

**MACHINE LEARNING**

**UE17EC337**

**Report On:**

**SOCIAL MEDIA SENTIMENT ANALYSER**

By:

|  |  |
| --- | --- |
| Kushal Nagaraj | PES1201701567 |
| Nagavishnu BK | PES1201700421 |
| Ratan Gundami | PES1201701556 |

**TABLE OF CONTENTS**

[**INTRODUCTION**: 3](#_Toc38055116)

[**PROBLEM STATEMENT and OBJECTIVE:** 3](#_Toc38055117)

[**METHODOLOGY:** 3](#_Toc38055118)

[**TOOLS USED:** 4](#_Toc38055119)

[**DATASETS USED:** 4](#_Toc38055120)

[**IMPLEMENTATION:** 5](#_Toc38055121)

[SOME PRE-REQUIRED KNOWLEDGE: 5](#_Toc38055122)

[STEPS: 6](#_Toc38055123)

[**RESULTS AND ANALYSIS:** 7](#_Toc38055124)

[1. Training Results: 7](#_Toc38055125)

[2. Prediction Results: 8](#_Toc38055126)

[3. Twitter Analysis: 8](#_Toc38055127)

[**CONCLUSION:** 9](#_Toc38055128)

[**FUTURE SCOPE:** 9](#_Toc38055129)

[**REFERENCES:** 9](#_Toc38055130)

[**APPENDDIX :** 9](#_Toc38055134)

[A. Code 9](#_Toc38055135)

[B. Sample data: 12](#_Toc38055136)

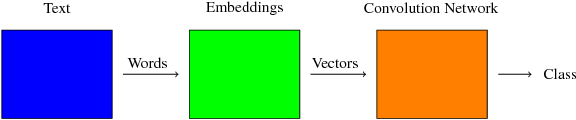
# **INTRODUCTION**:

We all know that there is a drastic increase in accessibility of internet and texting. This has led to more Toxic and Obscene conversations on the internet. Especially on the Social Networking sites like Twitter, where a huge amount of conversations takes place on various matters. Most of the messages sent are toxic and spread hatred. It’s hard to manually control every message as there are as many as a billion users in Twitter. Not only this, but a lot of bullying takes place online via text messages on other platforms too. According to Cyberbullying Research, more than half of the users have experienced threats or any other type of bullying. Thus we came up with the idea of lowering the usage of such messages by predicting the type of a message. The challenge is particularly interesting because of the heated discussions that toxic content online has influenced the overall health of the society. It is also interesting that the service providers are finally leveraging Machine Learning to supervise their service in a scalable way. So we thought of applying Machine Learning here to classify the types of messages. We developed a Deep learning and Natural Language Processing model to detect toxicity level of various comments and differentiating those into various categories based on the sentiment. By doing so we hope that we can control the usage of abusive language and threats. It can help us reduce the content which is toxic and obscene, hence leading to more discussion and user participation in online forum and also to create harmony among everyone.

# **PROBLEM STATEMENT and OBJECTIVE:**

To develop a model so that it will analyze a particular message and classify that message into different categories based on the sentiment. Also predict if it is an abusive, threat message or a happy, plain message. Here we are labeling our categories as *‘toxic’, ’severe toxic’, ‘obscene’, ‘threat’, ‘insult’ and ‘identity hate’.* Based on this we can have the idea on type of the message. At the end of this project, we will be able to predict what level of a particular category a message will be. For e.g. If the message gives 0 % threat but 100% insult it means that the message is insulting something or someone. A happy statement would have 0% in all the categories.

# **METHODOLOGY:**



* **Preprocessing**: Here we are cleaning the data so that what we have is just words and removing all the special characters which are of no use in our project. We are tokenizing each word from a sentence so that we don’t interfere with a new word every now and then. We are also calculating some statistics of our training and test data sets which we can use later.
* **Embedding**: The vector created above are one-hot encoding and requires a lot of space to store every word as there are thousands of words. Hence we are continuously embedding our words using Word2Vec. The benefits to this continuous embedding is that words with the similar predictive power will appear closer together on our word vector.
* **Neural Network Modeling:** Here we model our neural network layers and train them from the data set we processed from the above procedures. We will use CNN as a hidden layer and another fully connected hidden layer. We will adopt Adam optimizer and we will keep loss function as binary crossentropy.

