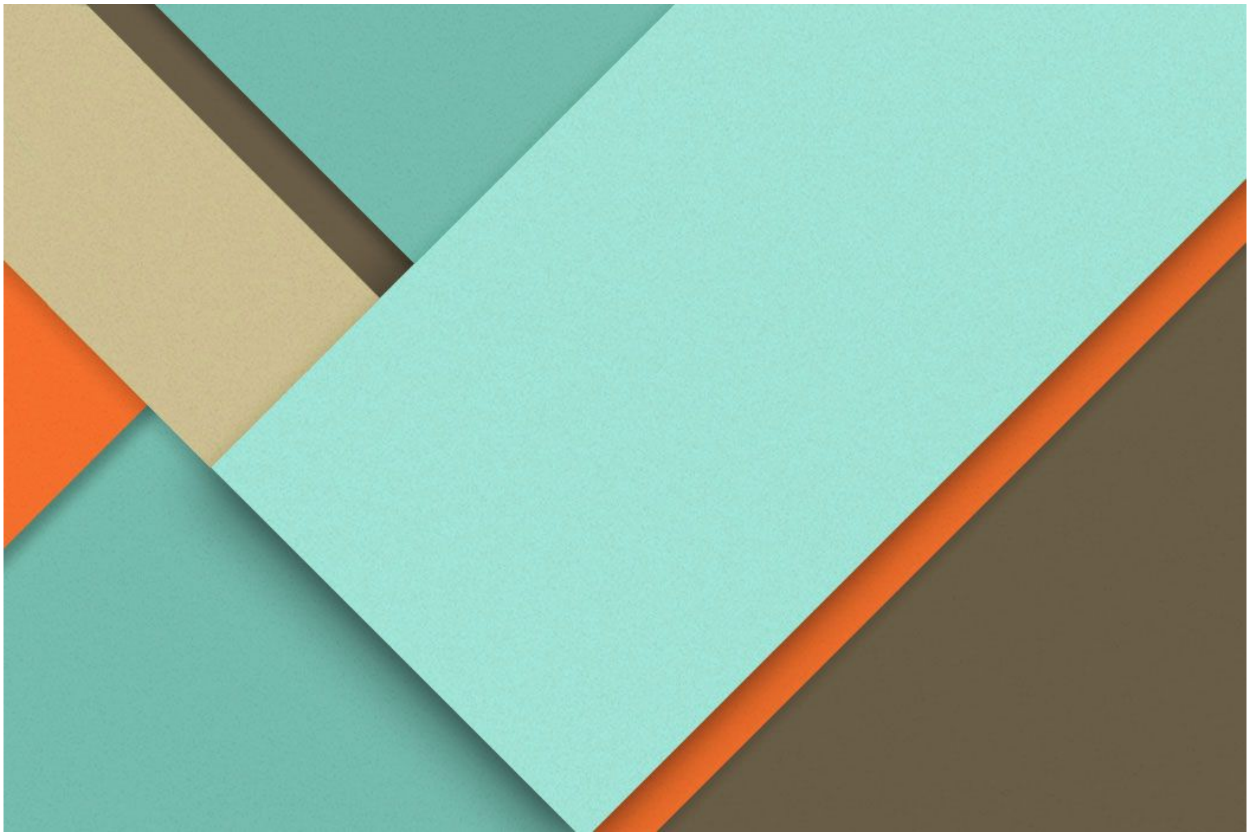


Twitter Sentiment Analysis

Fall 2017

Ashour Dankha

Viren Mody

Final Project

CS 583 Data & Text Mining

Professor Liu, Bing

Table of Contents

Abstract

Sentiment analysis is a popular text analysis that typically focuses on determining whether a piece of writing is negative, neutral, or positive. Sentiment analysis essentially determines the opinion or attitude of what the message is. In this project, we propose a variety of popular machine learning techniques in order to train our program to be able to classify tweets. Our performance measurement consisted of using F1-Scores of positive, neutral, and negative classifications of tweets during the Obama vs Romney 2012 elections.

