Report On

Natural Language Processing Based Text Classification

Submitted To:

Richard Philips

Lecturer Dept. of Computer Science Engineering

Submitted By:

Kaniz Fatema(Id # 153402325)

Md. Shahadat Hossain(Id # 153402338)

Md. Shah Alam(Id # 153402336)

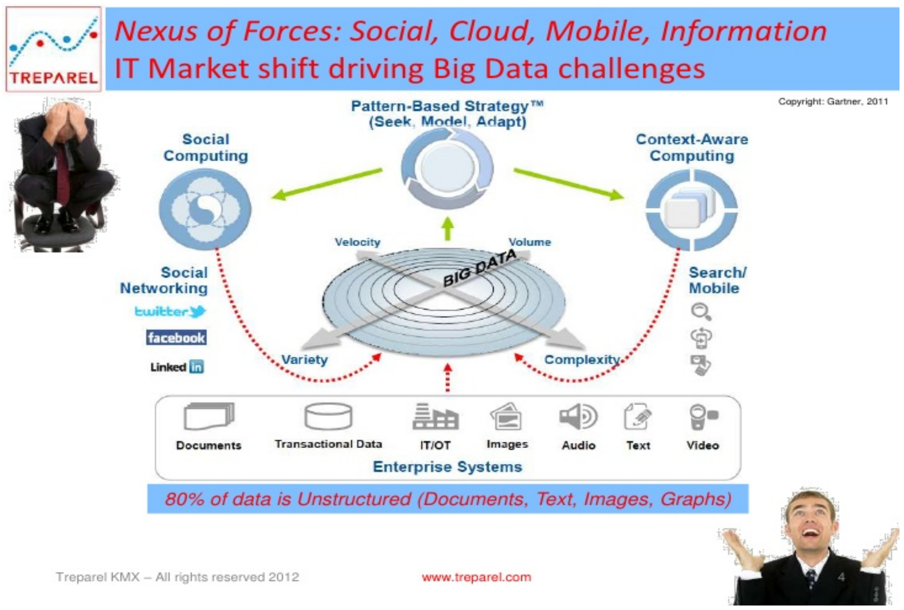
Fatema Akter Mukta(Id # 153402314)

Table of Contents:

Introduction:

Text Classification is a core problem to many applications like spam detection, sentiment analysis or smart replies. It is the process of assigning tags or categories text according to it’s content.

Unstructured data like everywhere: email, chats, web pages, social media, support vector, survey response etc. In this project, we classify text in unstructured form turn into structured form using text classification process. We analyze sentiment analysis for movies datasets which has 2000 negative and positive comment.



.

Background Study:

Definition of Text Classification:

Text classification is a set of documents {d1…………………..di} which classify a set of thematic categories{c1………………..cj} .

It is a pre-defined classes to textual documents. It is also referred as text categorization. A text classification task defined as:

Given a set of documents{d1, d2, d3…………………….dn} and

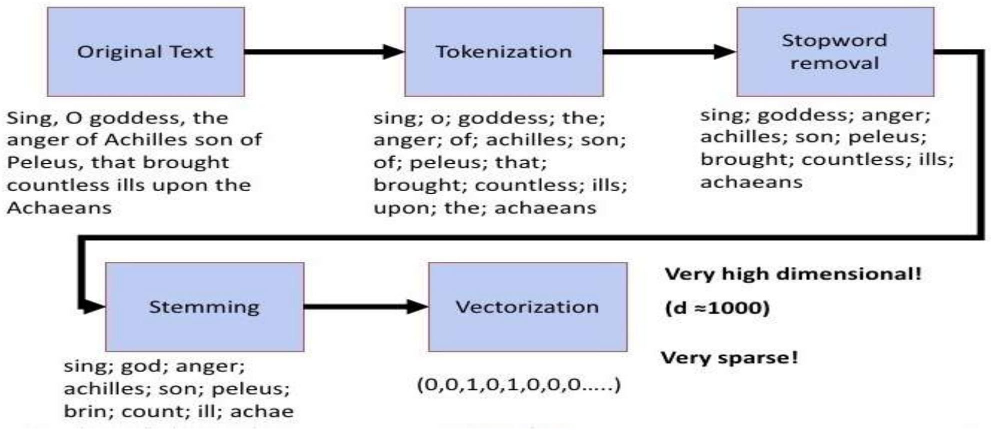
a set of classes{c1, c2, c3………………………………..cm} assign a label form the set to c and document d.

