OVERVIEW

Objective

1. To evaluate the person’s opinion in certain cases.
2. To teach machine to analyze various grammatical nuances.
3. To implement an algorithm for automatic classification of text into positive, negative or neutral.
4. To make a rigid model with advanced tools like NLTK and ML library.

Literature Review

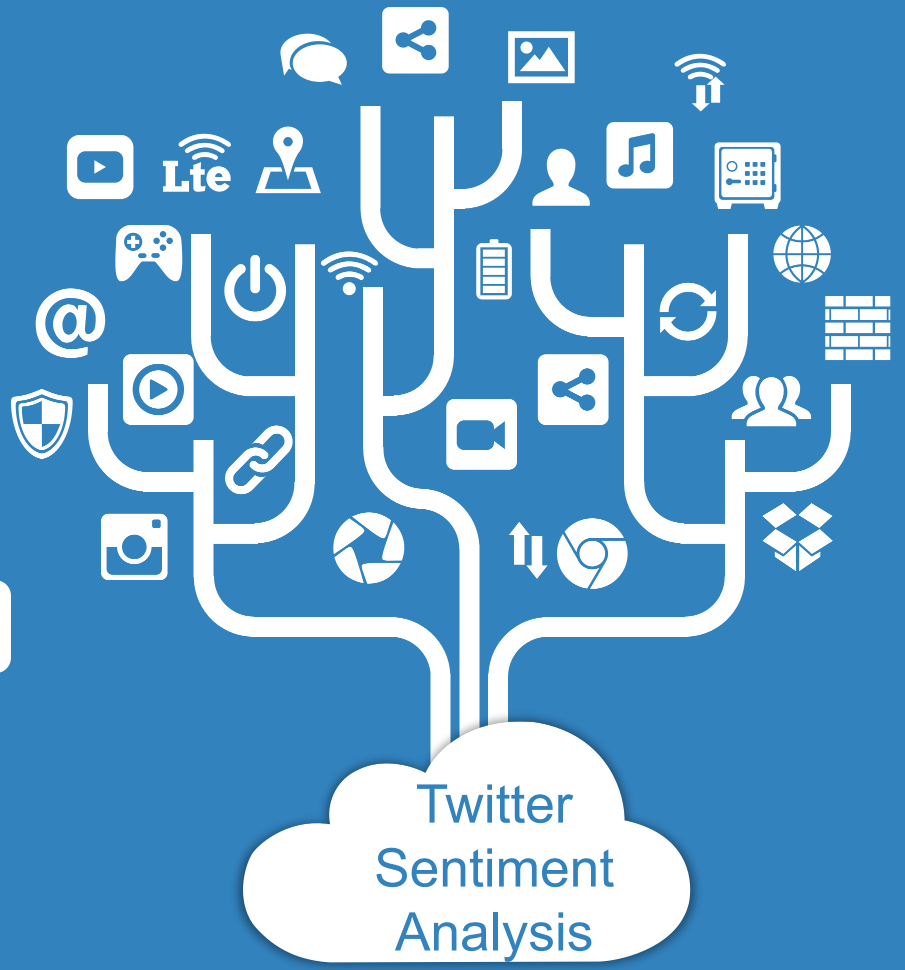
- Sentiment Analysis of in the domain of micro-blogging is relatively a new research topic.
- Best results in sentiment classification use supervised learning techniques such as Naive Bayes and SVM,but manual labeling required for supervised learning is very expensive.
- Some work has been done on unsupervised and semi-supervised approaches,and there is a lot of room for improvement.

Motivation

- Deep Impact of Social Media
- Sentiment, an Important aspect of Social Media
- Empirically studying properties of social interactions.
- Accurate tools for extracting semantic information.
- NLTK and ML powerful tools for feature extraction.

Scope and Application

- Social Networking sites for determining opinions of people on products or brands.
- Review on currently published movies, songs and products graphically.
- Business, politics and public actions.