Introduction

Purpose

The purpose of this graduate data mining research project, given about 7,000 tweets each for both Obama and Romney, is to research and experiment on a variety of methods to perform sentiment analysis on the data to build the best classification model. We are only considering positive, negative and neutral tweet classifications. The criteria is to build the classification model that produces the highest average F1-score primarily for positive and negative tweets.

Programming Description

The program was coded in Python3.x and used the scikit-learn Machine Learning Library for Python. The dataset of raw tweets was used for both training and testing the classifier from the Obama vs Romney 2012 Presidential Elections. The training data we received included classifications of, positive, negative, neutral and mixed, which was used to experiment with a variety of supervised learning classifiers provided by the scikit library. We also used a library for natural language processing. The library was called nltk and was used for a variety of reasons that will later be discussed in the report.

Technique: 3 Stages

Stage 1: Preprocessing

In order for us to be able to train our program to learn from the data provided, we needed to alter the data to make sure we are using only the important aspects of the tweet. An important step for training models is the pre-processing step. Since we were given raw tweets, there was a lot of unnecessary information that needed to be removed.

Here are the steps (in order) of what we did to clean up our data.

a) **Shuffling**: Since our data was sorted by date and time, we learned that some models classified tweets based on what they *currently* know to fit any future tweets or data. This was problematic as we observed that there was a lower performance on our F-scores. To handle this, we shuffled our data to have more of a mixed and unbiased set of data when fitting each data point / tweet.

b) **Correct Classifications**: Since our focus was to target positive, neutral, and negative tweets, we removed any tweets that did not have these classifications.

c) **Regular Expressions**: We used a commonly known technique called Regular Expressions (regex) where we removed links, tags, and non-alphanumeric characters. Links really serve no purpose as we did not use any form of web-crawling to see what the website was about.

d) **Hashtags**: Hashtags are commonly used with tweets. It's a way for the person tweeting tag a common word or sentence to the topic discussed in the tweet. However, the underlying issue becomes that hashtags have words that are not separated through spaces. In order for us to handle this, we used a library called nltk which is a natural language processing library that had a pre-defined dictionary. The dictionary allowed us to determine if the characters we were extracting from the hashtag made up a word in the dictionary.

e) **Stop Words**: Stop words are essentially words that are very common in text. Some words include, *as*, *the*, *is*, *and*, and etc. Since these are common words in tweets (and in our English vocabulary), we did not need to use stop-words for our count vectorization because it would not help our results since they are common in every tweet (regardless of the tweets classification).

f) **Word Stemmers**: We used both Porter Stemmer and the Snowball Stemmer to strip words down to their roots. This pre-processing step helps in developing a better classification model by associating a word to its root word so that when used during the vectorization process, all "similar" words will be contained in the same bucket. For example, the derived words "loves", "loved", "loving", etc. can all be reduced to "lov" and therefore all occurrences of "lov" in tweets will be classified as positive.

Once the pre-processing step is complete (i.e. we have cleaned up the tweets), we were then able to use the tweets to fit into our classifiers and train on the data.