# **TOOLS USED:**

* **KERAS**: We will use Keras for most of our implementation as it is flexible and easy to use. We will use tokenizers and padding for some parts of preprocessing and will also use models and layers for developing our own neural network.
* **NLTK**: We will use this Natural language Toolkit for word tokenizing as it is more efficient. We will use an already trained and developed model ‘punkt’.
* **GENSIM:** We will use Word2Vec from GENSIM for our embedding purpose. This is where it continuously embeds the words by creating high-dimensional word embedding of lower-dimensional vectors and thereby reducing size.
* We also used **Numpy** and **Pandas** for our convenience.

# **https://miro.medium.com/max/930/1*qADtcBq7-S3rC9k2HqeiCQ.jpegDATASETS USED:**

We are using train and test dataset which we got from Kaggle Competition where they did Naive Based Support Vector Machine model using these datasets. The data used consist of many Wikipedia comments which have been labelled by humans according to their relative toxicity. The train.csv file which is our training set, contains comments with their binary labels.

Toxic levels in the comments classification in train dataset

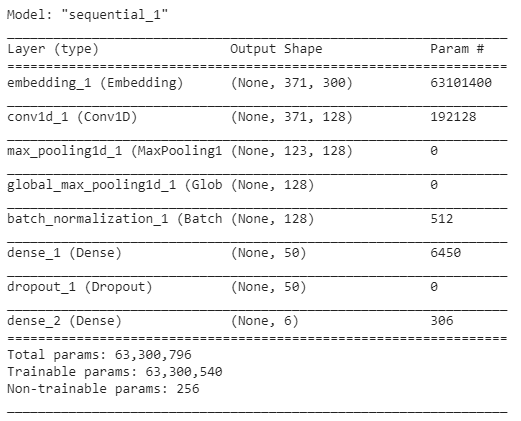
# **IMPLEMENTATION:**

## SOME PRE-REQUIRED KNOWLEDGE:

* Tokenizer in Keras: It comes under text preprocessing. This class allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf, etc.
* Word2Vec in Genism: Word2Vec is a more recent model that embeds words in a lower-dimensional vector space using a shallow neural network. The result is a set of word-vectors where vectors close together in vector space have similar meanings based on context and word-vectors distant to each other have differing meanings. For example, strong and powerful would be close together and strong and Paris would be relatively far. It takes in pairs (word1, word2) generated by moving a window across text data, and trains a 1-hidden-layer neural network based on the synthetic task of given an input word, giving us a predicted probability distribution of nearby words to the input. If the hidden layer has 300 neurons, this network will give us 300-dimensional word embeddings.
* Punkt Sentence Tokenizer in Natural Language Toolkit: The NLTK data package includes a pre-trained Punkt tokenizer for English. This tokenizer divides a text into a list of sentence by using an unsupervised algorithm to build a model for abbreviation words, collocations, and words that start sentences. It must be trained on a large collection of plain text in the target language before it can be used.
* Convolutional Neural Network - CNN (1d): [Convolutional neural network](https://machinelearningmastery.com/crash-course-convolutional-neural-networks/) models were developed for image classification problems, where the model learns an internal representation of a two-dimensional input, in a process referred to as feature learning. This same process can be harnessed on one-dimensional sequences of data. The model learns to extract features from sequences of observations and how to map the internal features to different activity types. The benefit of using CNNs for sequence classification is that they can learn from the raw time series data directly and in turn do not require domain expertise to manually engineer input features.
* Binary Cross-Entropy loss: Also called **Sigmoid Cross-Entropy loss.** It is a **Sigmoid activation** plus a **Cross-Entropy loss**. Unlike **Softmax loss** it is independent for each vector component (class), meaning that the loss computed for every CNN output vector component is not affected by other component values. That’s why it is used for **multi-label classification**, were the insight of an element belonging to a certain class should not influence the decision for another class.

