CASE STUDY 3: Textual analysis of movie reviews

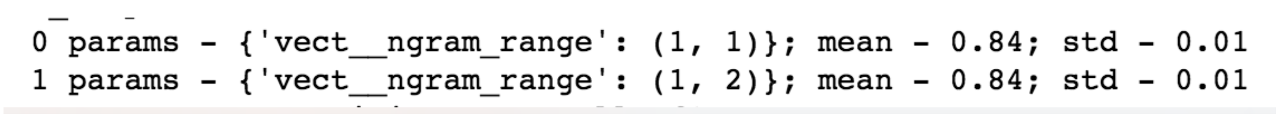
By
Hang Ding, Fangling Zhang, Qingquan Zhao, Yihao Zhou, Tongge Zhu

Introduction

Textual analysis is a method to describe and analyze contents, structures and characteristics of text comments. It has been widely used to reveal static information such as documentation, standards or user guides of legacy systems. In this case, we studied and built a sentiment analysis and polarity model to analyze a movie review dataset by fitting a linear classifier on features extracted from the text of the user messages, in order to guess whether the opinion of the author is positive or negative.

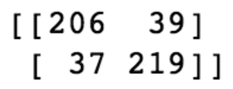
Problem 1: Basic Trainings

In order to obtain basic trainings, we followed a standard sentimental analysis procedure from http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html. After loading the dataset and define two categories as "neg" and "pos", we first split the raw data into four part to 75% training data and 25% testing data. Then, a pipeline was built containing a Tfidf-vectorizer, which filters out tokens that are either too rare or too frequent in the dataset, and a classifier, where the Linear Support Vector Machine was chosen in default for training data. In this pipeline, an additional parameter 'vect_ngram_range' was set to be either in the range of (1, 1) or (1, 2), which basically means a sentence was tokenized to both one word or two words combinations (more detailed explanation will be discussed in problem 2). With the pipeline set up, a GridSearchCV function followed by fitting the pipeline on the training set using grid search for the parameters to evaluate which of the parameter combinations would afford the best fittings. After the cross-validation process, we thus obtained the scores for each parameter set as explored by the grid search (Figure 1).



```
0 params - {'vect__ngram_range': (1, 1)}; mean - 0.84; std - 0.01
1 params - {'vect__ngram_range': (1, 2)}; mean - 0.84; std - 0.01
```

Figure 1. Optimal parameters selected by grid search.



```
[[206  39]
 [ 37 219]]
```

Figure 2. confusion matrix of prediction result

With a well-trained parameters and classifier in hand, we can then perform the prediction of which classes of our testing data belongs to by calling the predict function of grid search. The prediction result represented by a confusion matrix as shown in Figure 2 indicates that 206 out of 245 negative comments and 219 out of 256 positive comments were predicted correctly.

Problem 2: Explore the scikit-learn TfidVectorizer class

In this section, our target is to explore the effect of three parameters: min_df, max_df, and ngram_range. Before that, several terms are defined as follows:

- **Term Frequency-Inverse Document Frequency(TF-IDF):** It stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.
- **Term Frequency(TF):** It measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:
  TF(t) = (Number of times term t appears in document) / (Total number of terms in document).
- **Inverse Document Frequency (IDF):** It measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:
  IDF(t) = log_e(Total number of documents / Number of documents with term t in it).
- **min_df:** The vocabulary will ignore the terms that have a document frequency strictly lower than the given min_df values. In our case, this parameter could be an absolute count number if it is integer, or percentage representing a proportion of documents if it is a float. This parameter is ignored if vocabulary is not None.
- **max_df:** The vocabulary will ignore the terms that have a document frequency strictly higher than the given max_df values. In our case, this parameter could be an absolute count number if it is integer, or percentage representing a proportion of documents if it is a float. This parameter is ignored if vocabulary is not None.
- **n-gram:** in the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sequence of text or speech.

- **ngram_range:** The lower and upper boundary of the range of n-values for different n-grams to be extracted. When upper boundary of ngram_range is larger, the shape of vectorized result is larger and we will get more features.

With these terms clarified, we then explored the three TfidVectorizer parameters mentioned above in altering features. The logic behind this tuning process is that we change one parameter and fix the other two, thus we are able to observe the change of features accordingly. Firstly, max_df and n_gram_range were fixed, and a systematic screening of min_df value from 0 to 99 was performed. The plotting result as shown in Figure 3 clearly indicates that the feature numbers decreased exponentially once the min_df increases.