Stage 2: Feature Extraction

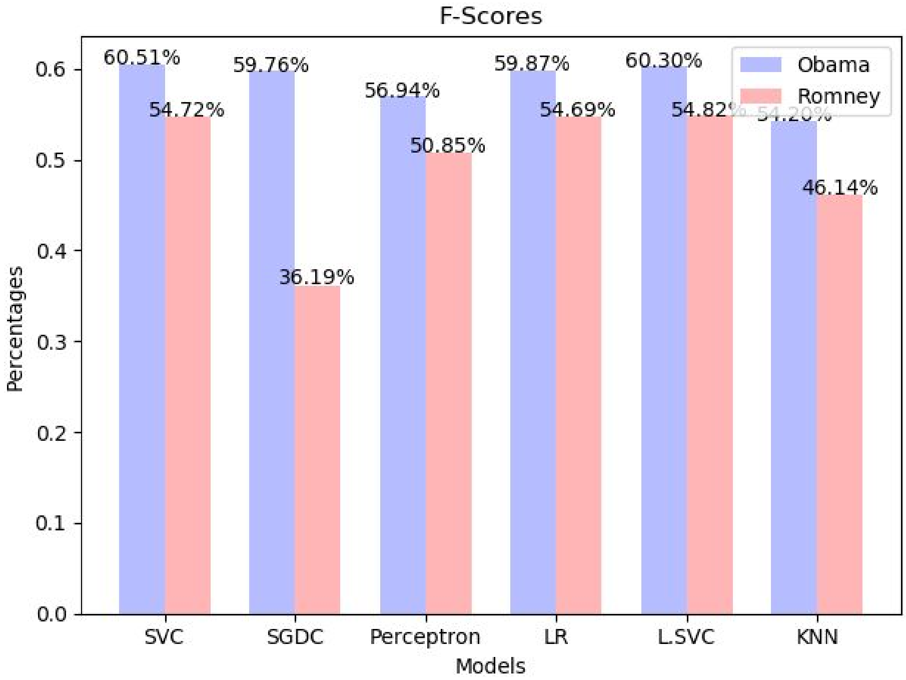
After preprocessing, in feature extraction, we transform our tweets into a numerical feature vector or a “bag of words” using scikit-learn’s CountVectorizer. With CountVectorization, we needed to tune specific parameters to get the best results. In particular, we set the n-gram range to unigrams and bigrams and the max number of features to 1200. In addition, we used scikit-learn’s TfidfTransformer which allowed us to transform our tweets into a “bag of words” and count the occurrence of them. However, TF-IDF (Term Frequency times Inverse Document Frequency) is a special case of feature transformation as it puts more weight on less frequent words and less weight on more frequent words in the text. This allowed us to get more of an impact on specific words that were used on specific classifications (positive, neutral, and negative) when training our models.

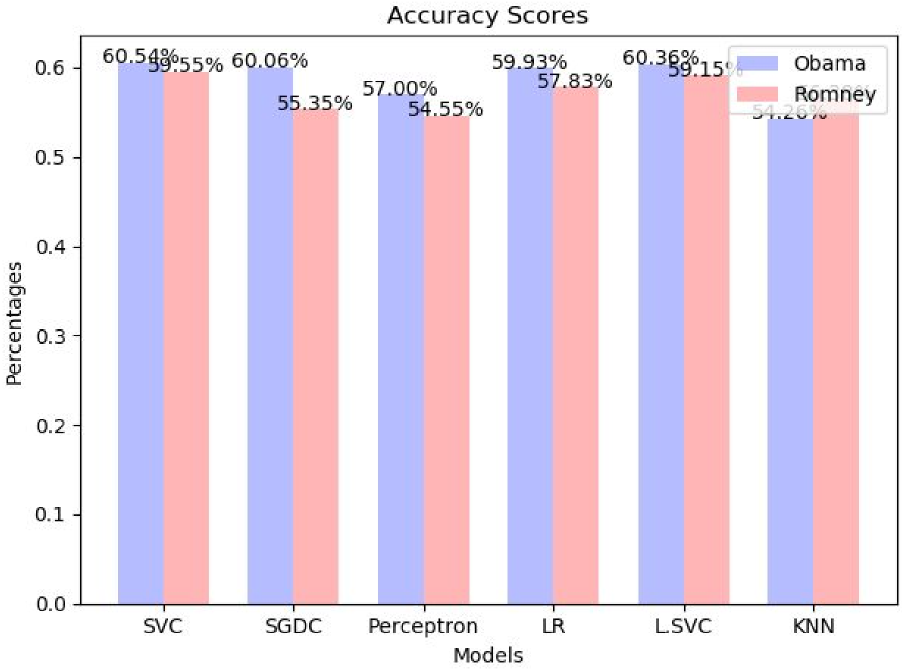
Stage 3: Classification Model (Fitting to model and tuning parameters)

1) **Classification Methods**: After feature extraction, since we were given tweets with their classifications, we tested a variety of supervised learning classification methods. The models below is what we used.

   a) SVC: Support Vector Classifier (SVM: Support Vector Machine)
   b) SGDC: Stochastic Gradient Descent Classifier
   c) Linear Model Perceptron
   d) NC: Nearest Centroid
   e) LR: Logistic Regression
   f) kNN: k-Nearest Neighbors Classifier (In our example, we used k = 50)
   g) LSVC: Linear Support Vector Classifier
   h) DTree: Decision Tree Classifier
   i) RF: Random Forest Classifier
   j) Ada: Ada Boost Classifier
   k) NB: Naive Bayesian Classifier

