CS583 Data Mining and Text Mining

Project II – Twitter Sentiment Analysis

Project Report

Project Members,

Muthiah Muthu Palaniappan

Lakshmi Kamala Kotha

The University of Illinois at Chicago

Fall 2015

Abstract

Since the last decade there has been a huge growth in the use of Social Websites and in the past few years the Micro Blogging sites have also grown in pace. Spurred by the growth of Micro Blogging sites like Twitter, companies and media organizations started seeking for information about what people feel and think about their product and services, through the micro blogging sites. There are a lot of tweets specific to each product and service but they would be of use only if the companies and organizations get to know if the tweets are supporting their product or not, which is exactly what this project of ours does. It finds the orientation of each tweet whether to be Positive, Negative or Neutral. Based on the summarized tweets orientation the companies and organizations can improve and market their products and services better.

Introduction

Problem Statement

A collection of tweets for both the presidential candidates Barack Obama and Mitt Romney, taken during the election campaigning were given. Every tweet corresponding to the respective presidential candidate would either support (positive) or oppose (negative) their standing in the presidential race. Apart from supporting and opposing there are other tweets that either does not express any opinion (neutral) about the presidential candidate or expresses a mixed opinion. We had to build and train a classifier that can learn from the set of labeled examples and then help in classifying the test tweets that are not labeled.

Building a Classifier

The excel sheet containing the tweets for both the presidential candidates Barack Obama and Mitt Romney had noise in it and it had four different kinds of class labels in it as stated above, of which we are interested only in three. The following are the techniques that are involved in building a classifier,

- Reading the required tweets
- Tweet Pre-Processing/Tweet Cleansing
- Implementation

The classifier was built and trained in Java with the help of lingpipe libraries for Natural Language Processing. The programmed implementation of all the techniques stated above are discussed in detail.

Techniques

Reading the Required Tweets

Reading all tweets from the given excel was done using JExcel libraries, which has a convenient way of reading, writing and modifying excel spreadsheets dynamically. The excel sheet with the given tweets had them labeled into four different classes. The classes are,

- ✓ Positive – Represented by 1

- ✓ Negative – Represented by -1
- ✓ Neutral – Represented by 0
- o Mixed – Represented by 2

Of the above given classes we are interested only in the classes that are ticked (Positive, Negative and Neutral).

The tweets as they are being read are checked for their assigned class labels, based on which they are taken for further processing.

Tweet Pre-Processing

The tweets to train the classifier are processed to remove noise and unwanted content in it so as to build the best classifier from it. A lot of preprocessing techniques are involved in getting the tweet ready for the classifier. The order in which these techniques are applied one after the other played a crucial part in training the classifier. Below are the preprocessing techniques that were employed in the cleaning the tweets in the order as given,

Convert tweets to lowercase – All the characters in the tweets that are read from excel were first converted into lower case characters.

Removing Hyperlinks – All the hyperlinks starting with 'http://' and 'https://' that are contained in tweets were removed using regular expression.

Removing HTML Tags – All the HTML tags like ', , <a>, ' were removed from the tweets using jsoup Java HTML parser. This API provides a convenient way of stripping all the HTML tags in a given string which came in handy for removing all the HTML tags from tweets.

Removing @ Tags – There were a lot of tweets containing names which were usually denoted using the @ tags. Being a noun they don't directly help in deciding the orientation of the tweets because of which removing them produced better results. All the @ tags in a tweet were removed using regular expressions.

Removing # Tags – Most of the tweet contained a hashtag which conveys the topic on which the tweet is about. For classification the topics did not help much in deciding the orientation because of which we removed the hash tags as well using the regular expression in java.

Removing Emoticons – Presence of emoticons like :), ;), :@ etc. were found in it tweet which are not required for classification and so all the emoticons were removed from the tweets.

Removing Special Characters – There were a lot of special characters like '&, *, /,..' that were contained in the tweet. All the special characters in the tweet were removed using regular expression in java.