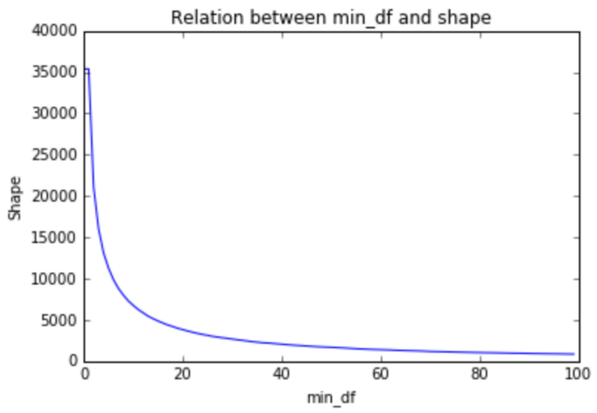


Figure 3. Screening of min_df values.

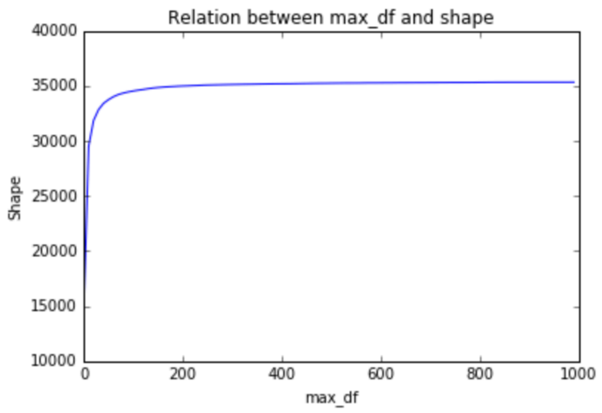
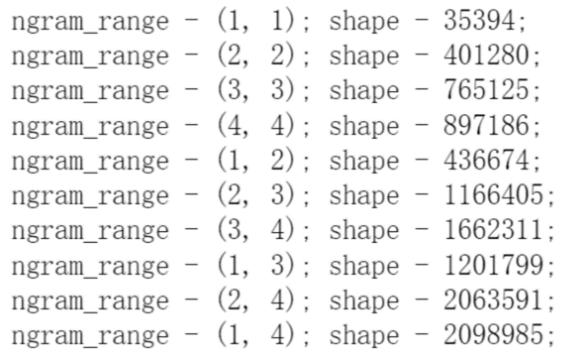


Figure 4. Screening of max_df values.

The similar screening process from 0 to 1000 was also applied to max_df by fixing min_df and n_gram_range accordingly. A similar result as shown in Figure 4 suggests that the feature numbers increased exponentially after max_df increases. Both results in Figure 3 and Figure 4 is rational because once min_df increases, more features with document frequency lower than min_df has been removed from features, whereas less features with document frequency higher than max_df values will be removed when max_df values increases.

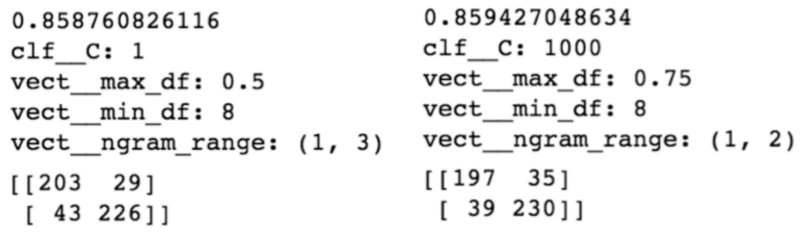

```
ngram_range - (1, 1); shape - 35394;
ngram_range - (2, 2); shape - 401280;
ngram_range - (3, 3); shape - 765125;
ngram_range - (4, 4); shape - 897186;
ngram_range - (1, 2); shape - 436674;
ngram_range - (2, 3); shape - 1166405;
ngram_range - (3, 4); shape - 1662311;
ngram_range - (1, 3); shape - 1201799;
ngram_range - (2, 4); shape - 2063591;
ngram_range - (1, 4); shape - 2098985;
```
Figure 5. Exploration of parameter n_gram_range.

Meanwhile, the effect of ngram_range parameter in modifying features were also explored in the similar fashion. A combination set of {(1,1), (2,2), (3,3), (4,4), (1,2), (2,3), (3,4), (1,3), (2,4), (1,4)} was searched one by one with fixed values of min_df and max_df. The results in Figure 5 show that more feature numbers are obtained after the gap of ngram_range range is increased. This result is clearly fit the definition of n_gram_range definition because when we look these data in detail, the feature number 436674 in range (1, 2) is exactly the sum of feature numbers of 35394 in range (1, 1) and 401280 in range (2, 2).

