tasks, but we haven't evaluated the performance of our recommender system on any basis and hence that would also be our future work. **For people planning to pursue projects similar to ours**, we have a one fine **suggestion to say**, one is to spend some time and go through the dataset that you've collected and notice what data that you could possibly save during the process of tokenization.

References:

1. http://nlp.stanford.edu/courses/cs224n/2004/huics_kgoel_final_project.htm
2. http://www.cim.mcgill.ca/~dpomeran/moviefinal.pdf
3. http://shop.oreilly.com/product/9780596529321.do (Programming Collective Intelligence Book)
4. http://forum.myquant.cn/uploads/default/original/1X/2065c4d1964e26331996cfa23d12acd185e3d7b6.pdf
5. http://www2.cs.uregina.ca/~hilder/my_students_theses_and_project_reports/alloteyMScProjectReport.pdf