## STEPS:

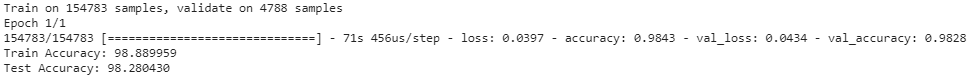
1. Pre-processing :
   * First we defined all the categories i.e. {toxic, severe\_toxic, threat, obscene, insult, identity\_hate} as a list.
   * We will also import the training and testing csv files using pandas and define x\_train, y\_train and x\_test
   * Implement the Tokenizer from Keras on both x\_train and x\_test with the default parameters. Make sure that char level = FALSE as we don’t want char level tokens.
   * We will perform post-padding on both the sets based on the max length we calculate from tokenizing.
2. Embedding :
   * Then download punkt from NLKT and perform word\_tokenize on all the words present in both x\_train and x\_test and add it to a list. This list converts non-meaningful sentences into meaningful sentences. That’s why NLP is an important part in this project.
   * Then create a Word2Vec model from the above list and pass the required parameters. Here we are creating a 300-dimensional embedding. Save this model.
   * Now we have to create out embedding matrix. To create this, first we get our embedding\_index (which is our total word vectors) with the help of numpy arrays.
   * Then we initialize the embedded\_matrix with zeros and for every vector (i.e. every word) we pass that vector with the index (which we got from the previous step) into the embedded\_matrix.
3. Neural Network Modeling :
   * The First layer is the embedded layer whose weights we will pass as the values of embedded\_matrix we got from embedding. To execute this we will use embedding function from embedding library in Keras.
   * The Second layer will be the 1D Convolution Layer with 128 neurons and kernel size as 5. We will go with relu (Rectified Linear) activation function. We are going to implement MaxPooling, GlobalMaxPooling for 1D after Convolution followed by Normalization. (Note: Though these are different layers in Keras technically the combine to a single neuron layer.)
   * The Third layer is fully connected layer of 50 neurons and relu as activation function. Here we will give the dropout (i.e. the fraction of I/P which is zero) as 0.3 to prevent overfitting.
   * The Fourth and the last layer is the Output Layer with 6 neurons, each for each category, with the activation function as sigmoid.

Here’s the summary of our Neural Network:

1. Training :
   * We will take the help of sklearn to split the training and validation set from training set. We will also use this sklearn to create y\_pred in a specific format from y\_train and y\_val so that we can use it easily.
   * We then compile the model with binary crossentropy as loss function and Adam as optimizer.
   * Then train the model with epochs = 1 and batch size = 64. Calculate the accuracy for both training and validation sets. Make any necessary changes to increase the accuracy but make sure it don’t overfit.

# **RESULTS AND ANALYSIS:**

## Training Results:



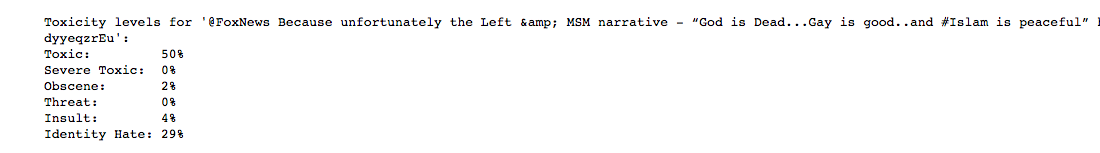
Train Accuracy: 98.89%

Test (Validation) Accuracy: 98.28%

|  |  |
| --- | --- |
|  | This is the result for the message:  “I will kill you”. The prediction should be a threat and from the probabilities we can see that it is a threat |
|  | This is the result for the message:  “Have a nice day”. The prediction should be it’s not toxic and from the probabilities we can see that it is a non-toxic and a happy message. |
|  | This is the result for the message:  “Go jump off a bridge jerk”. The prediction should be usage of offensive language and insult of a person which resembles the result. |
|  | This is the result for the message:  “Hola mierda joder”. This is an abusive sentence in Spanish. The prediction should be obscene but we can see the wrong prediction because we did not train Spanish sentences. |

## Prediction Results:

## https://miro.medium.com/max/1300/1*niDtT4UiB41A1zgZbw2V3g.pngTwitter Analysis:



# **CONCLUSION:**

We are able to successfully predict the sentiment of the message with the help of our Deep learning and NLP Model. Due to the different categories we can successfully categorize the statements into abusive usage of language or threats or any other categories depending on the probabilities.