Luoet highlighted the challenges and an efficient techniques to mine opinions from Twitter tweets. Spam and wildly varying language makes opinion retrieval within Twitter challenging task

Kamps used the lexical database WordNet to determine the emotional content of a word along different dimensions. They developed a distance metric on WordNet and determined semantic polarity of adjectives

Methodology and Tools

Preprocessing Tweets

1) Letter Casting
2) Tokenization
3) Noise Removal
4) Stopword Removal

NLTK

1) Stemming
2) Lemmatization
3) Vectorization

Training and Testing

1) Naive Bayes
2) Logistic Regression

Prediction

1) Predicting Tweets into Positive Negative and Neutral through Models.

Tools Used:

1) Python 3 IDLE
2) Jupyter Notebook
3) Numpy Pandas NLTK Sklearn Library
4) Training and Testing dataset extracted from Kraggle

Preprocessing of Data

Letter casing: Converting all letters to either upper case or lower case.

Tokenizing: Turning the tweets into tokens. Tokens are words separated by spaces in a text.

Noise removal: Eliminating unwanted characters, such as HTML tags, punctuation marks, special characters, white spaces etc.

Stopword removal: Some words do not contribute much to the machine learning model, so it's good to remove them.

```python
def preprocess_tweet_text(tweet):
    """
    Function to process the the tweet text and tranform it into format usable by Ma
    """

    # to convert all the characters of the tweet into lower case alphabets
    tweet.lower()

    # Remove urls from the tweets
    tweet = re.sub(r"http\S+|www\S+|https\S+", '', tweet, flags=re.MULTILINE)

    # Remove user related references from the tweets:: '@' and '#'
    tweet = re.sub(r'\@\w+|\#','', tweet)

    # Remove punctuations from the tweets
    tweet = tweet.translate(str.maketrans('', '', string.punctuation))

    # Remove stopwords from the tweets
    tweet_tokens = word_tokenize(tweet)
    filtered_words = [w for w in tweet_tokens if not w in stop_words]
    joined_text = " ".join(filtered_words)

    return joined_text
```

Fig: Preprocessing Snippet from Code

NLTK

Stemming: It may be defined as the process to remove the inflectional forms of a word and bring them to a base form called the stem.
We had used Porter Stemming.

Lemmatization: It is a process wherein the context is used to convert a word to its meaningful base form
We had used WordNet Lemmatizer.

Vectorization: The process of converting words into numbers is called Vectorization.

```
stemmer = PorterStemmer()
stemmed_words = [stemmer.stem(i) for i in processed_text]
```

```
lemmatizer = WordNetLemmatizer()
lemma_words = [lemmatizer.lemmatize(w, pos='a') for w in stemmed_words]
```

```
tf_vector = get_feature_vector(np.array(dataset["tweet_text"]).ravel())
```

Fig: NLTK Snippet from Code

Naive Bayes Classification

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors.

Likelihood

Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)\,P(c)}{P(x)}$$

Posterior Probability

Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Using Naive Bayes Model :

```
NB_model = MultinomialNB()
NB_model.fit(X_train, y_train)
```

```
l2]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Predicting the values and the Accuracy Score

```
y_predict_nb = NB_model.predict(X_test)
print("Accuracy Score for Naive Bayes Model is :: ", accuracy_score(y_test, y_predict_nb))
```

```
Accuracy Score for Naive Bayes Model is ::  0.5918937805730259
```

Classification Report :

