* **Describe an abusive language detection pipeline on a social network. You can use diagrams. (your answer should not exceed one page)**

This pipeline was aimed at classifying input texts according to the traits of abusive and non-abusive statements. This study was aimed at developing a technique based on a convolutional neural network + long short-term memory (CNN-LSTM) model by using a deep learning approach to detect abusive sentences.

There are different kinds of methods for detection pipeline approach which are content-based, graph-based & fusion method

**Content-based method**: It consists in extracting certain features from the content of each considered message, and to train a Support Vector Machine (SVM) classifier to distinguish abusive (*Abuse* class) and non-abusive (*non-abuse* class). These features are quite standard in Natural Language Processing (NLP).

**Graph-based method**: It completely ignores the content of the messages, and only focuses on the dynamics of the conversation, based on the interactions between its participants. It is three-stepped: (1) extracting a conversational graph based on the considered message as well as the messages preceding and/or following it; (2) computing the topological measures of this graph to characterize its structure; and (3) using these values as features to train an SVM to distinguish between abusive and non-abusive messages. The vertices of the graph model the participants of the conversation, whereas its weighted edges represent how intensely they communicate.

**Fusion method**: This is new approach for classification problems. It is a combination of content-based and graph-based method. It is a new method seeking to take advantage of both previously described ones. It is based on the assumption that the content- and graph-based features convey different information. Therefore, they could be complementary, and their combination could improve the classification performance

Implementation of Text classification pipeline for the prediction of abusive statements in social media

**Data collection (social media)**:

scraping data from Google, twitter, face book, Instagram, Research datasets, Kaggle data, offline data

(Combination of texts, images)

**Model training 2:**

**State of the art models:**

BERT, XL BERT

As many researches proved bidirectional transformers proved good results in classification problems

1

**Data Visulaziation :**

we implement different visualisations and word cloud to see the most frequent used abusive words

3

**Data Preprocessing:**

we create pipeline using NLTK, GENSIM to clean all text data.

2

**Model training 1** :

Splitting data into train and test sets

(1. Approach with SVM, naïve bayes classifier (ML))

(2. deep learning algorithms like

Char CNN, word CNN, hybrid CNN-LSTM)

A CNN-LSTM hybrid model yields good accuracy as proved by the researchers

Otherwise implement step 6

**Word Embedding Models:**

Converting text into vectors

(Word2vec, Fast text, etc.)

