A

Major Project On

## HCOVBI-CAPS : HATE SPEECH DETECTION USING

**CONVOLUTIONAL AND BI- DIRECTIONAL GATED RECURRENT UNIT WITH CAPSULE NETWORK**

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

P.ANOOHYA (207R1A05A9) R.EAKTA GOMATI (207R1A05B2) M.SAI NATH (207R1A05A2)

Under the Guidance of

### Dr.V.NARESH KUMAR

(Associate Professor)



### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**CMR TECHNICAL CAMPUS UGC AUTONOMOUS**

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

**2020-2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



## CERTIFICATE

This is to certify that the project entitled **“HCOVBI-CAPS : HATE SPEECH DETECTION USING CONVOLUTIONAL AND BI- DIRECTIONAL GATED RECURRENT UNIT WITH CAPSULE NETWORK”** being submitted by **P.ANOOHYA (207R1A05A9), R.EAKTA**

**GOMATI (207R1A05B2) & M.SAI NATH (207R1A05A2)** in partial fulfillment of the requirements for theaward of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

**Dr.V.NARESH KUMAR Dr.A. RAJI REDDY**

(Associate Professor) DIRECTOR

INTERNAL GUIDE

**Dr. K. SRUJAN RAJU EXTERNAL EXAMINER**

HOD

**Submitted for viva voice Examination held on**

### ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to thepeople who havebeen instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Dr.V.Naresh Kumar,** Associate Professor for his exemplary guidance, monitoring andconstant encouragement throughout the project work. The blessing, help and guidance given by himshall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to the Project

Review Committee (PRC) **Dr. J. Narasimharao, G.Vinesh Shanker, Ms. Shilpa,& Dr. K. Maheswari** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

#### P. ANOOHYA (207R1A05A9)

**R. EAKTA GOMATI (207R1A05B2)**

#### M. SAI NATH (207R1A05A2)

**ABSTRACT**

Adversaries and anti-social elements have exploited the rapid proliferation of computing technology and online social media in the form of novel security threats, such as fake profiles, hate speech, social bots, and rumors. The hate speech problem on online social networks (OSNs) is also widespread. The existing literature has machinelearning approaches for hate speech detection on OSNs. However, the effectiveness ofcontextual information at different orientations is understudied. This study presents a novel Convolutional, BiGRU, and Capsule network-based deep learning model, HCovBi-Caps, to classify the hate speech. The proposed model is evaluated over two Twitter-based benchmark datasets – DS1(balanced) and DS2(unbalanced) with the best performance of 0:90, 0:80, and 0:84 respectively considering precision, recall, and f- score over unbalanced dataset. In terms of training and validation accuracy, the proposed model shows the best performance of 0:93 and 0:90, respectively, over the unbalanced dataset. In comparative evaluation, HCovBi-Caps demonstrates a significantly better performance than state of-the-art approaches.

In addition, HCovBi-Caps shows comparatively better performance over the unbalanced dataset. We also investigate the impact of different hyperparameters on the efficacy of HCovBi-Caps to ascertain the selection of their values. We observedthat a higher value of routing iterations adversely affects the model performance, whereas a higher value of capsule dimension improves the performance.

### FIGURE NO FIGURE NAME PAGE NO

Figure 3.1

Architecture of HCOVBI-CAPS : Hate

Speech Detection Using Convolutional and 12

Bi- DirectionalGated Recurrent Unit With Capsule Network

Usecase Diagram of HCOVBI-CAPS : Hate

Figure 3.2

Speech Detection Using Convolutional and 14

Bi- Directional Gated Recurrent Unit With CapsuleNetwork

Figure 3.3

Class Diagram of HCOVBI-CAPS : Hate

Speech Detection Using Convolutional and 15

Bi- DirectionalGated Recurrent Unit With Capsule Network

Figure 3.4

Sequence Diagram of HCOVBI-CAPS :