Machine Learning from text to vector:

In this project, text divide into token through tokenization.

Than reduces word like alphabet, small word using stopword

removal. Than Stemming which is reduced infected word in a singular form. Lastly the text converts into binary form through vectorization.

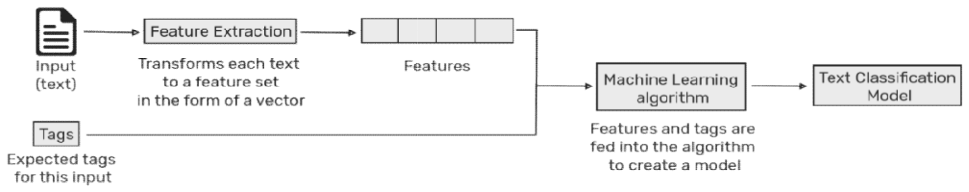

Original Text
Sing, O goddess, the anger of Achilles son of Peleus, that brought countless ills upon the Achaeans
Tokenization
sing; o; goddess; the; anger; of; achilles; son; of; peleus; that; brought; countless; ills; upon; the; achaeans
Stopword removal
sing; goddess; anger; achilles; son; peleus; brought; countless; ills; achaeans
Stemming
sing; god; anger; achilles; son; peleus; brin; count; ill; achae
Vectorization
(0,0,1,0,1,0,0,0.....)
Very high dimensional!
(d ≈1000)
Very sparse!

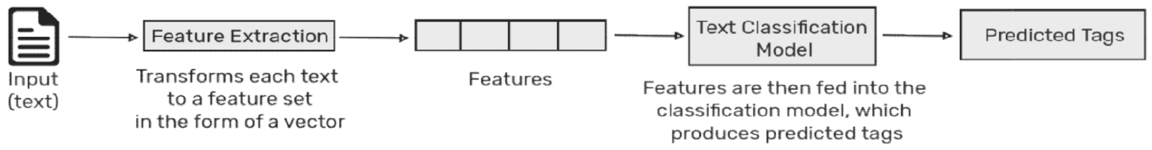
- =

How Text Classification works:

In a machine learning algorithm, it associates different types of text and it's estimation output. It performs several assigning task which includes:

1. Firstly classify a text using feature extraction. It can transform a text into numerical representation.
2. Then using machine learning algorithm which is the main to train data that consists a features sets and tags to produce a classification model.


Input
(text)
Feature Extraction
Transforms each text to a feature set in the form of a vector
Features
Tags
Expected tags for this input
Machine Learning algorithm
Features and tags are fed into the algorithm to create a model
Text Classification Model