Problem 3: Machine Learning Algorithms

In this section, three classifiers including Support Vector Machine, K-Nearest Neighbors, and Gaussian Naive Bayes were tested with optimization of parameters of these classifiers, as well as previously discussed min_df, max_df and n_gram_range values. For all these three classifiers, the max_df was search within {0.5, 0.75, 1.0}, min_df value was searched within {1,2,4,8,16,32}, and ngram_range value was set among {(1, 1), (2, 2), (1,2), (1,3), (1,4), (2,4)}. For each classifier, the optimal parameters will be discussed separately in the following.

Support Vector Machine (SVM): SVM is a 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. In our case, we tested the SVM classifier with the error penalty parameter C to be both 1000 and 1. When C equals 1000, the grid search offers a best score of 0.86 with the optimal min_df, max_df and n_gram_range values set to be 8, 0.75, (1, 2). On the other hand, if C was set to be 1, which means very low tolerance in generating hyperplane, the grid search offers a best score of 0.86, with the optimal min_df, max_df and n_gram_range values set to be 8, 0.5, (1, 3). The predicting result as shown in Figure 6 indicates that in both cases, the prediction accuracy is quite similar with very minor influence by the error penalty parameter change.

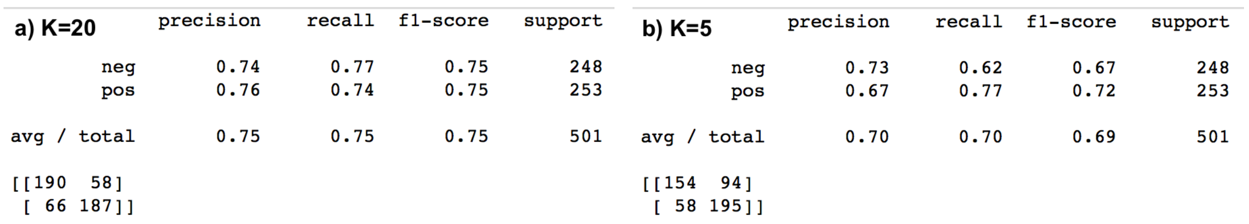


```
0.858760826116
clf__C: 1
vect__max_df: 0.5
vect__min_df: 8
vect__ngram_range: (1, 3)
[[203  29]
 [ 43 226]]
```

```
0.859427048634
clf__C: 1000
vect__max_df: 0.75
vect__min_df: 8
vect__ngram_range: (1, 2)
[[197  35]
 [ 39 230]]
```

Figure 6. Prediction result of SVM with C=1(left) and C=1000(right).

K-Nearest Neighbors (K-NN): K-NN is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure. In our studies, we explored its efficiency when K was set to both 20 and 5. When k was set equals 20, the grid search offers the best score of 0.73 with the best min_df, max_df and n_gram_range values set to be 2, 0.5 and (2, 4) respectively. The accuracy of K-NN is even worse when K is set to be 5. In this case, the grid search only affords the best score of 0.68. The predicting results of test data in both cases were shown in Figure 7.

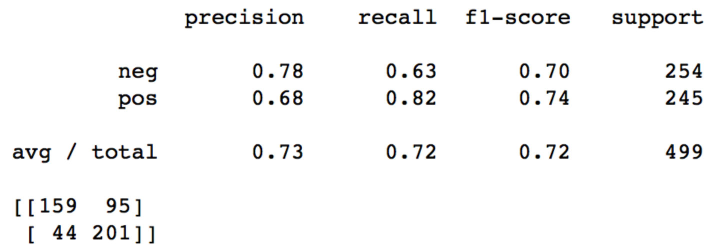


a) K=20
```
             precision    recall  f1-score   support

        neg       0.74      0.77      0.75       248
        pos       0.76      0.74      0.75       253

avg / total       0.75      0.75      0.75       501

[[190  58]
 [ 66 187]]
```

b) K=5
```
             precision    recall  f1-score   support

        neg       0.73      0.62      0.67       248
        pos       0.67      0.77      0.72       253

avg / total       0.70      0.70      0.69       501

[[154  94]
 [ 58 195]]
```

Figure 7. Prediction result of K-NN with a) K=20; and b) K=5

Gaussian Naive Bayes: GaussianNB is a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Unfortunately, GaussianNB produced a very similar predicting result as K-NN, worse than SVM (Figure 8).



```
             precision    recall  f1-score   support

        neg       0.78      0.63      0.70       254
        pos       0.68      0.82      0.74       245

avg / total       0.73      0.72      0.72       499

[[159  95]
 [ 44 201]]
```

Figure 8. Prediction result of GaussianNB.

