Individual Report on Project (GROUP NUMBER): Sarcasm Detection (Group Number: 45)
Name: Sai Geetha Kandepalli Cherukuru
Email address: kandepal@usc.edu
Rest of my group: Farhan Mazhar(fmazhar@usc.edu) and Suraj Rajasekhar(rajasekh@usc.edu)
URL of GitHub repository: https://github.com/surajr/SarcasmDetection

1. Project Overview

Sarcasm is a very cultural sentiment. Merriam-Webster defines sarcasm as *the use of words that mean the opposite of what you really want to say especially in order to insult someone, to show irritation, or to be funny*. Sarcasm Detection is something that would be a difficult task for a computer to understand. The ambiguous nature of sarcasm makes it hard, even for humans to decide if the utterance is sarcastic or not. Sarcasm Detection is a sub task in opinion mining. Recognition of Sarcasm can benefit many sentiment analysis NLP applications such as review ranking systems, dialogue systems and review summarization.

Important steps followed in Text Categorization are as follows: Preprocessing, Feature Engineering, Feature Selection, Classification and Model Evaluation. The algorithm relies on an assumption that a positive situation is followed by a negative one in the sarcastic tweet. i.e. [positive verb phrase] + [negative verb phrase].

Since detecting sarcasm is a complex problem and there is no state of art approach to evaluate and compare the metrics. Machine Learning algorithms such as Adaboosting, Decision Trees, Gaussian Naïve Bayes, Gradient Boosting, Logistic Regression and Random Forest are used for this binary classification problem. The performance of the model is measured by testing the model on unseen test dataset. Grouped the features together and calculated the respective F1 score.

Features	**F1 Score**
Lexical Features + POS bigrams	0.5233
Sentimental Features + Topic Modeler + Capitalization	0.6689
Sentiment + POS bigrams + Topic Modeler + Capitalization	0.7152

2. My Primary Responsibility

The list of Responsibilities, Methodologies and Evaluation metrics are as follows

I. Data Extraction

Extracting the right dataset is very crucial in any classification problem. For this project we extracted the dataset from Twitter. Twitter is a social networking platform, where all registered users can broadcast short messages called tweets. Twitter offers enormous amount of sarcastic data. And it is easy to retrieve tweets using Twitter API.

Retrieved 1000 tweets with tags #sarcastic and #sarcasm using tweepy and manually classified the tweets as Sarcastic tweets and Non-sarcastic tweets. Sarcastic tweets are the ones with #sarcastic and #sarcasm. Non-sarcastic tweets are the ones with #thrilled, #happy, #sad, #ElectionDay, #awesome and so on.

Overall the dataset includes about 1024 tweets categorized as Sarcastic and 960 tweets as Non-sarcastic. Data was divided into 75% training and 25% testing.

II. Annotation

The entire dataset was manually annotated by all the team members and differentiated between Sarcastic and Non-sarcastic tweets. Tweets with #sarcasm and #sarcastic are ignored if they were not sarcastic. Annotation was as follows Sarcastic: 1 and Non-sarcastic: 0.

We obtained a score of 0.6433 by evaluating the annotation using inter annotator Reliability–Fleiss Kappa.

III. Preprocessing

Preprocessing is an important task and the checks included in this project are as follows

a. Minimum length of each tweet has to be 3
b. Eliminated all the emoticons (☺,☹,:/,:*,:P,:D,(:) to an empty character
c. Ignoring the tweets that are in non ASCII character
d. Tweets were shuffled
e. Removal of #sarcasm and #sarcastic tags

IV. Feature Engineering

a. Spell Check

Tweets usually contain abbreviations or misspelled words. Thus checked and corrected all misspelled words using TextBlob library.

b. Lexical Features

Evaluated the lexical features of 2000 tweets. Checked for special characters like question marks or exclamatory marks and calculated count of each of them. Checked the dataset for any hashtags other than #sarcasm and #sarcastic or any other http tags present in the dataset. And checked for any capitalized word in tweets.

c. Tokenization

The task of chopping a document unit into pieces called tokens and possibly throwing away certain characters is called as Tokenization. Every token is an instance of sequence of characters in a particular document that are grouped together into a single unit.

Chopped tweets into useful semantic units called tokens using word_tokenize method from NLTK library.

d. Stemming and Lemmatization

Stemming is the process of transforming the words to their stem or root form. Lemmatization reduces the inflected words to their base word called Lemma with the help of vocabulary and morphological analysis.

Stemming algorithms such as Porter's Algorithm uses suffix stripping methods. 'sses' to 'ss', 'tional' to 'tion', 'ies' to 'i', 'ss' to 'ss' and 's' to null. For an example when Stemmer encounters words such as fisher, fishing and fished, it reduces these words to it's base word fish.

The goal of both Stemming and Lemmatization is reducing inflected words to their common base form as it would be useful to search for one of these words to return any document that contain the inflected form of word.

e. Part-of-speech tagging

Part-of-speech tagging (POS tagging) also called grammatical tagging is the process of assigning part of speech tag to every token in the dataset. POS taggers helps in distinguishing homonyms.