We may use this also as a custom embedding matrix while we train neural network

**Model evaluation & validation :**

Test accuracy

F1 score

Precision

Recall

Confusion matrix

Validate our model based on test accuracy

4

6

5

7

**Model Deployment :**

Deploying model using cloud virtual machine

like AZURE, AWS, etc

Model Registry

Model Serving

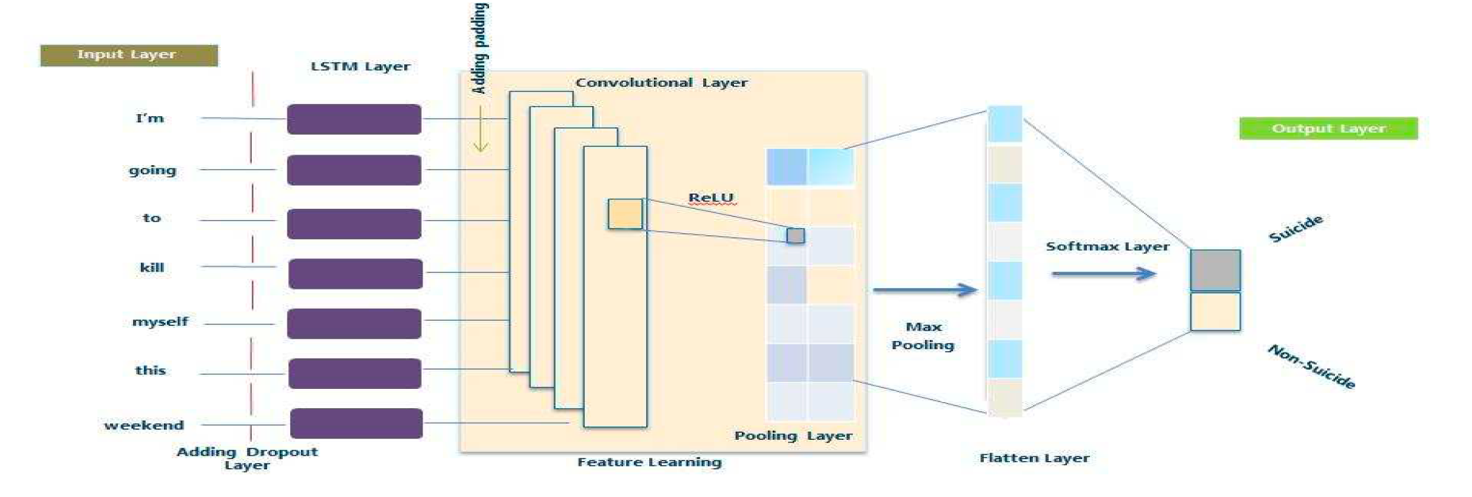
10

8

9

The problem can be simplified by sequence predictions based on the keywords that are abusive. we need to filter the most frequently used abusive words and map them to a sentence to predict whether the text is abusive or non-abusive and if they are then they can be removed from the platform to minimize the social instability. We as humans are able to detect whether the post is offensive or not and here, we are fusing this capability into machines so it relates well with AI. As it deals with understanding of the human written text by machines then this makes it an NLP problem too. Lastly, we need to understand how the ML and DL applies to our problem statement. A system or a computer program will be trained on the various offensive and non-offensive tweets so that it recognizes the patterns from them to distinguish between the two. Now in future our Machine Learning or Deep Learning model can accurately classify the unseen tweet as offensive/non-offensive and further action could be automated.

* **Explain the process for using a hybrid CNN-LSTM algorithm to detect aggressive content in a post.**



Non Abusive

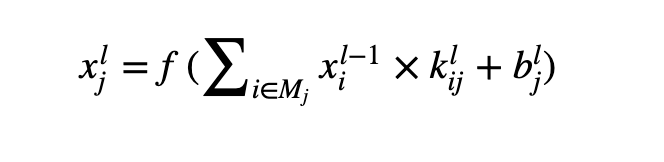
Abusive

SENTENCE

CNN-LSTM Architecture for Abusive and non-Abusive Detection

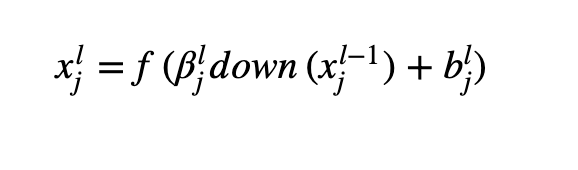
**Model Implementation Process for Text and Images:**

It can be seen from that this algorithm is mainly divided into feature extraction stage based on CNN and feature fusion stage based on LSTM. In the feature extraction stage, the forward propagation process of the image or text signal is as follows: it is assumed that the l layer is a convolutional layer, and the l − 1 layer is a pooling layer or an input layer. Then the calculation formula of the l layer is:



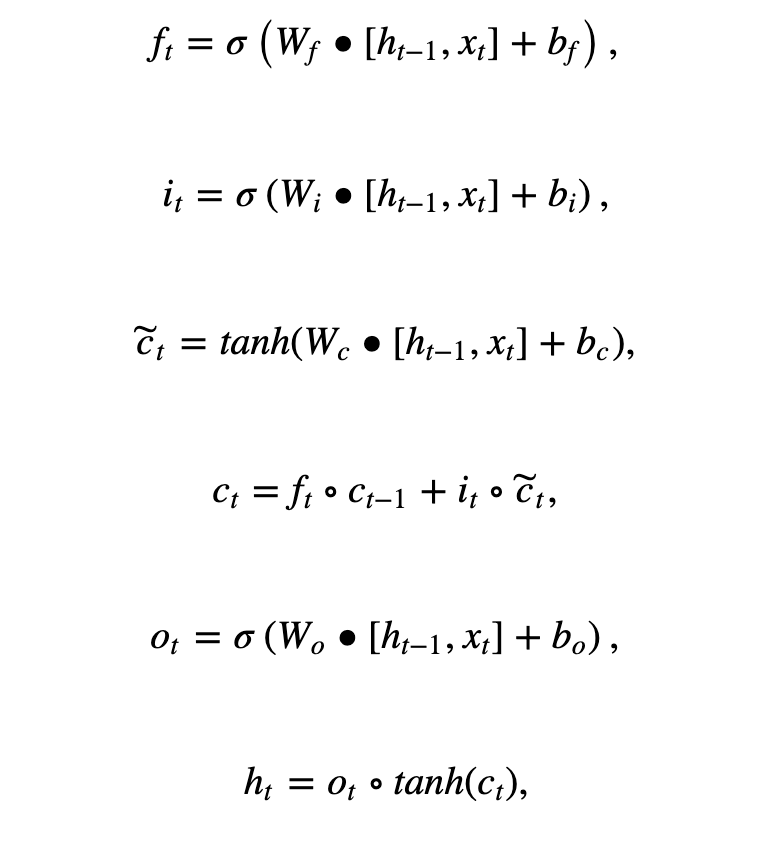
The xlj on the left of the above equation represents the *j*th feature text or image of the *l* layer. The right side shows the convolution operation and summation for all associated feature maps xl−1i of the *l* − 1 layer and the *j*th convolutional kernel of the *l*th layer, and then adds an offset parameter, and finally passes the activation function *f*(\*). Among them, *l* is the number of layers, *f* is the activation function, *Mj* is an input feature map of the upper layer, *b* is offset, and *k* is convolutional kernel.

