Malignant-Comments-Classifier

Submitted by:

Rutuja Patil

ACKNOWLEDGMENT

Nowadays users leave numerous comments on different social networks, news portals, and forums. Some of the comments are toxic or abusive. Due to numbers of comments, it is unfeasible to manually moderate them, so most of the systems use some kind of automatic discovery of toxicity using machine learning models. In this work, we performed a systematic review of the state-of-the-art in toxic comment classification using machine learning methods. We extracted data from 31 selected primary relevant studies. First, we have investigated when and where the papers were published and their maturity level. In our analysis of every primary study we investigated: data set used, evaluation metric, used machine learning methods, classes of toxicity, and comment language.

INTRODUCTION

- ## Business Problem Framing

  Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

- ## Conceptual Background of the Domain Problem

  Toxic comments are defined as comments that are rude, disrespectful, or that tend to force users to leave the discussion. If these toxic comment can be automatically identified, we could have safer discussions on various social networks, news portals, or online forums. Manual moderation of comments is costly, in-effective, and sometimes infeasible. Automatic or semi-automatic detection of toxic comment is done by using different machine learning methods, mostly different deep neural networks architectures.

- ## Review of Literature

  Recently, there is a significant number of research papers on the toxic comment classification problem, but, to date, there has not been a systematic literature review of this research theme, making it difficult to assess the maturity, trends and research gaps. In this work, our main aim was to overcome this by systematically listing, comparing and classifying the existing research on toxic comment classification to find promising research directions. The results of this systematic literature review are beneficial for researchers and natural language processing practitioners.

- ## Motivation for the Problem Undertaken

  The motivating principle behind our project is promoting nonmaleficence within online communities by identifying harmful comments and taking action against them. This is primarily experienced by those who prefer a safe and productive environment without negative distractions.

Analytical Problem Framing

Data Sources and their formats

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes ‘Id’, ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:

- **Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
- **Highly Malignant:** It denotes comments that are highly malignant and hurtful.
- **Rude:** It denotes comments that are very rude and offensive.
- **Threat:** It contains indication of the comments that are giving any threat to someone.

- **Abuse:** It is for comments that are abusive in nature.
- **Loathe:** It describes the comments which are hateful and loathing in nature.
- **ID:** It includes unique Ids associated with each comment text given.
- **Comment text:** This column contains the comments extracted from various social media platforms.

Data Preprocessing Done

Before developing any method to distinguish toxic comments from non-toxic ones, there are a few steps to apply on the data itself before doing anything else onto it.

1)Convert all messages to lower case
train['comment_text'] = train['comment_text'].str.lower()

2)Replace email addresses with 'email'
train['comment_text'] = train['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
'emailaddress')

3)Replace URLs with 'webaddress'
train['comment_text'] = train['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$', 'webaddress')
4) Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
train['comment_text'] = train['comment_text'].str.replace(r'£|\$', 'dollers')

5) Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber' train['comment_text'] =
train['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$', 'phonenumber')

6)Replace numbers with 'numbr'

```
train['comment_text'] = train['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')


train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in string.punctuation))

stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))

lem=WordNetLemmatizer()
train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
 lem.lemmatize(t) for t in x.split()))
```

- ## Hardware and Software Requirements and Tools Used

  Python was the major technology used for the implementation of machine learning concepts the reason being that there are numerous inbuilt methods in the form of packaged libraries present in python. Following are prominent libraries/tools we used in our project.