```
print("Classification_Report :: \n\n", classification_report(y_test, y_predict_nb))
```

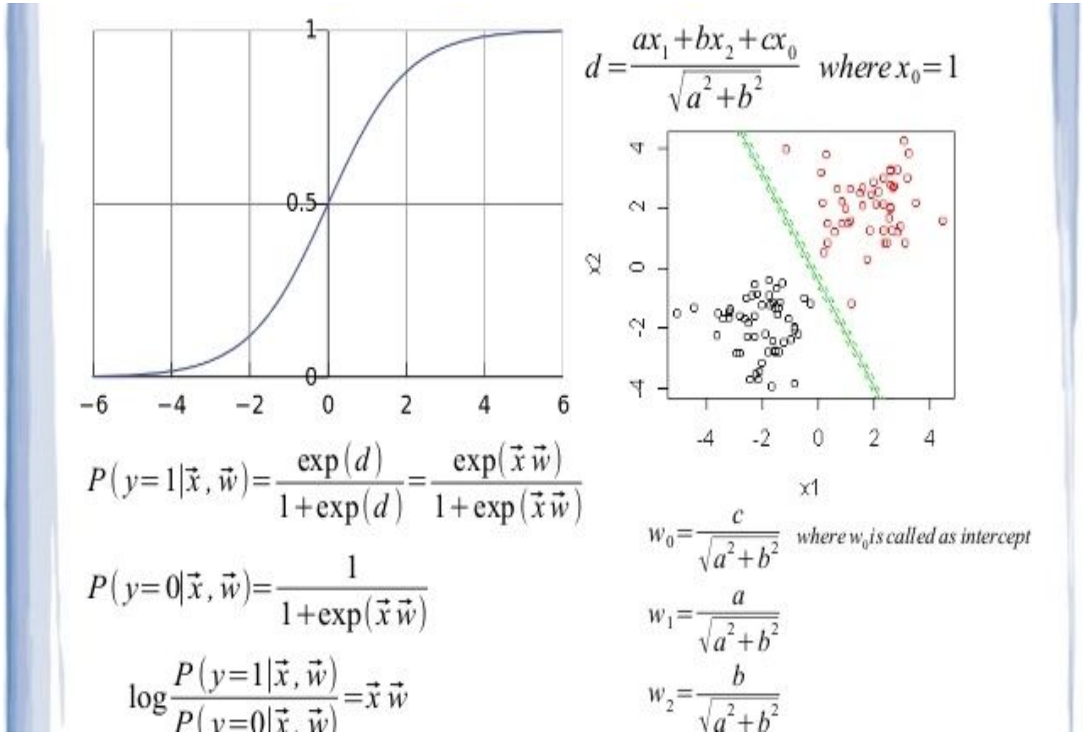
Classification_Report ::

	precision	recall	f1-score	support
negative	1.00	0.00	0.00	676
neutral	0.60	0.60	0.60	1809
positive	0.58	0.81	0.68	1808
accuracy			0.59	4293
macro avg	0.73	0.47	0.43	4293
weighted avg	0.66	0.59	0.54	4293

Fig: Bayes Model Snippet from Code

Logistic Regression Model

In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose.



Using Logistic Regression Model :

```python
# Training Logistics Regression model
LR_model = LogisticRegression(solver='lbfgs')
LR_model.fit(X_train, y_train)
```

Predicting the Values :

```python
y_predict_lr = LR_model.predict(X_test)
print("Accuracy Score for Logistic Regression Model is :: ",accuracy_score(y_test, y_predict_lr))
```

```
Accuracy Score for Logistic Regression Model is ::  0.6457023060796646
```

Classification Report

```python
from sklearn.metrics import classification_report

print("Classification_Report :: \n\n", classification_report(y_test, y_predict_lr))
```

Classification_Report ::

	precision	recall	f1-score	support
negative	0.63	0.24	0.35	676
neutral	0.61	0.74	0.67	1809
positive	0.69	0.70	0.70	1808
accuracy			0.65	4293
macro avg	0.64	0.56	0.57	4293
weighted avg	0.65	0.65	0.63	4293

Fig: Logistic Regression Snippet from Code

Result

It is the result obtained by using the Naive Bayes Algo.

We have predicted the tweets sentiment on the live data.

Using Naive Bayes Model for Prediction ::

In [20]:
```
test_prediction_nb = NB_model.predict(test_feature)

test_prediction_nb
```

Out[20]:
```
array(['neutral', 'positive', 'neutral', ..., 'positive', 'neutral',
       'positive'], dtype='<U8')
```

In [21]:
```
# Creating a Dataframe consising tweets and sentiment in a submission format

