

[**Text Analysis with Movie Review Data** ]

DS501 Case Study 3 : Textual analysis of movie reviews

11/16/2017

**Team 6 Haowen Zhu, Dekun Geng, Yin Hang, Yixin Luo, Weiqing Li**

Help a Movie Company with Movie Review Data

After seeing a movie, people are now likely to post their thoughts on SNS. As a movie company, we would like to know people’s attitude to this movie. Do they think it deserves an Oscar? Or do they think it should win a Golden Raspberry? So the question is how could we tell the positive and negative review apart.

Also, with this effort, we could be familiar with the machine learning methods of the main scikit-learn tools on analyzing a collection of text documents. It is very necessary for our future exploration on this topic.

Data Overview

We take sentient analysis on the movie reviews downloaded from http: //www. cs.cornell.edu/people/pabo/movie-review-data/, including 1000 positive and 1000 negative processed reviews. Within the folder “txt\_sentoken” are the processed down-cased text files used in Pang/:ee ACL 2004; the names of the two subdirectories in that folder, “pos” and “neg”, indicate the true classification (sentiment) of the component files according to our automatic rating classifier. Each line in each text file corresponds to a single sentence, as determined by Adwait Ratnaparkhi’s sentence boundary detector MXTERMINATOR. This data aims at sentiment-analysis experiments.

Extracting Features from Text Files

We first need to turn the text content into numerical feature vectors. In the term of text preprocessing, tokenizing and filtering of stopwords are included in a high level component that is able to build a dictionary of features and transform documents to feature vectors.

We divide each text into bags of N-grams of words. (1,1), (2,2) and (1,2), respectively, represents bags of unigrams, bigrams of words and combination of unigrams and bigrams. We come true tuning of parameter by using grid search.

Tf-idf, short for term frequency-inverse document frequency, is used as a numerical statistic that reflects how important a word is to a document. The tf-idf value increases proportionally to the number of times a word appears in the document, but is often offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Therefore, it makes sense for us to use this statistic, in the consideration of transferring words from occurrences to frequencies.

Build a classifier

After having our features, we can train a classifier to try to predict the category of a text. The third step is finding a classifier to fit the training dataset. We start it with Naïve Bayes and LinearSVC classifier, which provides a nice baseline for this task. We use Pipeline class to make these easier to work, and firstly applying it on the training dataset, then choosing the model of best performance on test dataset. Precision, recall and f1-score are provided for performance analysis of the results.

Find an Right 2 Dimentional Plot to Separate Positive and Negative Review

We would also like to find a 2 dimention plot in which the positive and negative reviews are separated.

We have tried a lot in this part. We have tried LDA(Latent Dirichlet Allocation) to find the topic, the overall sentiment polarity and sentiment polarity on selected part of the text. But they don’t give us a good result. Finally, we first do a feature engineering. We compute chi-squared stats between each non-negative feature and class. In statistics, the chi2 test is applied to test the independence of two events, where two events A and B are defined to be independent if  or, equivalently, and  . In feature selection, the two events are occurrence of the term and occurrence of the class. We then rank terms with respect to the following quantity:

|  |  |  |
| --- | --- | --- |
| \displaystyle X^2(\docsetlabeled,\tc |  |  |

Here, t and c are two class of the documents.   and means that when they are equal to 1, the document is in t or c. N is the observed frequency in  and E the expected frequency. For example,   is the expected frequency of t and c occurring together in a document assuming that term and class are independent.

An arithmetically simpler way of computing chi2 is the following:

|  |
| --- |
| begin{displaymath} X^2(\docsetlabeled,\tcword,c) = \frac{(\observationo_{11}+\o... ...ervati |

This score can be used to select the n\_features features with the highest values for the test chi-squared statistic from X, which must contain only non-negative features such as Booleans or frequencies (e.g., term counts in document classification), relative to the classes.

By chi2, we select 40 words with the highest chi2, which means they can best discriminate the text. Then, based on these word’s subjectivity, we choose 20 words as our key word. First, we have tried the count of these words as the first dimension. But it doesn’t work very well. Then we improved it with the weighted score on each word. The weighted score is the word’s sentiment polarity \* subjectivity.