TextBlob, CMU tweet NLP and NLTK POS taggers are used for POS tagging of token in the corpus.

	tweets	label
0	I love working midnights tweet	1
1	I hate when I buy a bag of air and there's chi...	1
2	my grandad always sounds so ill when i speak t...	0
3	I realize I'm annoying to everyone, so I won't...	0
4	I love when I find these dudes on vine!! #Foll...	1

```
featureset = []
```

```
import re

for i in range(0,df.size):
    temp = str(df["tweets"][i])
    temp = re.sub(r'[^\x00-\x7F]+','',temp)
    featureset.append((get_features(temp,topic_mod), df["label"][i]))
```

```
done
[('I', 'PRP'), ('love', 'VBP'), ('working', 'VBG'), ('midnights', 'NNS'), ('tweet', 'NN')]
[('I', 'PRP'), ('love', 'VBP')]
[('working', 'VBG'), ('midnights', 'NNS'), ('tweet', 'NN')]
1
```

f. N-grams

Tf-Idf (Term Frequency-Inverse Document Frequency) weighting for unigrams, bigrams and trigrams. Bigrams gave the best score with Random Forest classifier. Most important repeated words were Great, fun, yay, thanks, election.

V. Feature Selection

Feature Selection is an important process, as it is helpful in reducing over fitting and enhances the accuracy.

Extra-Trees (Extremely randomized trees) are a class of ensemble methods used to select important features.

VI. Machine Learning Techniques

a. Gaussian Naïve Bayes

Naïve Bayes Classification is a supervised machine learning technique that assumes for a given class features are independent. Although assumption of independence is poor, Naïve Bayes works well. In this project, a Naïve Bayes model is created from the training data and using the parameters of the model, tweets in development data were classified as Sarcastic or Non-sarcastic. Toolkit used was Scikit.

b. Gradient Boosting

Gradient Boosting is one of the most powerful machine learning technique for classification problems which provides prediction model and involves three elements such as Loss Function, Weak Learner to make the predictions and an Additive Model for minimizing the loss function by adding weak learners.

The classifiers were used to classify the tweets in development data with the help of parameters in the model. The accuracy score for Gradient Boosting and Gaussian Naïve Bayes machine learning techniques is shown in the below screenshot

```
GradientBoosting : 0.769423558897
GNB : 0.679197994987
```

VII. Evaluation Results

a. Findings

sargasm @Sargazzm · Nov 18

When people see me eating and ask me are you eating ? No no i'm trying to choke myself to death **#sarcastic**

1 8

```
[{'Blob Polarity': 0.1,
  'Blob Subjectivity': 0.8,
  'Capitalization': 0,
  'Negative Sentiment': 0.89136904761904767,
  'POS_1': 2.0,
  'POS_2': 1.0,
  'POS_3': 8.0,
  'POS_4': 0.0,
  'Positive Sentiment': 0.19419642857142858,
  'Topic :': 0.5025000000000025,
  'first half Blob Polarity': 0.0,
  'first half Blob Subjectivity': 0.0,
  'first half sentiment': 0.059523809523809527,
  'negative Sentiment first half': 0.040922619047619048,
  'negative Sentiment second half': 0.8504464285714286,
  'positive Sentiment first half': 0.10044642857142858,
  'positive Sentiment second half': 0.09375,
  'second half Blob Polarity': 0.1,
  'second half Blob Subjectivity': 0.8,
  'second half sentiment': -0.7566964285714286,
  'sentiment': -0.69717261904761907}]
```

```
res= clf.predict(test_result)
```

Sarcastic


```
[{'Blob Polarity': -0.13333333333333333,
  'Blob Subjectivity': 0.3333333333333333,
  'Capitalization': 0,
  'Negative Sentiment': 0.125,
  'POS_1': 3.0,
  'POS_2': 2.0,
  'POS_3': 3.0,
  'POS_4': 1.0,
  'Positive Sentiment': 0.375,
  'Topic :': 0.50250000000000306,
  'first half Blob Polarity': 0.0,
  'first half Blob Subjectivity': 0.0,
  'first half sentiment': 0.0,
  'negative Sentiment first half': 0.0,
  'negative Sentiment second half': 0.125,
  'positive Sentiment first half': 0.0,
  'positive Sentiment second half': 0.375,
  'second half Blob Polarity': -0.13333333333333333,
  'second half Blob Subjectivity': 0.3333333333333333,
  'second half sentiment': 0.25,
  'sentiment': 0.25}]
```

```
res= clf.predict(test_result)
```

```
Sarcastic
```

b. Precision, Recall and F1 score

Gaussian Naïve Bayes			
Precision	**Recall**	**F1 score**	**Support**
0.74	0.67	0.70	205
0.68	0.75	0.71	194

Gradient Boosting		
Precision	**Recall**	**F1 score**
0.75	0.81	0.78
0.78	0.71	0.75

3. Other Project Work

a. As a team worked on creating the slides for presentation

4. Online Resources

a. http://cs229.stanford.edu/proj2015/044_report.pdf
b. An Introduction to Information Retrieval textbook – 6th Edition
c. Definition from Merriam-Webster dictionary

5. References

a. TextBlob – POS tagging, Spell Check, Sentiment Analysis
https://textblob.readthedocs.io/en/dev/index.html
b. Scikit learn – TFIDF, ML algorithms such as clustering, SVD, GridSearch and CV
http://scikitlearn.org/stable
c. TweetNLP – Unique POS tagger for tweets
http://www.cs.cmu.edu/~ark/TweetNLP/
d. GenSim – Phrase Extraction, Word2Vec, Topic Modelling
https://pypi.python.org/pypi/gensim
e. Contextualized Sarcasm Detection on Twitter
http://homes.cs.washington.edu/~nasmith/papers/bamman+smith.icwsm15.pdf
f. Automatic Sarcasm Detection: A survey, Cornell library, 20 Sept 2016
https://arxiv.org/abs/1602.03426