# **FUTURE SCOPE:**

Due to a very high success rate, our model can be used in the Social networking sites like Twitter to identify the offensive message. Hence we can control cyber-bullying to an extent where if we detect any offensive messages we can straight away block the message. This model can also be implemented in search engines where we can see if the queries asked by user are valid or invalid.

# **REFERENCES:**

1. Datasets : <https://www.kaggle.com/jhoward/nb-svm-strong-linear-baseline/data>
2. Keras Documentation for programming - <https://keras.io/>
3. Sentiment analysis using convolutional neural network with fastText embeddings by Igor Santos, Nadia Nedjah. DOI- 10.1109/LA-CCI.2017.8285683
4. About 1D-CNN - <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
5. Dependency Based Convolutional Neural Networks for Sentence Embedding by Mingbo Ma, Liang Huang, Bowen Zhou and Bing Xiang.

# **APPENDDIX :**

1. Code:

import sys, os, re, csv, codecs, numpy as np, pandas as pd

import matplotlib.pyplot as plt

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

from keras.layers import Dense, Input, Embedding, Dropout, Activation

from keras.layers import GlobalMaxPool1D

from keras.models import Model

from keras import constraints, optimizers, layers

from google.colab import drive

drive.mount("/content/gdrive")

#Class labels

list\_classes = ["toxic", "severe\_toxic", "obscene", "threat", "insult", "identity\_hate"]

#Read the data

toxicWordsTrain = pd.read\_csv("/content/gdrive/My Drive/Colab Notebooks/train.csv");

toxicWordsTest = pd.read\_csv("/content/gdrive/My Drive/Colab Notebooks/test.csv")

y\_train = toxicWordsTrain[list\_classes].values

x\_train = toxicWordsTrain["comment\_text"]

x\_test = toxicWordsTest["comment\_text"]

# Tokenize and Pad

tokenizer = Tokenizer(num\_words=None,

filters='!"#$%&()\*+,-./:;<=>?@[\\]^\_`{|}~\t\n',

lower=True,

split=" ",

char\_level=False)

tokenizer.fit\_on\_texts(list(x\_train))

tokenized\_train = tokenizer.texts\_to\_sequences(x\_train)

tokenized\_test = tokenizer.texts\_to\_sequences(x\_test)

word\_index = tokenizer.word\_index

vocab\_size = len(word\_index)

print('Vocab size: {}'.format(vocab\_size))

longest = max(len(seq) for seq in tokenized\_train)

print("Longest comment size: {}".format(longest))

average = np.mean([len(seq) for seq in tokenized\_train])

print("Average comment size: {}".format(average))

stdev = np.std([len(seq) for seq in tokenized\_train])

print("Stdev of comment size: {}".format(stdev))

max\_len = int(average + stdev \* 3)

print('Max comment size: {}'.format(max\_len))

print()

processed\_X\_train = pad\_sequences(tokenized\_train, maxlen=max\_len, padding='post', truncating='post')

processed\_X\_test = pad\_sequences(tokenized\_test, maxlen=max\_len, padding='post', truncating='post')

#Embeddings

import gensim

from gensim.models import Word2Vec

from os import listdir

from numpy import array

from numpy import asarray

from numpy import zeros

import nltk

from nltk.tokenize import word\_tokenize

nltk.download('punkt')

l4=[]

for i in x\_train:

t3=word\_tokenize(i)

l4.append(t3)

for i in x\_test:

t3=word\_tokenize(i)

l4.append(t3)

#print(l4)

model = Word2Vec(l4, size=300, window=5, min\_count=10,sg=1,iter=20)

model.save("w2v.model")

words = list(model.wv.vocab)

vocab\_size = len(words)