Assuming that the *l* layer is pooling layer (down sampling layer), the *l* − 1 layer is the convolutional layer. The formula for the *l* layer is as follows:



In the above formula, *l* is the number of pooling layer, *f* is the activation function, *down(\*)* is the down sampling function; *β* is the down sampling coefficient, and *b* is the offset.

In the feature hybrid stage, the network uses three threshold structures to control the state of the cell that preserves long-term memory. The meaning of long short-term memory is: *ct* corresponds to long-term memory, and c˜t corresponds to short-term memory. The σ(\*) in Expressions is a Sigmoid function. If the output of Sigmoid function is 1, then the information is fully remembered. If the output is 0, then it is completely forgotten. If the output is the value between 0 and 1, it is the proportion of information to be remembered. The gate is actually equivalent to a fully connected layer and its input is a vector and output is a real vector between 0 and 1. It uses the output vector of the “gate” multiplied by the vector we want to control. The forgetting gate *ft* determines how much historical information can be remained in a long-term state *ct*; c˜t is used to describe the short-term state of current input. The input gate *it* determines how much of the current network input information can be added to the long-term state *ct*; the output gate *ot* controls how much of the aggregated information is available as the current output. The expressions are as follows:



The above are the formulas of the forward propagation process of the image signal. “•” means matrix multiplication, and “∘” means multiplication by elements of the same position. The output of the last time step of the LSTM network includes current unit state c64 and current output h64. We take h64 as the overall output of the LSTM part, which is the input of SOFTMAX. After the signal passed through the SOFTMAX, the judgment of the category is given in the form of probability. In the algorithm training stage, the network adopts the error back propagation method to iteratively update the weights and offsets until the number of epochs is reached.

In our experiment, we use multiple convolutional filters with various parameter initializations to extract multiple maps from the text.

**Convolutional Layer**

Convolutional layer is a part of CNN neural network initially designed for an image recognition with a strong performance. In recent years, however, CNN has become an incredibly versatile model used for a wide range of multiple text classification tasks. When applying CNN on a well-structured and organized text, the model will discover and learn patterns that would otherwise be lost in a feed-forward network. For instance, a word “down” in the context of “down to earth” and “feeling down” has a different sentiment.

In addition, CNN can extract features regardless of where they occur in a sentence. CNN is similar to Feed-forward Neural Networks where the connections between the nodes do not form a cycle. Thus, a single neuron in CNN represents a region within an input sample such as a piece of image or text.

After each feature sequence is extracted by the LSTM model which is H = [h1, h2, h3, ..., hT] T where stands hT for a m-dimensional feature vector of the word in the text sequence where T is the number of LSTM expansion steps equal to the text sequence length. the CNN input matrix with fixed-length inputs; thus, every input length is standardized to T by trimming the longer sentences and padding the shorter sentences with zeros. The convolutional filter where j is the number of the words in the window, k is the dimension of the word embedding vector.

**Pooling Layer**

Pooling layer’s function is to minimize a dimensionality of each rectified feature map and retain the most important information. Its characteristic feature is to make the input representations smaller and more manageable aggregating information. It reduces the number of parameters and computations in the network resulting in an ability to control over-fitting. In our study, we use a max pooling

operation, which represents the most important information in each feature map.

**Flatten Layer**

CNN flatten layer aims to transform a pooled feature map into a column vector which makes an input to the neural network of the classification task. As the next step, the pooled feature maps are flattened through a reshape function to make the feature vector pulls concatenated.

**Output Layer**

Main function of output or fully connected layer is to calculate a probability of suicide and non-suicide text. It uses a text feature vector from a convolutional and pooling layer’s output which is followed by considerable activation functions for preventing gradient explosion or vanishing problems. We can apply Sigmoid function , SoftMax function, Hyperbolic tangent function or Rectified linear unit widely used in classifying an input text into a binary classification based on the labelled training dataset.