But these selection has its shortcoming. It is unlikely to find some words which all documents share in common and with high discriminant ability. So there will always be a lot of samples has score around 0.

For the second dimension, we choose the SVM result of the samples. We use the SVM built before and get the confidence score of each sample. the binary case, confidence score for self.classes\_[1] where >0 means this class would be predicted. We also have tried to improve our SVM with the weighted tfidf. The weight score is that if the word is adj. or adv. , we multiply its tfidf with 1.5. But it doesn’t show any superiority to the original one.

The Outcome

According to the results, combination of unigrams and bigrams is much better than other ones in Problem 1. Meanwhile, LinearSVC performs better than others in our case.

|  |  |  |  |
| --- | --- | --- | --- |
|  | LinearSVC | MultinomialNB | BernoulliNB |
| Precision-avg | 0.87 | 0.84 | 0.83 |
| Recall-avg | 0.87 | 0.84 | 0.83 |
| F1-score-avg | 0.87 | 0.84 | 0.83 |

Table 1. Comparison between different methods in terms of combination of unigrams and bigrams

|  |  |  |  |
| --- | --- | --- | --- |
| LinearSVC | (1,1) | (2,2) | (1,2) |
| mean | 0.83 | 0.83 | 0.85 |
| std | 0.01 | 0.01 | 0.01 |

Table 2. Comparison between unigrams, bigrams and combination of them by using LinearSVC

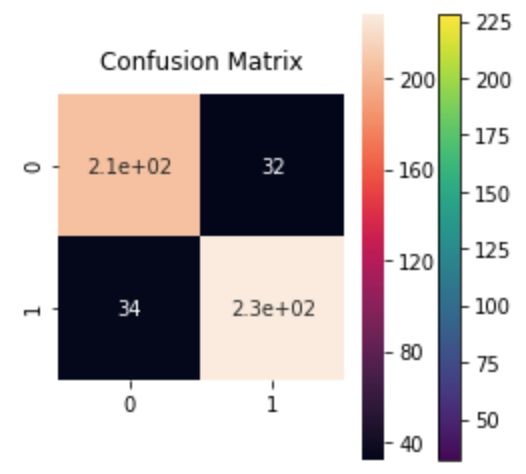


Figure 1. the confusion matrix of LinearSVC in terms of combination of unigrams and bigrams

For the Tfidf, we found the best max\_df is 0.75, and the min\_df is 0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | recall | f1-score | support |
| Neg | 0.87 | 0.82 | 0.85 | 256 |
| Pos | 0.82 | 0.87 | 0.85 | 244 |
| avg/total | 0.85 | 0.85 | 0.85 | 500 |

Table 3. The performance of SVM with Tfidf and max\_df = 0.75, min\_df = 0

And the best parameter for TfidfVectorizer is ngram(1,3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | recall | f1-score | support |
| Neg | 0.88 | 0.87 | 0.87 | 256 |
| Pos | 0.89 | 0.88 | 0.87 | 244 |
| avg/total | 0.87 | 0.87 | 0.87 | 500 |

Table 4. The performance of SVM with ngram(1,3)

We calculated scores for different values of the parameters and find out the best values of parameters for TfidfVectorizer:min\_df, max\_df and ngram\_range.

min\_df, In the vocabulary we built, terms that have the lowest document frequency . 0.000 represents 0% proportion of documents. min\_df is used for removing terms that appear too infrequently. For example: min\_df = 0.00 means "ignore terms that appear in less than 0 document".max\_df, In the vocabulary we built, terms that have the best document frequency . 0.75 represents 75% proportion of documents. max\_df is used for removing terms that appear too frequently, also known as "corpus-specific stop words". For example: max\_df = 0.75 means "ignore terms that appear in more than 75% of the documents".

ngram\_range in the shape of tuple (min\_n, max\_n) indicates the lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that min\_n <= n <= max\_n will be used. We get the best ngram\_range (1, 3) means that all values of n such that 1<=n<=3 will be used. Also we can say that unigrams, bigrams and trigrams will be extracted.