Two Wrong Prediction Examples: We picked two examples from all the incorrect prediction results to analyze why the prediction is wrong. The first example is review for movie "Love is devil". The actual sentiment of the review is positive, whereas the prediction estimates the review to be negative. By reading the review context carefully, we believed it is because the review used many sentimentally negative words to describe the abnormal personality of the main character in the movie. But overall the review appraises the movie to be well-done. The second example is review for movie "Star Wars the Phantom Menace". Our prediction is positive but the actual sentiment is not. We believed in this case, the reason is that a large portion of the review is written about how good the previous Star Wars movies are. The algorithm could not tell whether these positive words is to this movie or previous movies.

In summary, based on our studies and parameters we screened for, we identified that Linear Support Vector Machine offers us the highest prediction accuracy in our case. It is probably because that Support Vector Machine is more suitable for a large number of predictors compared to K-NN.

Problem 4: Open Ended Question: Finding the right plot

In this section, our purpose is trying to find a two-dimensional plot in which the positive and negative reviews can be separated. To do so, we have tried several methods with or without the assistance of SelectKBest function in scikit-learn module, including Principle Component Analysis (PCA) dimensional reduction, K-means unsupervised machine learning algorithm, as well as a simple split by specific positive and negative words. Although these methods are not able to separate the positive and negative reviews completely, we do show some promising progress on separation, and partially split these two classes into two parts in two-dimensional space.

Our first two experiments focus on using Principal Components Analysis (PCA), a method of identifying the vectors describing the greatest variance within a dataset. This vector space can be used as a means of classifying other vectors in the original space according to their similarity. A naive approach was accomplished by simply taking all the eigenvectors of a dataset's covariance matrix and reducing to two-dimensions (Figure 9). Since it is a sparse matrix, none of features can cover a large percentage of features, thus the percentage of variance explained by each of the selected components are not high enough for separating (explained variance ratio of first two components: [0.00412175, 0.00374217]). A better PCA approach was then performed by first picking up a number of features chosen by SelectKBest function in Scikit-learn before doing PCA. A series of parameters for SelectKBest was tested, and 600 features was eventually used to produce a much better separating result as shown in Figure 10.

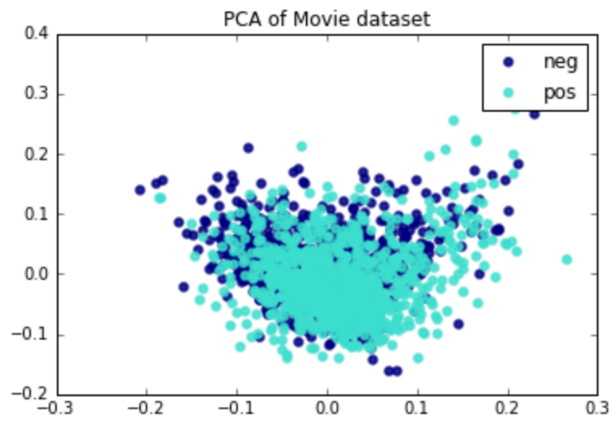


Figure 9. plotting after PCA using all features in data

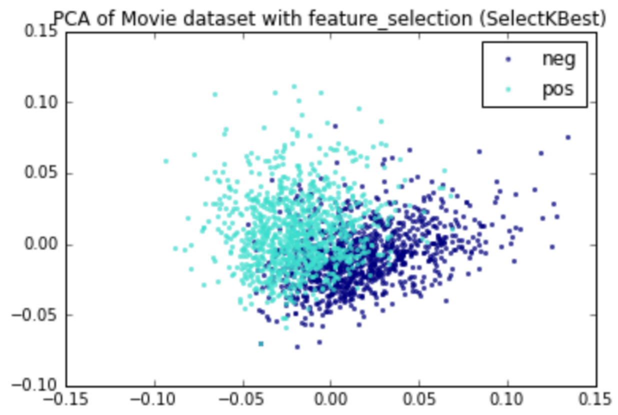


Figure 10. plotting after PCA using 600 features selected by SelectKBest function