#save model in ASCII (word2vec) format

filename = 'embedding\_word2vec.txt'

model.wv.save\_word2vec\_format(filename, binary=False)

#create embedding matrix

embedding\_dim = 300

embeddings\_index = {}

f = open('embedding\_word2vec.txt')

for line in f:

values = line.split()

word = values[0]

coefs = asarray(values[1:], dtype='float32')

embeddings\_index[word] = coefs

f.close()

#print('Found {} word vectors.'.format(len(embeddings\_index)))

embedding\_matrix = zeros((len(word\_index)+1, embedding\_dim))

for word, i in word\_index.items():

if i > vocab\_size-1:

break

else:

embedding\_vector = embeddings\_index.get(word)

if embedding\_vector is not None:

embedding\_matrix[i] = embedding\_vector

#neural network modeling

import keras.backend

from keras.models import Sequential

from keras.layers import Dense, Conv1D, MaxPooling1D

from keras.layers import Dropout, GlobalMaxPooling1D, BatchNormalization

from keras.layers.embeddings import Embedding

from keras.optimizers import Adam

model = Sequential()

# Add Embedding layer

model.add(Embedding(len(word\_index)+1, embedding\_dim, weights=[embedding\_matrix], input\_length=max\_len, trainable=True))

# Add Convolutional layer

model.add(Conv1D(filters=128, kernel\_size=5, padding='same', activation='relu'))

model.add(MaxPooling1D(3))

model.add(GlobalMaxPooling1D())

model.add(BatchNormalization())

# Add fully connected layers

model.add(Dense(50, activation='relu'))

model.add(Dropout(0.3))

model.add(Dense(6, activation='sigmoid'))

model.summary()

#training

from sklearn.model\_selection import train\_test\_split

[X, X\_val, y, y\_val] = train\_test\_split(processed\_X\_train, y\_train, test\_size=0.03, shuffle=False)

model.compile(loss='binary\_crossentropy', optimizer='Adam',metrics=['accuracy'])

graph = model.fit(X, y, epochs=1,batch\_size=64,validation\_data=(X\_val, y\_val),verbose=1)

loss, acc\_train = model.evaluate(X, y, verbose=0)

loss, acc = model.evaluate(X\_val, y\_val, verbose=0)

print('Train Accuracy: %f' % (acc\_train\*100))

print('Test Accuracy: %f' % (acc\*100))

#testing

predictions = model.predict(processed\_X\_test, verbose=0)

#process string

def toxicity\_level(string):

new\_string = [string]

new\_string = tokenizer.texts\_to\_sequences(new\_string)

new\_string = pad\_sequences(new\_string, maxlen=max\_len, padding='post', truncating='post')

prediction = model.predict(new\_string)

print("Toxicity levels for '{}':".format(string))

print('Toxic: {:.0%}'.format(prediction[0][0]))

print('Severe Toxic: {:.0%}'.format(prediction[0][1]))

print('Obscene: {:.0%}'.format(prediction[0][2]))

print('Threat: {:.0%}'.format(prediction[0][3]))

print('Insult: {:.0%}'.format(prediction[0][4]))

print('Identity Hate: {:.0%}'.format(prediction[0][5]))

print()

return

toxicity\_level('i will kill you')

toxicity\_level('have a nice day')

toxicity\_level('go jump off a bridge jerk')

toxicity\_level('hola mierda joder')

## Sample data:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | Comment text | toxic | Severe\_toxic | obscene | threat | insult | Identity hate |
| ff7093e953e68d1e | Ok I am sorry, but I have created some articles that you have improved, so I'm not all bad. | 0 | 0 | 0 | 0 | 0 | 0 |
| 0020e7119b96eeeb | Stupid peace of shit stop deleting my stuff asshole go die and fall in a hole go to hell! | 1 | 1 | 1 | 0 | 1 | 0 |
| c5aa95fe38bcfe3a | Your a loser. Why the hell would you delete a page of a band on the verge of making it? | 1 | 0 | 0 | 0 | 1 | 0 |

--------------------------------------------------------------------------------------------------------------------------