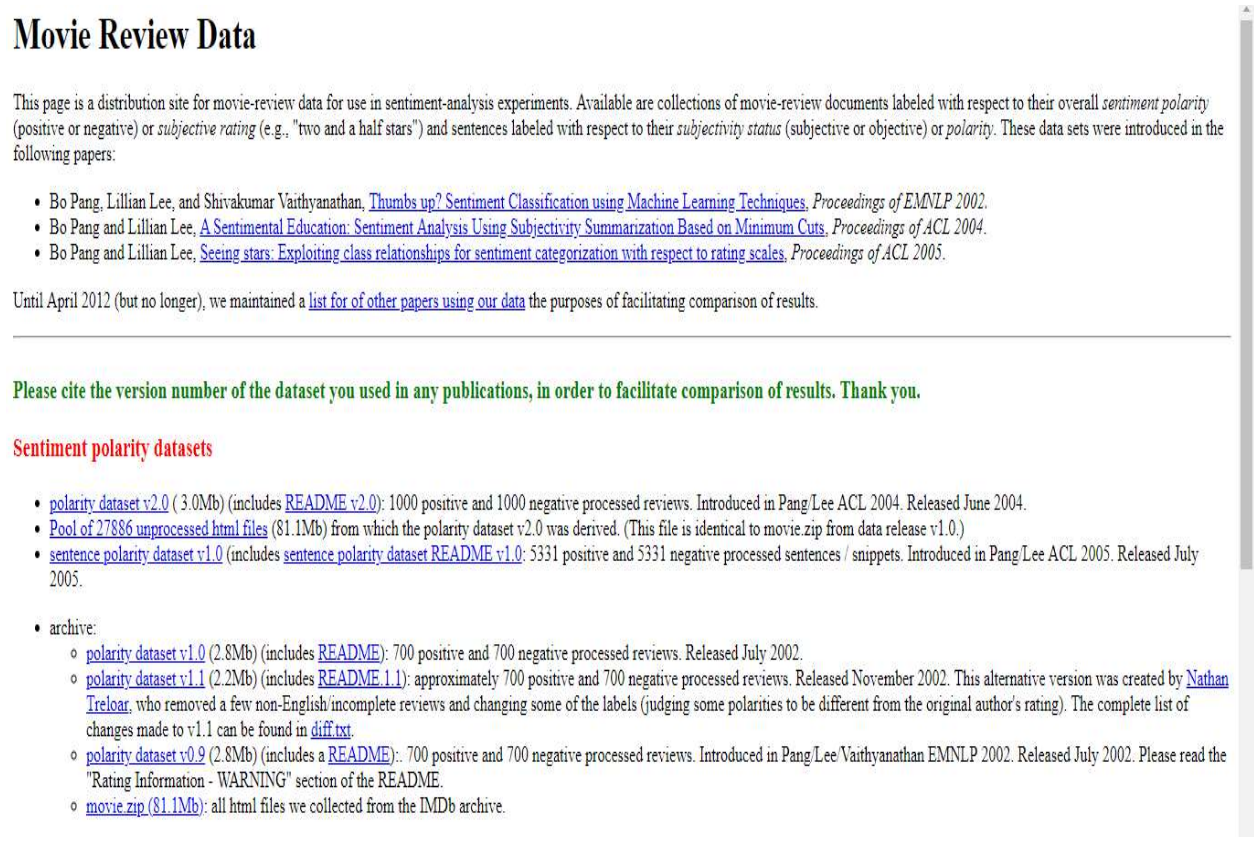

Input
(text)
Feature Extraction
Transforms each text to a feature set in the form of a vector
Features
Text Classification Model
Features are then fed into the classification model, which produces predicted tags
Predicted Tags

Methodology:

Data Collection:

In this project, we classify movies sentiment analysis datasets.

Here is the data collection for movie datasets.



Movie Review Data

This page is a distribution site for movie-review data for use in sentiment-analysis experiments. Available are collections of movie-review documents labeled with respect to their overall *sentiment polarity* (positive or negative) or *subjective rating* (e.g., "two and a half stars") and sentences labeled with respect to their *subjectivity status* (subjective or objective) or *polarity*. These data sets were introduced in the following papers:

- Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, Thumbs up? Sentiment Classification using Machine Learning Techniques, *Proceedings of EMNLP 2002*.
- Bo Pang and Lillian Lee, A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts, *Proceedings of ACL 2004*.
- Bo Pang and Lillian Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, *Proceedings of ACL 2005*.

Until April 2012 (but no longer), we maintained a list for of other papers using our data the purposes of facilitating comparison of results.

Please cite the version number of the dataset you used in any publications, in order to facilitate comparison of results. Thank you.

Sentiment polarity datasets

- polarity dataset v2.0 (3.0Mb) (includes README v2.0): 1000 positive and 1000 negative processed reviews. Introduced in Pang/Lee ACL 2004. Released June 2004.
- Pool of 27886 unprocessed html files (81.1Mb) from which the polarity dataset v2.0 was derived. (This file is identical to movie.zip from data release v1.0.)
- sentence polarity dataset v1.0 (includes sentence polarity dataset README v1.0: 5331 positive and 5331 negative processed sentences / snippets. Introduced in Pang/Lee ACL 2005. Released July 2005.

