Casestudy 3

Shaowei Gong, Houjue Wang, William Grudzinski, Wenjing Li

1. Dataset

We collect our data from <http://www.cs.cornell.edu/people/pabo/movie-review-data/,> and download the polarity dataset v2.0 dataset. This dataset is a folder which contains 2 subfolders containing 1000 positive or negative .txt format comments files separately.

1. Motivation

Analyzing text documents is special, because we cannot train our model with text directly. Thus we need to find a way to quantify our text data and convert them to a format that can be parsed by model. Then we can use these converted data to train our machine learning model and make prediction.

1. Analyze data

1) In order to analyze these text documents data, we need a model that can convert the text into variable. In this case we use sentimental analysis. Sentimental analysis can convert a collection of raw documents to a matrix of TF-IDF features. So, generally speaking, we convert our text documents folder into a high dimension sparse matrix that can be used to train our machine learning model. In detail, we split our dataset, randomly select 75percent of dataset as our training set and other 25percent as test set. And then use in sklearn.feature\_extraction.text to fit and transform training set first and then transform TfidfVectorizerthe test set.

1. After converting our text documents into a matrix of TF-IDF features, we can use this to train our machine learning model. In this case, we choose to use LinearSVC, KNeighborsClassifier, RandomForestClassifier as our model, which usually perform well in high dimension sparse data situation.

In this case, we choose use\_idf=True,min\_df=4,max\_df=0.5,ngram\_range=(1,2) as parameter in TfidfVectorizer to fit and transform text document into a matrix of TF-IDF features.

LinearSVC:

1. use penalty='l2', loss='squared\_hinge', dual=True, tol=0.0001, C=1.0, multi\_class='ovr', fit\_intercept=True, intercept\_scaling=1, class\_weight=None, verbose=0, random\_state=None, max\_iter=1000 as parameters.

We get 0.866 prediction accuracy in our test set.

confusion\_matrix:

Capture1

1. use penalty='l2', loss='squared\_hinge', dual=True, tol=0.1, C=10, multi\_class='ovr', fit\_intercept=True, intercept\_scaling=1, class\_weight=None, verbose=2, random\_state=None, max\_iter=1000 as parameters.

We get 0.87 prediction accuracy in our test set.

confusion\_matrix:

Capture2

KNeighborsClassifier :

1. use n\_neighbors=5, weights='uniform', algorithm='auto', leaf\_size=30, p=2, metric='minkowski', metric\_params=None as parameters.

We get 0.666 prediction accuracy in our test set.

confusion\_matrix:

Capture3

1. use n\_neighbors=700, weights='distance', algorithm='auto', leaf\_size=30, p=2, metric='minkowski', metric\_params=None as parameters.

We get 0.786 prediction accuracy in our test set.

RandomForestClassifier:

1. use n\_estimators=1000, criterion='gini', max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features='auto', max\_leaf\_nodes=None, bootstrap=True, oob\_score=False, random\_state=None, verbose=0, warm\_start=False, class\_weight=None as parameters.

We get 0.83 prediction accuracy in our test set.

confusion\_matrix:

Capture5

1. use n\_estimators=500, criterion='gini', max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features='auto', max\_leaf\_nodes=None, bootstrap=True, oob\_score=False, random\_state=None, verbose=0, warm\_start=False, class\_weight=None as parameters.

We get 0.818 prediction accuracy in our test set.

confusion\_matrix:



1. After trying three models above, it is obvious that selection of parameters has significant impact in prediction accuracy. So how can we selection a good set of parameters is critical. In this case, we choose to use grid search combined with build-in pipeline function in python to select a relatively good combination of parameter, both in sentimental analysis phase and training phase.

LinearSVC:

clf\_\_C: 10

clf\_\_dual: False

clf\_\_tol: 0.1

is a relatively good set of parameter for LinearSVC model

vect\_\_max\_df: 0.8

vect\_\_min\_df: 5

vect\_\_ngram\_range: (1, 2)

is a relatively good set of parameter for TfidfVectorizer model.

We get 0.85 prediction accuracy in our test set.

confusion\_matrix:

Capture7

KNeighborsClassifier:

clf\_\_n\_neighbors: 400

clf\_\_weights: 'distance'

is a relatively good set of parameter for KNeighborsClassifier model

vect\_\_max\_df: 0.5

vect\_\_min\_df: 10

vect\_\_ngram\_range: (1, 2)

is a relatively good set of parameter for TfidfVectorizer model.

We get 0.788 prediction accuracy in our test set.

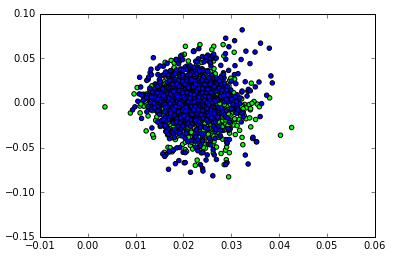
confusion\_matrix:

Capture8

1. We attempt to visualize our data in a 2D plot. Thus, we need make some efforts to reduce the dimension of our data. In this case, we use TruncatedSVD in sklearn.decomposition which can work with sparse matrices efficiently.

We use use\_idf=True,min\_df=10,max\_df=0.5,ngram\_range=(1,1) as parameters in TfidfVectorizer model to fit and transform text data. After processing, we get a matrix with 8109 features.

Then, we use TruncatedSVD to reduce the dimension. Unfortunately, the first component of TruncatedSVD model only covers 0.03284155 variability, the second component only covers 0.00465772 variability.



Using these 2 component to plot, dots mix together with each other. And we fail to find a better way to reduce dimensions.

1. Conclusion

For this dataset, as show above, when we use use\_idf=True, min\_df=4, max\_df=0.5, ngram\_range=(1,2) as parameter in TfidfVectorizer to fit and transform text document and use LinearSVC with penalty='l2', loss='squared\_hinge', dual=True, tol=0.1, C=10, multi\_class='ovr', fit\_intercept=True, intercept\_scaling=1, class\_weight=None, verbose=2, random\_state=None, max\_iter=1000 as parameters to train our model. We can get relatively good prediction accuracy.

We get 0.87 prediction accuracy in our test set.

confusion\_matrix:

Capture2