  1)NUMPY

- NumPy is a general-purpose array-processing package. it provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.
- Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

  2)JUPYTER NOTEBOOK

  The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It includes data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

  3)LIBRABIRES USED-

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report,roc_curve,roc_auc_score,auc
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import
accuracy_score,classification_report,confusion_matrix,f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score,GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import
RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string
```

Model/s Development and Evaluation

- ## Testing of Identified Approaches (Algorithms)
    a. Logistic Regression
    b. Decision Tree Classifier
    c. Random Forest Classifier
    d. Xgboost
    e. AdaBoost Classifier
    f. KNeighbors Classifier

- ## Run and Evaluate selected models

1)Logistic Regression

```
LG = LogisticRegression(C=1, max_iter = 3000)

LG.fit(x_train, y_train)

y_pred_train = LG.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = LG.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

<u>Result-</u>

```
Training accuracy is 0.9595520103134316
Test accuracy is 0.9552974598930482
```

2)Decision Tree Classifier

```
DT = DecisionTreeClassifier()

DT.fit(x_train, y_train)
y_pred_train = DT.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = DT.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

<u>Result-</u>

```
Training accuracy is 0.9988898736783678
Test accuracy is 0.9391920120320856
```

3)Random Forest Classifier

```
        RF = RandomForestClassifier()

RF.fit(x_train, y_train)
y_pred_train = RF.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

Result-

```
Training accuracy is 0.9988809210467416
Test accuracy is 0.9546707887700535
```

4)Xgboost

```
import xgboost
xgb = xgboost.XGBClassifier()
xgb.fit(x_train, y_train)
y_pred_train = xgb.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = xgb.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

Result-

```
Training accuracy is 0.9614052050600274
Test accuracy is 0.9526236631016043
```

5)AdaBoost Classifier

```
                ada=AdaBoostClassifier(n_estimators=100)
ada.fit(x_train, y_train)
y_pred_train = ada.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = ada.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

Result-

```
Training accuracy is 0.951118631321677
Test accuracy is 0.9490307486631016
```

6)KNeighbors Classifier

```
        knn=KNeighborsClassifier(n_neighbors=9)
knn.fit(x_train, y_train)
y_pred_train = knn.predict(x_train)
```

```
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = knn.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

<u>Result-</u>

```
Training accuracy is 0.922300110117369
Test accuracy is 0.9173629679144385
```

- ## Key Metrics for success in solving problem under consideration

a. Logistic Regression

	precision	recall	f1-score	support
0	0.96	0.99	0.98	42950
1	0.93	0.61	0.74	4922
accuracy			0.96	47872
macro avg	0.94	0.80	0.86	47872
weighted avg	0.95	0.96	0.95	47872

b. Decision Tree Classifier

	precision	recall	f1-score	support
0	0.96	0.97	0.97	42950
1	0.71	0.69	0.70	4922
accuracy			0.94	47872
macro avg	0.84	0.83	0.83	47872
weighted avg	0.94	0.94	0.94	47872

c. Random Forest Classifier

	precision	recall	f1-score	support
0	0.96	0.99	0.98	42950
1	0.85	0.67	0.75	4922
accuracy			0.95	47872
macro avg	0.91	0.83	0.86	47872
weighted avg	0.95	0.95	0.95	47872

d. Xgboost

	precision	recall	f1-score	support
0	0.96	0.99	0.97	42950
1	0.92	0.59	0.72	4922
accuracy			0.95	47872
macro avg	0.94	0.79	0.85	47872
weighted avg	0.95	0.95	0.95	47872

e. AdaBoost Classifier

	precision	recall	f1-score	support
0	0.95	0.99	0.97	42950
1	0.88	0.58	0.70	4922
accuracy			0.95	47872
macro avg	0.92	0.79	0.84	47872
weighted avg	0.95	0.95	0.94	47872

f. KNeighbors Classifier

	precision	recall	f1-score	support
0	0.92	1.00	0.96	42950
1	0.89	0.22	0.36	4922
accuracy			0.92	47872
macro avg	0.90	0.61	0.66	47872
weighted avg	0.91	0.92	0.89	47872

CONCLUSION

Toxic comment classification is a complex research problem tackled by several machine learning methods. Our research has shown that harmful or toxic comments in the social media space have many negative impacts to society. The ability to readily and accurately identify comments as toxic could provide many benefits while mitigating the harm. Also, our research has shown the capability of readily available algorithms to be employed in such a way to address this challenge.