submission_result_nb = pd.DataFrame({'tweet_id': test.tweet_id, 'sentiment':test_prediction_nb})
submission_result_nb
```

Out[21]:

	tweet_id	sentiment
0	264238274963451904	neutral
1	218775148495515649	positive
2	258965201766998017	neutral
3	262926411352903682	positive
4	171874368908050432	neutral
...	...	...
5393	210378118865756160	neutral
5394	245177521304399872	positive
5395	259280987089932288	positive
5396	201113950211940352	neutral
5397	237999067286876160	positive

5398 rows × 2 columns

In [22]:
```
# Total number os tweets grouped according sentiment

test_result = submission_result_nb['sentiment'].value_counts()
test_result
```

Out[22]:
```
positive    3177
neutral     2220
negative       1
Name: sentiment, dtype: int64
```

Result

It is the result obtained by using the Logistic Regression Model.

We have predicted the tweets sentiment on the live data.

Using Logistic Regression Model for Prediction ::

In [23]:
```
test_prediction_lr = LR_model.predict(test_feature)

test_prediction_lr
```

Out[23]:
```
array(['neutral', 'positive', 'neutral', ..., 'neutral', 'neutral',
       'positive'], dtype=object)
```

In [24]:
```
# Creating a Dataframe consising tweets and sentiment

submission_result_lr = pd.DataFrame({'tweet_id': test.tweet_id, 'sentiment':test_prediction_nb})
submission_result_lr
```

Out[24]:

	tweet_id	sentiment
0	264238274963451904	neutral
1	218775148495515649	positive
2	258965201766998017	neutral
3	262926411352903682	positive
4	171874368908050432	neutral
...	...	...
5393	210378118865756160	neutral
5394	245177521304399872	positive
5395	259280987089932288	positive
5396	201113950211940352	neutral
5397	237999067286876160	positive

5398 rows × 2 columns

In [25]:
```
# Total number os tweets grouped according sentiment