Removing repeating characters – There were a lot of words in tweets that had a same character that was repeated for more than 3 times in the same word like 'hellooooooo' which is usually done to express the excitement. Since using the word as such to train the classifier is not going to help in building a better classifier, all such words were checked for repeating

characters for occurrences of more than 3 and the same was replaced with just one character of the same instead of 3 or more characters. This again was done using regular expression in java.

Stop words removal – Stop words usually refer to the most common words in a language that are needed for writing a sentence to make them complete but they help very little in natural language processing because of which they are removed. All the stop words in the tweet were removed with the help of a stop word list.

Removing words of length less than or equal to 2 – All the words in the tweets whose length were 2 or less than 2 were removed from the tweet as they are similar to stop words and don't help in natural language processing.

Spelling Check - After all the above preprocessing steps the remainder words in a tweet were found to have the common typographical mistakes which were corrected using the Jazzy Spell checker for java.

Stemming – Finally, before sending the tweet for training the classifier, all the words in the tweet were stemmed using Porters Stemming implementation in lingpipe for java. This helps in finding the root word of every word.

Implementation

Two classifiers were implemented, of which one was using Naïve Bayes Classifier and the other was using the Dynamic Language Model Classifier.

Training a classifier

- All the preprocessed tweets were put into the respective text documents based on the assigned class label for each tweet.
- The lingpipe's implementation of classifier were used for building a classifier with the preprocessed tweets. The classifier is trained for the Positive tweets with the tweets that were placed in the text document corresponding to the positive tweets. Likewise the classifier is trained for both Negative and Neutral tweets.
- The trained classifier is compiled to an object output that is saved in the local hard disk for classification of unlabeled tweets. Saving it to the hard disk saves time of training the classifier time and again, when some tweets are to be evaluated.

Classifying an unlabeled tweet

- The classifier is initialized with the help of the object file that was saved into the local hard disk after training.
- The unlabeled tweet is then passed to the classifier to get the class that it belongs to.

Naïve Bayes Classifier – Naïve Bayes classifier is a simple probabilistic classifier based on the Bayes theorem with strong independence assumption between features. This classifier was implemented using Naïve Bayes classifier implementation in lingpipe library.

Dynamic LM Classifier – This works based on the concept of Language Models for text classification. It's dynamic as it employs a lot of base classifiers like Conditional Classifier, Ranked Classifier, Scored Classifier, etc. for Language Modeling. Training is based on a multivariate estimator for the category distribution and dynamic language models for the per-category character sequence estimators. This classifier was also implemented using the lingpipe library.

Evaluation

Naïve Bayes Classifier

	Obama (in %)	Romney (in %)
Positive Precision	53.69	50.63
Positive Recall	41.24	41.82
Positive F-Score	46.65	45.80
Negative Precision	52.14	63.06
Negative Recall	61.92	71.67
Negative F-Score	46.65	45.80
Neutral Precision	50.00	41.34
Neutral Recall	50.44	36.58
Neutral F-Score	46.65	45.80

Dynamic Language Model Classifier

	Obama (in %)	Romney (in %)
Positive Precision	61.16	60.75
Positive Recall	45.19	41.82
Positive F-Score	51.98	49.54
Negative Precision	52.97	62.84
Negative Recall	67.44	75.94
Negative F-Score	59.34	68.77
Neutral Precision	53.88	44.42
Neutral Recall	51.03	38.02
Neutral F-Score	52.42	40.97

Conclusion

The classifier was trained and tested using the give tweets. As in the previous section it is evident that, of the two classifiers that were implemented, the Dynamic Language Model Classifier outperformed the Naïve Bayes classifier. However, high accuracy was not achieved. Steps for enhancing the classifier further would be to implement Part of Speech tagging, bagging and the TF-IDF of a word to finds its importance in a tweet. This would improve the quality of the classifier.