- archive:
  - polarity dataset v1.0 (2.8Mb) (includes README): 700 positive and 700 negative processed reviews. Released July 2002.
  - polarity dataset v1.1 (2.2Mb) (includes README.1.1): approximately 700 positive and 700 negative processed reviews. Released November 2002. This alternative version was created by Nathan Treloar, who removed a few non-English/incomplete reviews and changing some of the labels (judging some polarities to be different from the original author's rating). The complete list of changes made to v1.1 can be found in diff.txt.
  - polarity dataset v0.9 (2.8Mb) (includes a README):. 700 positive and 700 negative processed reviews. Introduced in Pang/Lee/Vaithyanathan EMNLP 2002. Released July 2002. Please read the "Rating Information - WARNING" section of the README.
  - movie.zip (81.1Mb): all html files we collected from the IMDb archive.

Logistic Regression Algorithm:

Here conducting text classification project using logistic regression algorithm which is the formula of linear regression and probability function. This is the logistic regression formula

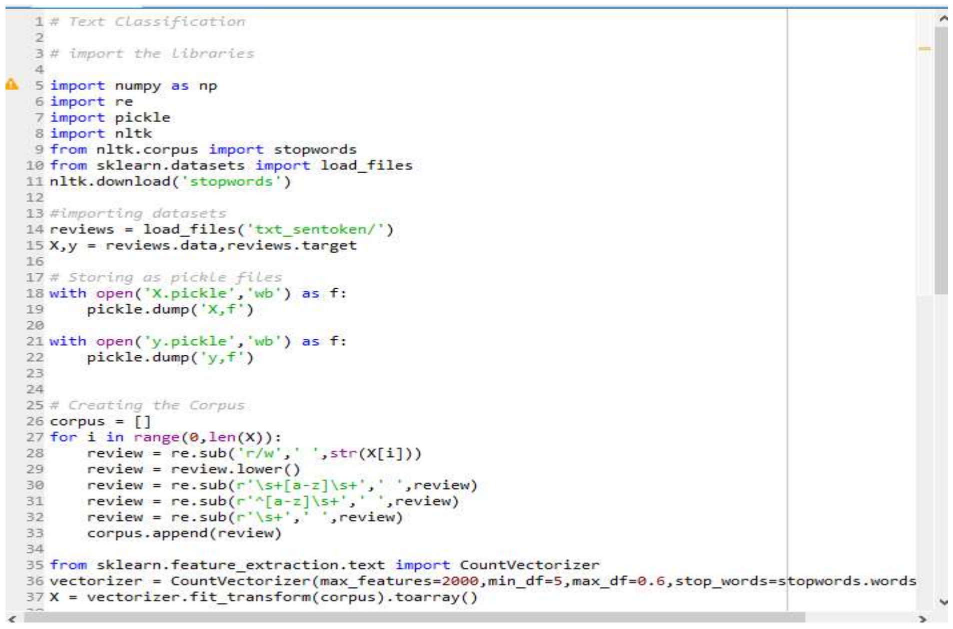
$$\frac{e^{(\beta_0+\beta_1 x)}}{1+e^{(\beta_0+\beta_1 x)}}$$

Process of Implementation:

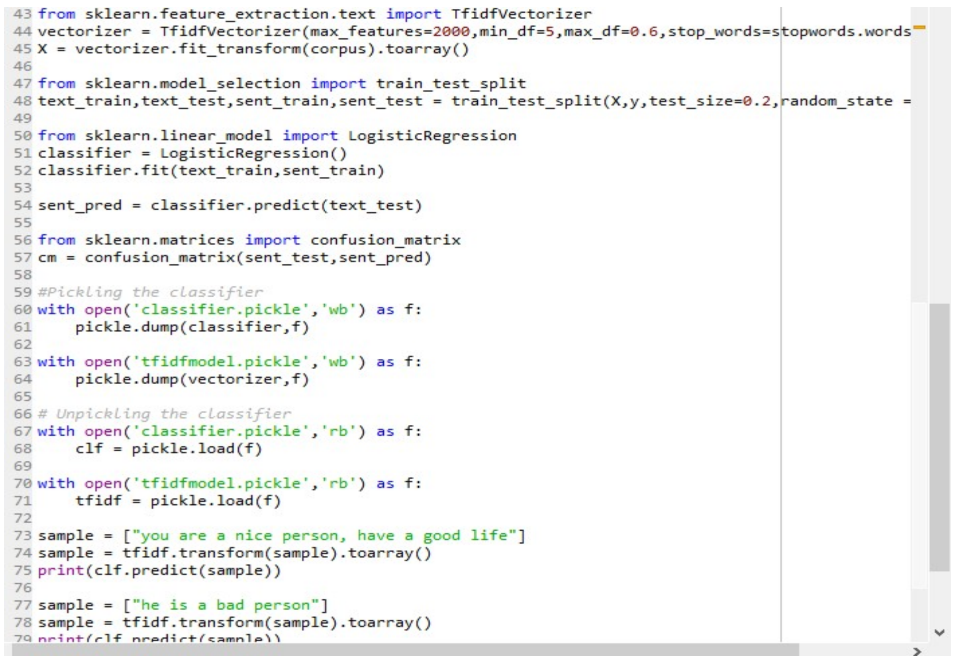
Following are the steps for creating a text classification project:

1. Importing the libraries.
2. Importing the datasets.
3. Pickling and Unpickling the file.
4. Creating corpus model.
5. Text Preprocessing.
6. Converting text to numbers.
7. Training and test sets.
8. Evaluating the model

Discussion:



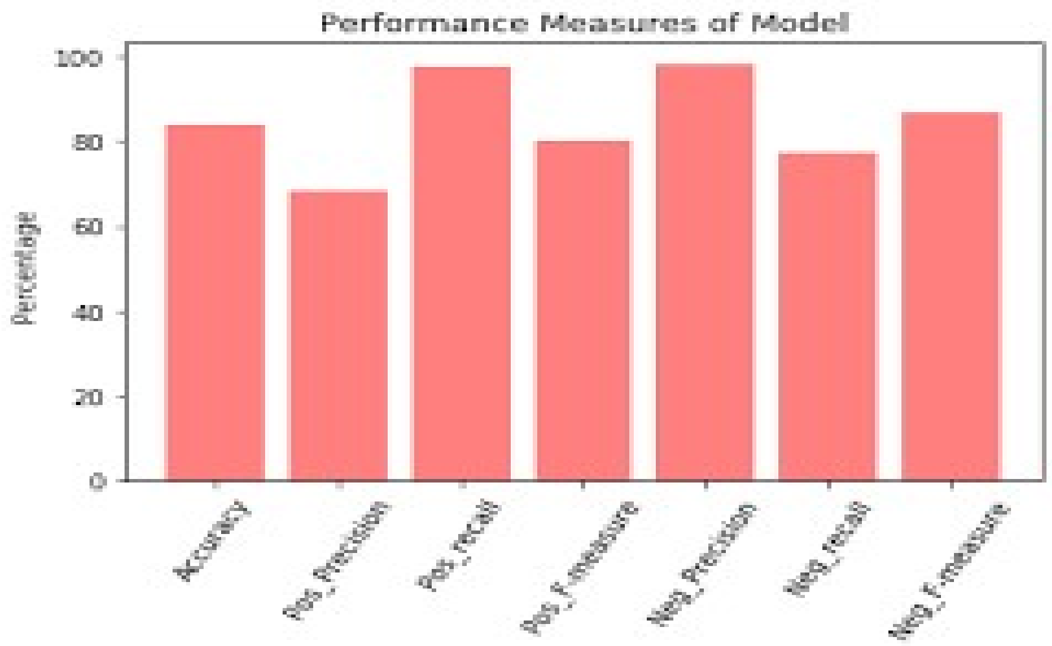
```python
# Text Classification

# import the libraries

import numpy as np
import re
import pickle
import nltk
from nltk.corpus import stopwords
from sklearn.datasets import load_files
nltk.download('stopwords')

#importing datasets
reviews = load_files('txt_sentoken/')
X,y = reviews.data,reviews.target

# Storing as pickle files
with open('X.pickle','wb') as f:
    pickle.dump('X,f')

with open('y.pickle','wb') as f:
    pickle.dump('y,f')


# Creating the Corpus
corpus = []
for i in range(0,len(X)):
    review = re.sub('r/w',' ',str(X[i]))
    review = review.lower()
    review = re.sub(r'\s+[a-z]\s+',' ',review)
    review = re.sub(r'^[a-z]\s+',' ',review)
    review = re.sub(r'\s+',' ',review)
    corpus.append(review)

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=2000,min_df=5,max_df=0.6,stop_words=stopwords.words
X = vectorizer.fit_transform(corpus).toarray()
```



```python
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(max_features=2000,min_df=5,max_df=0.6,stop_words=stopwords.words
X = vectorizer.fit_transform(corpus).toarray()

from sklearn.model_selection import train_test_split
text_train,text_test,sent_train,sent_test = train_test_split(X,y,test_size=0.2,random_state =

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(text_train,sent_train)

sent_pred = classifier.predict(text_test)

from sklearn.matrices import confusion_matrix
cm = confusion_matrix(sent_test,sent_pred)

#Pickling the classifier
with open('classifier.pickle','wb') as f:
    pickle.dump(classifier,f)

with open('tfidfmodel.pickle','wb') as f:
    pickle.dump(vectorizer,f)

# Unpickling the classifier
with open('classifier.pickle','rb') as f:
    clf = pickle.load(f)

with open('tfidfmodel.pickle','rb') as f:
    tfidf = pickle.load(f)

sample = ["you are a nice person, have a good life"]
sample = tfidf.transform(sample).toarray()
print(clf.predict(sample))

sample = ["he is a bad person"]
sample = tfidf.transform(sample).toarray()
print(clf.predict(sample))
```