k-means unsupervised machine learning was also used for this experiment. k-means is one of the simplest algorithms to solve the clustering problems by defining k centers based on minimizing the sum of squares distance from the centroids of the partitions to the samples. Each of these centroids are supposed to be placed in the centers of each clusters. In our case, two target classes are expected from raw data, therefore we set the parameters of k to be 2. By computing and minimizing the sum of Euclid squares distance, two centroids should be able to obtained. For each single data in our dataset, the distance of this data spot to each centroid were used as X and Y axis for plotting purpose. Ideally, a data spot belonging to one class should be closer to the centroid representing to its class as predicted by k-means. However, the results as shown in Figure 11 is not perfect enough for separating purpose. None of other parameters of either min_df, max_df or ngram_range could further improve this results.

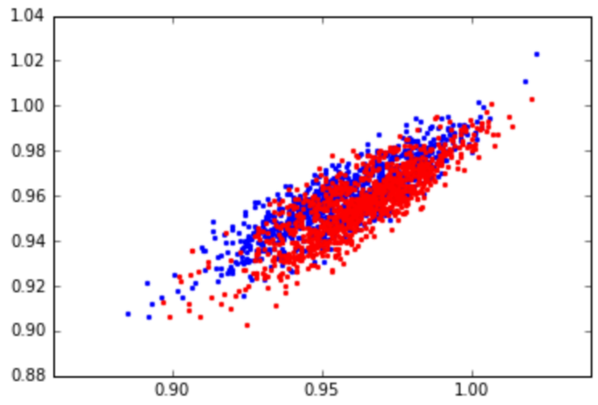


Figure 11. Plotting of distances to centroids calculated by k-means algorithms using all features

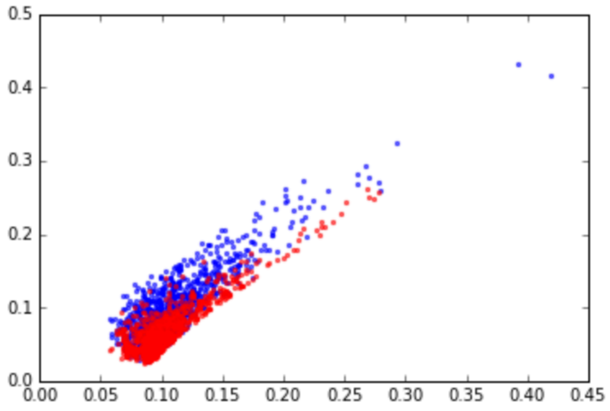


Figure 12. Plotting of distances to centroids calculated by k-means algorithms using 20 best features selected by SelectKBest

Inspired by the improvement of PCA results after decreasing features with the help of SelectKBest function in Scikit-learn, we also bring this method to our k-mean approach. A series of numbers was set up in selecting features and eventually, 20 best features achieved a much better separation in positive and negative reviews (Figure 12).

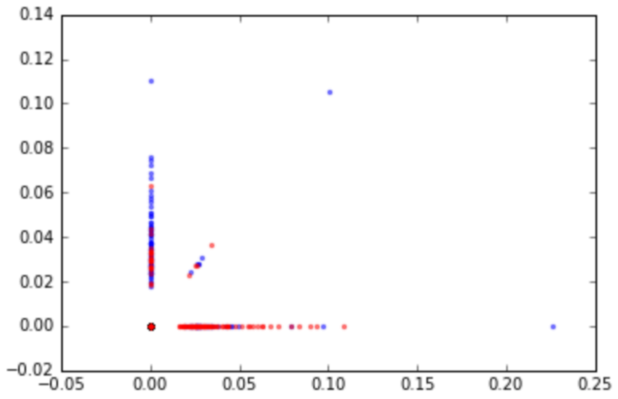


Figure 13. Separation of reviews based on a specific positive word "amazing" and negative word "terrible".

The last experiment we did is trying to separate positive and negative reviews simply based on one representative positive word and one representative negative word. We surprisingly find that by choosing "amazing" and "terrible" as two features, the Tfidf values of all reviews are also visibly separated (Figure 13). Meanwhile, we do understand that this very basic splitting method might be biased because we manually choose two single features. Also, it might also be more or less misleading since whatever two features we choose, they cannot cover all related features because of the ngram_range parameters we set up.

Reference:

All the term definitions are cited either from Wikipedia or the Scikit-learn official documentations.