Originally, we tested each classifier using their default parameters, but to find the parameters that produced the best results, we used scikit-learn’s GridSearchCV. GridSearchCV performs exhaustive testing by trying each classifier with every possible combination of the given set of parameters to find the best set of parameters. Our results from using 10-fold cross validation on the given training data showed that SVC (Support Vector Classifier) produced the best F1-score. Only a subset of the above set of classifiers results are shown below:





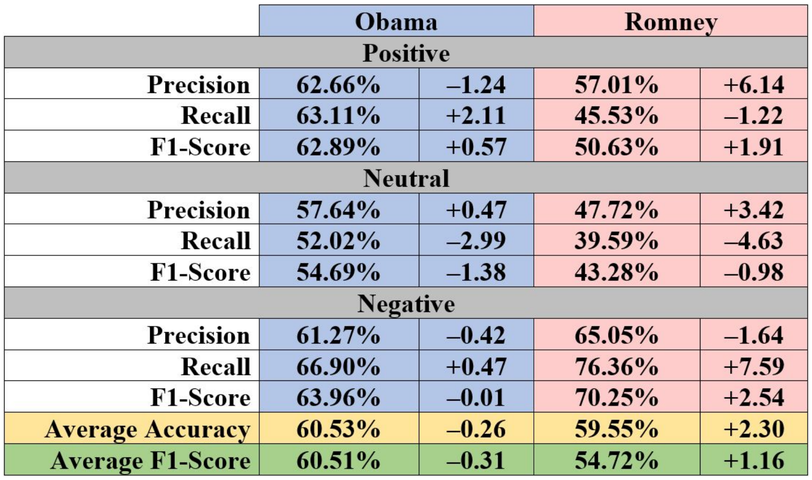
For more results on precision and recall visit the following links:

- Precision: https://i.imgur.com/3EVoQTb.png
- Recall: https://i.imgur.com/HXY6ID8.png

Evaluation

I. 10-Fold Cross Validation

Using 10-fold cross validation, Support Vector Machine produced the best the results of all classifiers. After the first presentation, we further tuned our parameters and included hashtags separated into words. The following are the results (Note: table includes the change in results from the first presentation):



	Obama		Romney	
Positive				
Precision	62.66%	–1.24	57.01%	+6.14
Recall	63.11%	+2.11	45.53%	–1.22
F1-Score	62.89%	+0.57	50.63%	+1.91
Neutral				
Precision	57.64%	+0.47	47.72%	+3.42
Recall	52.02%	–2.99	39.59%	–4.63
F1-Score	54.69%	–1.38	43.28%	–0.98
Negative				
Precision	61.27%	–0.42	65.05%	–1.64
Recall	66.90%	+0.47	76.36%	+7.59
F1-Score	63.96%	–0.01	70.25%	+2.54
Average Accuracy	60.53%	–0.26	59.55%	+2.30
Average F1-Score	60.51%	–0.31	54.72%	+1.16

II. Test Data

For the final evaluation, we used our support vector classifier against the test data we were given to predict the class of each tweet. However, we received test data without classifications, therefore the results are pending.

Conclusion

For this data mining project, we implemented a range of supervised learning techniques since we were given tweets with their classifications. Based on our results, Support Vector Machine was the best supervised learning classification model for predicting Obama's and Romney's tweets during the 2012 Presidential Election. Examining results throughout the process has shown us that every step has proven vital to performance success, from data preprocessing to tuning parameters for different classification models, and of course the different classifiers themselves.

Future work on Sentiment Analysis includes researching more on deep-learning and neural network techniques, using Amazon clusters to tune parameters, and learning more on what the best model and approach (i.e. data transformation, feature extractions, categorial information) would be when dealing with political tweets.

References

Supervised Learning. (n.d.). Retrieved from http://scikit-learn.org/stable/supervised_learning.html#supervised-learning

Web Data Mining - Exploring Hyperlinks, Contents and Usage Data, By Bing Liu, Second Edition, Springer, July 2011, ISBN 978-3-642-19459-7