Visual Tweets Sentimental Analysis

Lilong Jiang (jiang.573)

April 23, 2014

1 Introduction

In this project, I build a visual online tweets sentimental analysis tool, which allows users to do the sentimental analysis for a certain topic. The user interface is shown in Figure 1. We implement two classifiers and compare the performance and accuracy in both classifiers.

2 Dataset

We get our datset from the Sentiment140 [1]. The training dataset in this dataset is collected by the Twitter API, whose content contains the emoticons. The test dataset in this dataset is manually annotated.

3 Preprocessing

There are several challenges for the tweets processing: firstly, the tweets are very short, limited as 140 characters. Secondly, the tweets are very sloppy and it contains a lot of misspellings, slangs, abbreviations and emoticons. The preprocessing step is very important for the later classification. We preprocess all the tweets as follows:
(1) Remove usernames and urls.
(2) I use an emoticon dictionary [**?**], in which each emoticon is labeled as emotion(positive or negative). The emoticons in the tweets are replaced by happy or sad.
(3) Replace all sequences of repeated characters by three characters, for example, convert goooood to goood.
(4) Expand abbreviations with a dictionary[2]. For example, lol is converted into laughing out loud.
(4) NLTK[3] is used to tokenize the tweets, remove the stop words, punctuations, and lemmatization.

[1]`http://www.sentiment140.com/`
[2]`http://www.noslang.com/`
[3]`http://www.nltk.org/`

(5) PyEnchant[4] is used to check whether the token is a correct English word.

4 Classifiers

We use two classifiers in this project. One is the naive Bayes classifier and the other is the SVM.

4.1 Naive Bayes Classifier

The Naive Bayes classifier is a simple probalistic classifier which is based on Bayes theorem. Each tweet will be classified into the class with the highest posterior probability.
$C_map = argmax_{c \in C}(logP(c) + \sum_{1 \geq k \leq n_d logP(t_k|c)})$
The conditional probability of a particular word given a class as the relative frequency of term t in documents belongs to class c:
$P(t|c) = \frac{T_{ct}+1}{\sum_{t \in V}(T_{ct}+1)} = \frac{T_{ct}+1}{\sum_{t \in V}(T_{ct})+B}$
where B is the number of documents.
We use add-one smoothing to address the problem that a particular word doesn't appear in a particular class.

4.2 Feature Selection

I implement two feature selection methods. One is the frequency-based feature selection. The other is based on the mutual information.

4.2.1 Frequency-based Feature Selection

4.2.2 Mutual Information

Mutual INformation is used to measure how much a term t contributes to the class c.
$I(U;C) = \frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_{0.}N_{.0}}$
where N_{10} is the number of documents that contain t and are not in class c. N_{11} is the number of documents that contain t and are in class c. N_{01} is the number of documents that dont contain t and are in class c. N_{00} is the number of documents that dont contain t and are not in class c. $N_{1.}$ is the number of documents that contain t. $N_{1.}$ is the number of documents that contain t.
We use add-1 smoothing for each count to void the divided by 0.

4.3 SVM Classifier

We use Libsvm[5] to classify the tweets.

[4]`http://pythonhosted.org/pyenchant/`
[5]`http://www.csie.ntu.edu.tw/ cjlin/libsvm//`

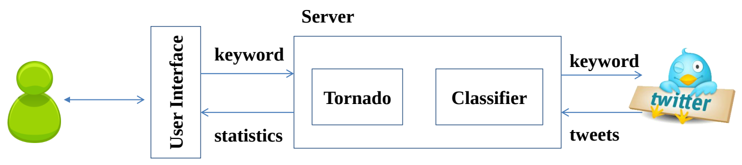

Figure 1: Architecture

4.4 Experiment

We compare these two classifier on the performance and accuracy. 2000 tweets are used to train the classifers (1000 tweets are positive and 1000 tweets are negative) and 359 tweets are used to test the classifier.

4.4.1 Parameter Sensitivity

Shrinking Size
We plot the accuracy of Bayes classifier with the changing of the shrinking ratio in the mutual information. The result is shown in Figure. As we can see

4.4.2 Performance

4.4.3 Accuracy

5 Online Sentimental Analysis

5.1 Architecture

In the front end, we use d3[6] to visualize the final result and Tornado[7] as a webserver. When users submit a topic, the server gets 1000 tweets from the twitter through twitter api and analyzes each tweet and returns the statistics information and tweets to the client.

5.2 Data Collection

TwitterSearch[8] is a python-based interface which implements the Twitter Search API.

5.3 Online Analysis

[6]http://d3js.org/
[7]http://www.tornadoweb.org/
[8]http://twittersearch.readthedocs.org/en/latest//