Sentiment Analysis of Twitter Data
Final Project Report

Course: Information Retrieval and Extraction

Team Number 10

Name: Nurendra Choudhary

Roll no: 201325186

Name: P Yaswanth Satya Vital Varma

Roll no: 201301064

INTRODUCTION:

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgment or evaluation, affective state (that is to say, the emotional state of the author when writing), or the intended emotional communication (that is to say, the emotional effect the author wishes to have on the reader).

AIM OF THE PROJECT:

The purpose of this project is to build an algorithm that can accurately classify Twitter messages as positive or negative, with respect to a query term.

Our hypothesis is that we can obtain high accuracy on classifying sentiment in Twitter messages using machine learning techniques.

DATASET:

1600000 sentences annotated as positive, negative. http://help.sentiment140.com/for-students/

Sentiment140 Dataset

Data is made up of text sentences annotated with two classes: Positive or Negative

PROCEDURE:

Step 1: Preprocessing (Get raw data in suitable format)

1. Case Folding of the Data (Turning everything to lowercase)
2. Punctuation Removal from the data
3. Common Abbreviations and Acronyms expanded.
4. Hash-Tag removal.

Step 2: Training Distributed Semantic Representation (Word2Vec Model)

1. Use a Python Module called gensim.models.word2vec for this.
2. Train a model using only the sentences (after preprocessing) from the corpus.
3. This generates vectors for all the words in the corpus.
4. This model can now be used to get vectors for the words.
5. For unknown words, we use the vectors of words with frequency one.

Step 3: Make Vectors according to Language Model

1. For Unigram, take one vector for one word.
2. For Bigram, take one vector for two words.

Step 4: Training For Machine Learning Scores

1. Use training data to make the models for:
    a. Support Vector Machines - Scikit Learn Python
    b. Multi Layer Perceptron Neural Network - Scikit Learn Python
    c. Naive Bayes Classifier - Scikit Learn Python
    d. Decision Tree Classifier - Scikit Learn Python
    e. Random Forest Classifier - Scikit Learn Python
    f. Logistic Regression Classifier - Scikit Learn Python
    g. Recurrent Neural Networks - PyBrain module Python
2. Classify the test data using the models.
3. Compare the output results with the actual classes to get the accuracy.

4. Divide data into 4:1 ratio and apply 5-fold Cross-validation to get accuracies.

Note: Based on the results, we chose Hybrid of Unigram and Bigram with Random Forest Classifier to be the part of our system as they gave the best results.

Step 5: Emoticon Scoring

1. Search for Emoticons in the given text using RegEx or find.
2. Use a dictionary to score the emoticons.
3. Use this emoticon score in the model.

Step 6: Lexical Scoring

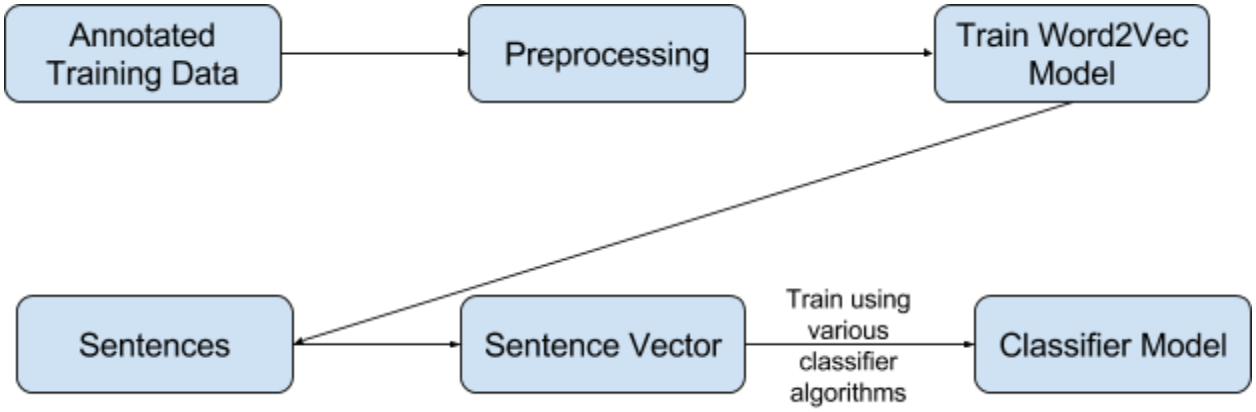
1. Get the text.
2. Lemmatize the text.
3. Score the Lemmatized text using dictionaries.
4. The Score will be used in the final system.
5. This will be given more weightage as this is more definite.