test_result2 = submission_result_lr['sentiment'].value_counts()
test_result2
```

Out[25]:
```
positive    3177
neutral     2220
negative       1
Name: sentiment, dtype: int64
```

Future Enhancement

Potential improvement can be made on our data collection and analysis method.

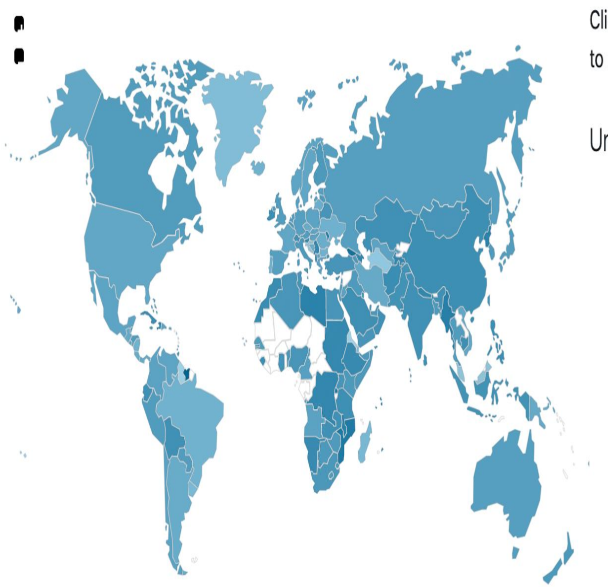
Analyzing sentiments on emo/smiley

Business Model

Future research can be done for more accurate algorithm

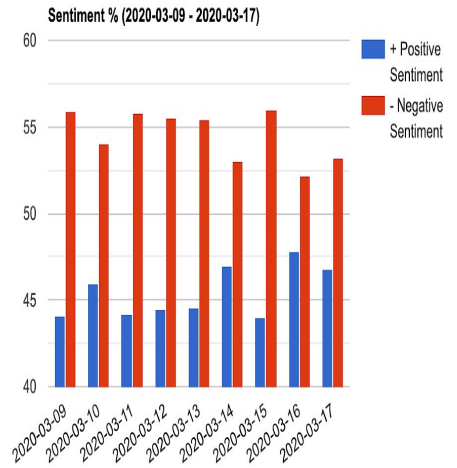
Faster processing system

Academies of Science, Engineering and Medicine.



Click map for sentiment. Map color intensity reflects positive to negative sentiment ratio.

United States of America



Top Countries (most positive on Twitter)

Philippines, Mainland China, Ireland, Saudi Arabia, Tanzania India, Sri Lanka, Switzerland, Ecuador, Pakistan

Top Countries (most negative on Twitter)

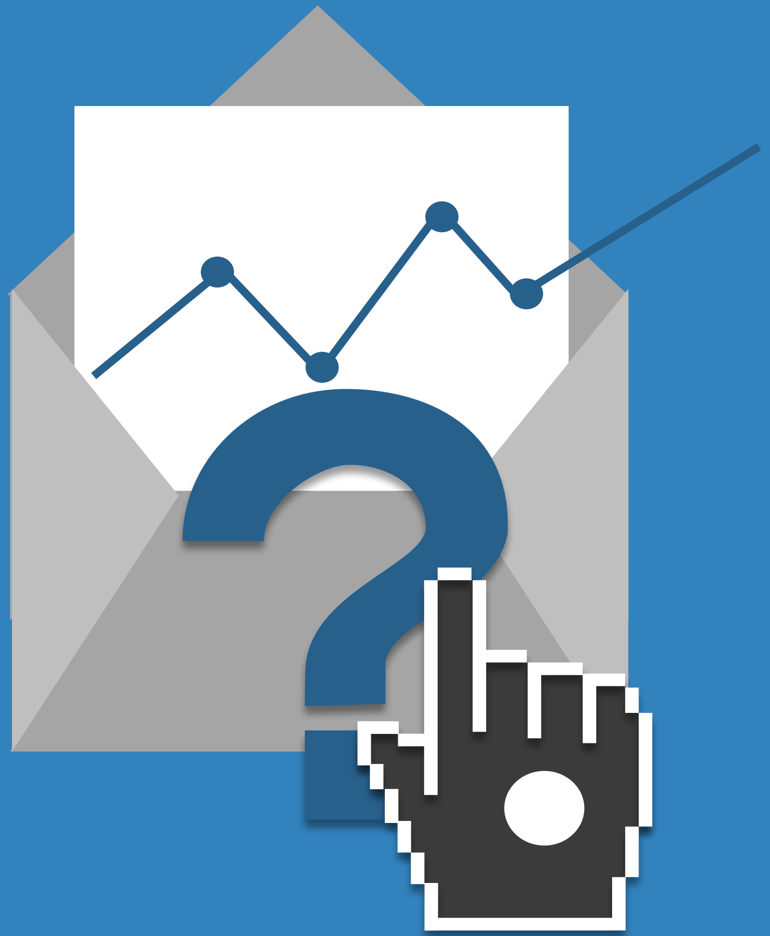
Brazil, Portugal, Hong Kong, Argentina, Singapore, United States of America, France, Sweden, Malaysia, Mexico

Conclusion

We conclude that using different NLTK classifier it is easier to classify tweets and more we improve training data sets more we can get accurate results.

REFERENCES

Twitter Sentiment Analysis



1. Alexandra Pak and Patrick Paroubek.Twitter as a corpus for Sentiment analysis and opinion mining.In Proceeding of international conference on Language Resources and Evaluation(LREC),2010.
2. Luciano Barbosa and Junlan feng. Robust Sentiment Detection on twitter from biased and noisy data.In proceedings of the international conference of computational linguistics(COLING),2010.
3. Peter D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL), 2002.
4. https://arxiv.org/ftp/arxiv/papers/1509/1509.04219.pdf
5. https://www.ijcaonline.org/research/volume139/number11/kharde-2016-ijca-908625.pdf