By chi2, we first select 40 words:

|  |  |  |  |
| --- | --- | --- | --- |
| Word | | | |
| american | family | overall | terrible |
| attempt | great | perfect | true |
| awful | hilarious | perfectly | unfortunately |
| bad | life | poor | war |
| boring | looks | reason | waste |
| brilliant | memorable | ridiculous | wasted |
| dull | mess | script | wonderful |
| effective | minute | stupid | world |
| excellent | nothing | subtle | worse |
| fails | oscar | supposed | worst |

Table 5. 40 words selected by Chi2

Then we choose the word which subjectivity(range 0-1) higher than 0.5

|  |  |  |  |
| --- | --- | --- | --- |
| word | POS | polarity | subjectivity |
| awful | NN | -1.0 | 1.0 |
| bad | JJ | -0.6999999999999998 | 0.6666666666666666 |
| boring | NN | -1.0 | 1.0 |
| brilliant | NN | 0.9 | 1.0 |
| effective | JJ | 0.6 | 0.8 |
| excellent | NN | 1.0 | 1.0 |
| great | JJ | 0.8 | 0.75 |
| hilarious | JJ | 0.5 | 1.0 |
| memorable | JJ | 0.5 | 1.0 |
| perfect | NN | 1.0 | 1.0 |
| perfectly | RB | 1.0 | 1.0 |
| poor | JJ | -0.4 | 0.6 |
| ridiculous | JJ | -0.3333333333333333 | 1.0 |
| stupid | JJ | -0.7999999999999999 | 1.0 |
| terrible | JJ | -1.0 | 1.0 |
| true | JJ | 0.35 | 0.65 |
| unfortunately | RB | -0.5 | 1.0 |
| wonderful | NN | 1.0 | 1.0 |
| worse | JJR | -0.4 | 0.6 |
| worst | JJS | -1.0 | 1.0 |

Table 6. 20 words with subjectivity>0.5

So for the first dimension, its histogram is:

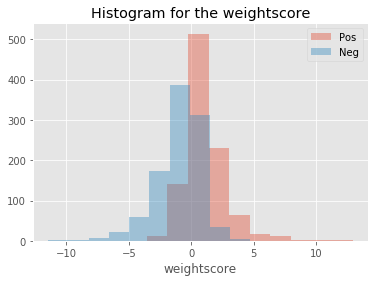


Figure2. Histogram for the weightscore(20 selected features count \* weight)

With the second dimension, we choose the confidence score of every sample with tfidf. The max\_df = 0.75, min\_df=0 and n\_gram = (1,3)

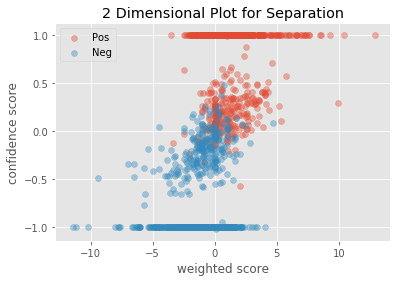


Figure 3. 2 Dimensional Plot for Separation

To test our plot, we perform an Linear Discriminant Analysis on it with test size = 0.5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | recall | f1-score | support |
| Neg | 0.93 | 0.90 | 0.92 | 494 |
| Pos | 0.91 | 0.93 | 0.92 | 506 |
| avg/total | 0.92 | 0.92 | 0.92 | 1000 |

Table 7. Performance of LDA

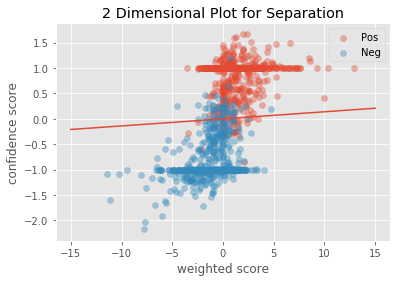


Figure4. 2 Dimensional Plot with LDA

Perspective

The outcome is not very well. We still didn’t come up to a great idea that could separate them. And we will keep on working that.