* **Implement using python3 a Facebook and / or Instagram connector allowing to collect posts (image, text and comments linked to images) by report to a defined subject, example "the death of President Jacques Chirac".  
  It is recommended to store texts and images in a MongoDB database.**

**Scrapped my profile that has images and text comments**

**Step1:**

**Installing the required packages in new environment:**

**Selenium, wget, requests, beautifulsoup4, logging, collections, re, json**

**Firstly, Import them**

#imports here

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.common.by import By

from selenium.webdriver.support.wait import WebDriverWait

import time

**Step2:**

**Need to install chrome web driver in the specified location and Login:**

### Disabling Alerts/Notifications

chrome\_options = webdriver.ChromeOptions()

prefs = {"profile.default\_content\_setting\_values.notifications" : 2}

chrome\_options.add\_experimental\_option("prefs",prefs)

#specify the path to chromedriver.exe (download and save on your computer)

driver = webdriver.Chrome('/Users/tulasiramponaganti/Desktop/WebscrapingFacebook-main/chromedriver',chrome\_options=chrome\_options)

#open the webpage

driver.get("http://www.facebook.com")

#target username

username = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "input[name='email']")))

password = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "input[name='pass']")))

#Remember to enter username and password

username.clear()

username.send\_keys("tulasi\*\*\*\*\*@gmail.com")

password.clear()

password.send\_keys("\*\*\*\*\*\*\*\*")

#target the login button and click it

button = WebDriverWait(driver, 2).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "button[type='submit']"))).click()

#We are logged in!

STEP3:

Extracting Images from my photo gallery of a profile

#wait 5 seconds to allow your new page to load

time.sleep(5)

images = []

#itterate over both uploaded and tagged images respectively

for i in ["photos\_by"]:

driver.get("https://www.facebook.com/tulasi.ram.56863/" + i + "/")

 time.sleep(5)

    #scroll down

    #increase the range to sroll more

    #example: range(0,10) scrolls down 650+ images

    for j in range(0,5):

     driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

      time.sleep(10)

#target all the link elements on the page

    anchors = driver.find\_elements\_by\_tag\_name('a')

    anchors = [a.get\_attribute('href') for a in anchors]

    #narrow down all links to image links only

    anchors = [a for a in anchors if str(a).startswith("https://www.facebook.com/photo")]

    print('Found ' + str(len(anchors)) + ' links to images')

   #extract the [1]st image element in each link

    for a in anchors:

    driver.get(a) #navigate to link

     time.sleep(5) #wait a bit

     img = driver.find\_elements\_by\_tag\_name("img")

     images.append(img[1].get\_attribute("src")) #may change in future to img[?]

print('I scraped '+ str(len(images)) + ' images!')

Step 4 :

Extracting all the comments linked with images

def extract\_comments(session, base\_url, post\_bs, post\_url):

comments = list()

show\_more\_url = post\_bs.find('a', href=re.compile('/story\.php\?story'))['href']

 first\_comment\_page = True

 logging.info('Scraping comments from {}'.format(post\_url))

  while True:

 logging.info('[!] Scraping comments.')

 time.sleep(3)

  if first\_comment\_page:

  first\_comment\_page = True

  else:

    post\_bs = get\_bs(session, base\_url+show\_more\_url)

   time. sleep(3)

    try:

    comments\_elements = post\_bs.find('div', id=re.compile('composer')).next\_sibling\ .find\_all('div', id=re.compile('^\d+'))

    except Exception:

     pass

 if len(comments\_elements) != 0:

   logging.info('[!] There are comments.')

        else:

        break

        for comment in comments\_elements:

         comment\_data = OrderedDict()

         comment\_data['text'] = list()

            try:

                comment\_strings = comment.find('h3').next\_sibling.strings

                for string in comment\_strings:

                    comment\_data['text'].append(string)

            except Exception:

                pass

             try:

                media = comment.find('h3').next\_sibling.next\_sibling.children

                if media is not None:

                    for element in media:

                        comment\_data['media\_url'] = element['src']

                else:

                    comment\_data['media\_url'] = ''

            except Exception:

                pass

            comment\_data['text’] = comment.find('h3').a['href'].split('?')[0]

            comments.append(dict(comment\_data))

        show\_more\_url = post\_bs.find('a', href=re.compile('/story\.php\?story'))

        if 'View more' in show\_more\_url.text:

  logging.info('[!] More comments.')

            show\_more\_url = show\_more\_url['href']

        else:

         break

    return comments

def save\_data(data):

    """Converts data to JSON.

