**Fan Base Prediction**

Introduction

The textual content of a tweet can reveal information about the author of tweets posted by fans of different teams during a sport game describe similar events in different terms and sentiments. One aspect is that recognizing that supporting a sport team has a lot to do with the user location so that we can try to use the textual content of the tweet posted by a user to predict their location.

The textual data is form tweets including **#superbow**l, posted by the users whose specified location is either in the state of Washington or Massachusetts.

Base on this information and gathering labeled data, we can predict Train a binary classifier to predict the location of the author of a tweet (Washington or Massachusetts), given only the textual content of the tweet. Several machine learning method will be applied.

Data Extraction

The very first thing to do is filter the data so that we can have tweets posted by the users whose specified location is either in the state of Washington or Massachusetts. Then we can label the data as shown in Table 1.

Table 1 Features and Labels

|  |  |
| --- | --- |
| **Location** | **Label** |
| Washington | 0 |
| Massachusetts | 1 |

The method of labeling the data is using the bag of words. We can build a list for Washington and New England independently. Then consider the tweets that include the substrings in the list in the location field. In addition, we exclude “DC” and “D.C” to get rid of the DC fans in capital. The list can include all the cities, towns and areas as well as the states. We store the data as Python pickle data as following:

* “Washington.p”: Containing the name string list of cities, towns and areas for Washington
* “NewEngland.p”: Containing the name string list of cities, towns and areas for NewEngland

In summary, we can have the fan base prediction data in Table 2. Washington got nearly 40% while New England got 60% or so in the extracted data.

Table 2 Fan Base Prediction Data

|  |  |
| --- | --- |
| **Total Number** | 56765 |
| **Massachusetts** | 34021 |
| **Washington** | 22744 |

Data preprocessing

We split data into 85:15 for training and testing at first. So 85% random data will be use as training data while the left 15% will act as testing data.

For textual data, the method we applied in project one can be also handy for the tweet analysis. The first step is to do in the data preprocessing is to drop certain words or terms so as to avoid unnecessarily large feature vectors, terms that are too frequent in almost every document, or are very rare, are dropped out of the vocabulary. Here, we use the “*WordNetLemmatizer*” to do the lemmatize so as to return the words back to stem. Also we applied the stop words, which acts as a filter to get rid of the words that are too common. The parameters for “*min\_df*” is 2 and for “*max\_df*” is 0.99.

After we first tokenize each document into words and exclude the stop words, punctuations as well as using stemmed version of words, the TFxIDF vector representations can be applied. Considering using the normalized count of the vocabulary words in each document to build representation vectors.

where 𝑡𝑓(𝑡,𝑑) represents the frequency of term 𝑡 in document d, and inverse document frequency is defined as:

where 𝑛 is the total number of documents, and 𝑑𝑓(𝑡) is the document frequency; the document frequency is the number of documents that contain the term 𝑡. The size of TFxIDF sparse matrix is shown in Table 3.

Table 3 The Size of TFxIDF of Data Set

|  |  |
| --- | --- |
| Rows | Columns |
| 56765 | 10445 |

For large scale data, dimension reduction is rather essential. Here we apply LSI to do the truncate SVD. When applying LSI to the TFxIDF matrix corresponding to the tweet textual data, the parameter picked denoted as *k*. After this each document is mapped to a *k*-dimensional vector. By selecting a non-sparse subset of the total feature to reduce the dimensionality using mean-squared errors, it minimizes mean squared residual between the original data and reconstruction from its low-dimensional approximation. By the way, the LSI representation is obtained by computing left and right singular vectors corresponding to the largest values of the term-document TFxIDF matrix. We have D to as the TF-IDF matrix, then the SVD of the matrix could be expressed as:

D = U Σ V

where U and V are both unitary matrices and Σ is the corresponding singular value matrix.

Using truncated SVD, we can briefly have a look of how many variances are retained in this r = 1000 dimensions.

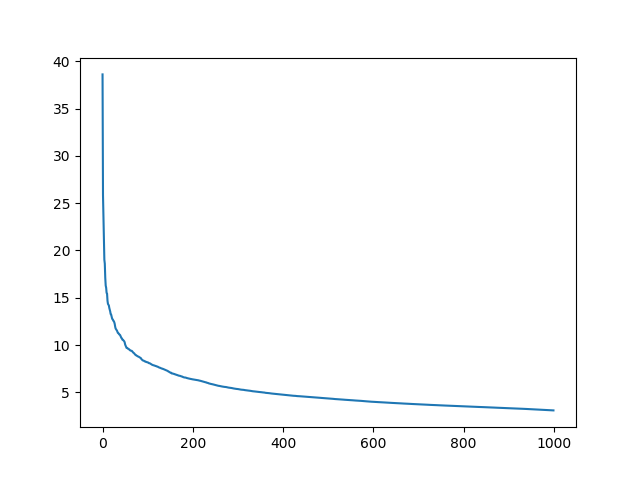


Fig. Singular Value for Truncate SVD of 1000

In the following part, we will apply 200 as the Truncate SVD dimension, so when applying LSI to the TFxIDF matrix corresponding to the tweet textual data, the parameter picked denoted as *200*. After this each document is mapped to a *200*-dimensional vector.

The next part will show the result of different classifier based on distinct machine learning models base on the training set get above.

Learning and Prediction

In this part, we will go over training seven popular classifiers to predict the outcomes with ROC curve, plotted and reporting confusion matrix, and calculate accuracy, recall and precision.

Support Vector Machine

Table 8. Confusion Matrix--SVC

|  |  |  |
| --- | --- | --- |
|  | Predicted MA | Predicted WA |
| Actual MA |  |  |
| Actual WA |  |  |

Logistic Regression

K-Nearest Neighbors

Naïve Bayer’s

AdaBoost

Random Forest Classifier

Neural Network Classifier