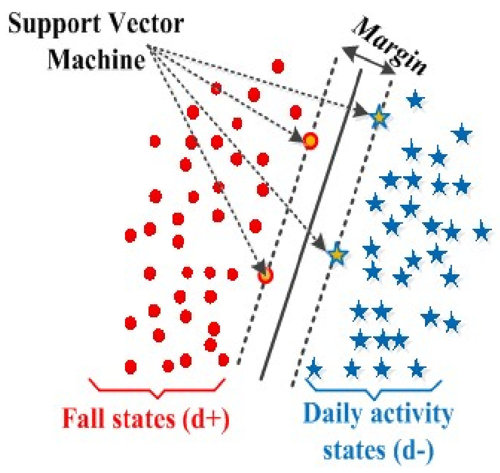
Result:

Calculating movies reviews data using logistic regression algorithm. Here is the result:



Future Scope:

Here we just use one algorithm which is logistic regression algorithm. In future we use Naïve Bayes and SVM algorithm for classifying a text secure safety precaution.



Conclusion:

This work focuses on extracting the new feature from the movie review that have an extremely great effect on deciding the opinion of movie reviews and applying the preprocessing strategy of the information. After this analysis of sentiment is done by applying the extracted features to the supervised learning i.e. Logistic Regression Classifier. The proposed methodology find the pattern of sentences by applying POS (Part of Speech) as well as dependency parser to locate the best feature from the dataset. As per the rules or POS we would offer score to sentence and discover the accuracy by using supervised learning approach. Additionally we might want to bring Big Data into picture.

References:

[1] M. Kepa, J. Szymanski, "Two stage SVM and kNN text documents classifier," In: Pattern Recognition and Machine Intelligence, Kryszkiewicz M. (Ed.), Lecture Notes in Computer Science, Vol. 9124, pp. 279-289, 2015.

[2] X. Fang, "Inference-Based Naive Bayes: Turning Naive Bayes Cost-Sensitive," vol. 25, no. 10, pp. 2302-2314, 2013. [8] C. H. Wan, L. H. Lee, R. Rajkumar, and D. Isa, "A hybrid text classification approach with low dependency on parameter by integrating K-nearest neighbor and support vector machine," Expert Syst. Appl., vol. 39, no. 15, pp. 11880-11888, 2012.

[3] Aggarwal, C., Zhai, C.: A Survey of Text Classification Algorithms. Mining Text Data pp. 163–222 (2012)

[4] Ghavidel Abdi, H., Vazirnezhad B., Bahrani, M.: Persian text classification. In: 4th conference on Information and Knowledge Technology (2012)