    """

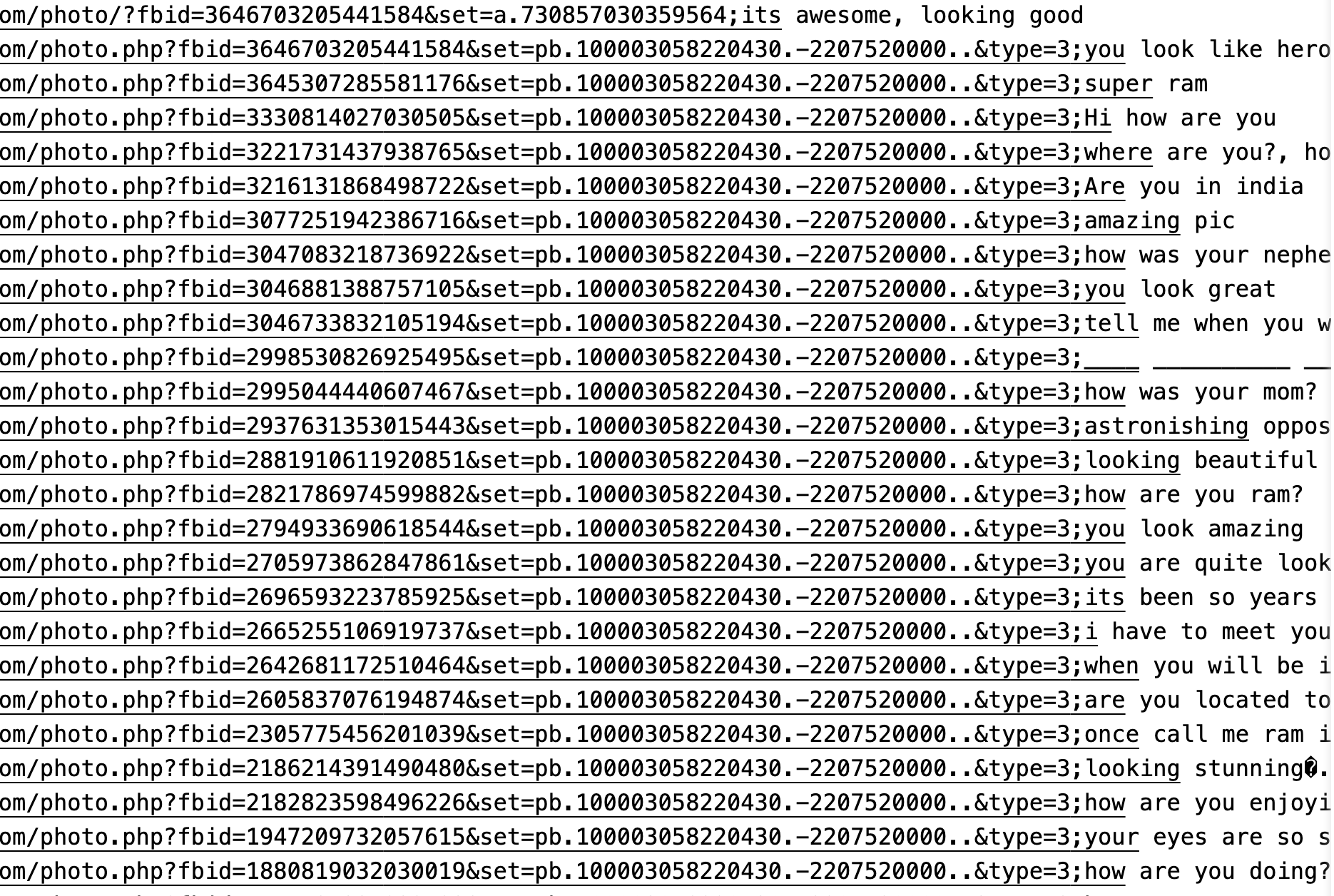
    with open('posts\_data.json', 'w') as json\_file:

        json.dump(data, json\_file, indent=4)

        df1 =  pd.read\_json("posts\_data.json")

        df1.to\_csv("comments.csv")

**Resultant Final Data file :**

****

Step 5:

Storing data file into mongodb

import csv

import pymongo

import json

import pandas as pd

import sys, getopt, pprint

client = pymongo.MongoClient("mongodb://localhost:27017")

df = pd.read\_csv("final.csv")

data = df.to\_dict(orient ="records")

db = client["MachineLearning"]

print(db)

db.MachineLearning.insert\_many(data)