RESULTS:

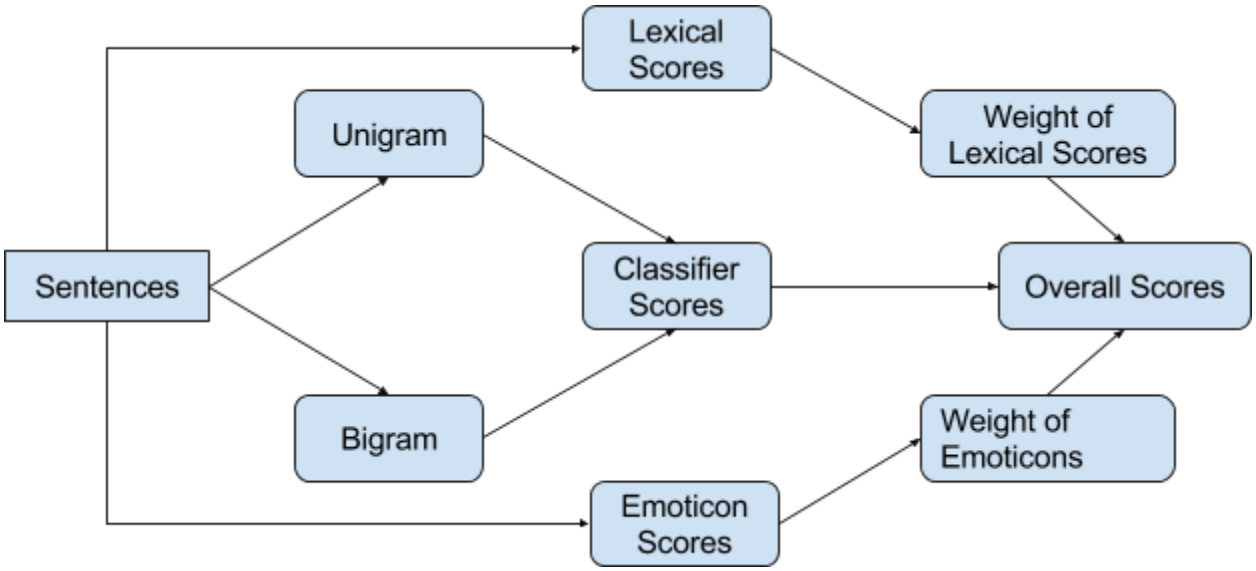
	Unigram	Bigram	Unigram+Bigram
Support Vector Machines	71.1%	-NA-	74.3%
Naive Bayes Classifier	64.2%	62.8%	65.0%
Logistic Regression	67.4%	72.1%	71.6%
Decision Trees	60.4%	60.0%	61.5%
Random Forest	67.1%	70.%	71.3%
Multi-Perceptron Neural Network	68.6%	72.7%	74%
Recurrent Neural Networks	69.1%	70.4%	71.5%

Work Flow Diagram:

Training Classifier and Word2Vec Model



The Overall Scoring Process Goes Like This



CHALLENGES:

- ➔ **Randomness in Data:** Twitter is written by Users, hence it is not very formal.
- ➔ **Emoticons:** Lots of types of emoticons with new ones coming very frequently.
- ➔ **Abbreviations:** Users use a lot of abbreviation slangs like AFAIK, GN, etc. Capturing all of them is difficult.
- ➔ **Grapheme Stretching:** Emotions expressed through stretching of normal words. Like, Please -> Pleaaaaaseeeeee
- ➔ **Reversing Words:** Some words completely reverse sentiment of another word. E.g: not good == opposite(good)
- ➔ **Technical Challenges:** Classifiers take a lot of time to train, hence silly mistakes cost a lot of time.

FUTURE IMPROVEMENTS:

- ➢ Handle Grapheme Stretching
- ➢ Handle authenticity of Data and Users
- ➢ Handle Sarcasm and Humor

SUPPORT:

- ❑ Github Link of the project: https://github.com/Akirato/Twitter-Sentiment-Analysis-Tool
- ❑ Webpage of the project: http://akirato.github.io/Twitter-Sentiment-Analysis-Tool/
- ❑ YouTube Link of the Demo video: https://youtu.be/VuR16P87yPE
- ❑ For any more doubts or details, raise an issue on Github, it will be sorted out.