Hate Speech Detection Using Convolutional 16

and Bi- DirectionalGated Recurrent Unit With Capsule Network

Figure 3.5

Data Flow Diagram of HCOVBI-CAPS :

Hate Speech Detection Using Convolutional 17

and Bi- Directional Gated Recurrent Unit With CapsuleNetwork

### SCREENSHOT NO. SCREENSHOT NAME PAGE NO.

|  |  |  |
| --- | --- | --- |
| Screenshot 5.1 | User Register | 34 |
| Screenshot 5.2 | User Login | 34 |
| Screenshot 5.3 | Service Provider Login | 35 |
| Screenshot 5.4 | Details of users registered | 35 |
| Screenshot 5.5 | Hate Speech Prediction | 36 |
| Screenshot 5.6 | Hate Speech Result | 36 |

**ABSTRACT** i

LIST OF FIGURES ii

LIST OF SCREENSHOTS iii

1. [INTRODUCTION 1](#_TOC_250039)
   1. [PROJECT SCOPE 1](#_TOC_250038)
   2. [PROJECT PURPOSE 2](#_TOC_250037)
   3. [PROJECT FEATURES 3](#_TOC_250036)
2. [SYSTEM ANALYSIS 4](#_TOC_250035)
   1. [PROBLEM DEFINITION 5](#_TOC_250034)
   2. [EXISTING SYSTEM 6](#_TOC_250033)
      1. [DISADVANTAGES OF EXISTING SYSTEM 7](#_TOC_250032)
   3. [PROPOSED SYSTEM 7](#_TOC_250031)
      1. ADVANTAGES OF PROPOSED SYSTEM 8
   4. [FEASIBILITY STUDY 8](#_TOC_250030)
      1. [ECONOMIC FEASIBILITY 9](#_TOC_250029)
      2. [TECHNICAL FEASIBILITY 9](#_TOC_250028)
      3. [SOCIAL FEASIBILITY 10](#_TOC_250027)
   5. [HARDWARE & SOFTWARE REQUIREMENTS 10](#_TOC_250026)
      1. [HARDWARE REQUIREMENTS 10](#_TOC_250025)
      2. [SOFTWARE REQUIREMENTS 11](#_TOC_250024)
3. [ARCHITECTURE 12](#_TOC_250023)
   1. [PROJECT ARCHITECTURE 12](#_TOC_250022)
   2. [DESCRIPTION 13](#_TOC_250021)
   3. [USE CASE DIAGRAM 14](#_TOC_250020)
   4. [CLASS DIAGRAM 15](#_TOC_250019)
   5. [SEQUENCE DIAGRAM 16](#_TOC_250018)
   6. [DATA FLOW DIAGRAM 17](#_TOC_250017)
4. [IMPLEMENTATION 18](#_TOC_250016)
   1. [SAMPLE CODE 18](#_TOC_250015)
5. [SCREENSHOTS 34](#_TOC_250014)
6. [TESTING 37](#_TOC_250013)
   1. [INTRODUCTION TO TESTING 37](#_TOC_250012)
   2. [TYPES OF TESTING 37](#_TOC_250011)
      1. [UNIT TESTING 37](#_TOC_250010)
      2. [INTEGRATION TESTING 38](#_TOC_250009)
      3. [FUNCTIONAL TESTING 38](#_TOC_250008)
      4. [WHITE BOX TESTING 39](#_TOC_250007)
      5. [BLACK BOX TESTING 39](#_TOC_250006)
      6. [SYSTEM TESTING 39](#_TOC_250005)
   3. [TEST CASES 40](#_TOC_250004)
      1. CLASSIFICATION 40
7. [CONCLUSION & FUTURE SCOPE 41](#_TOC_250003)
   1. PROJECT CONCLUSION 41
   2. [FUTURE SCOPE 41](#_TOC_250002)
8. REFERENCES 42
   1. [REFERENCES 42](#_TOC_250001)
   2. [GITHUB LINK 43](#_TOC_250000)

# 1. INTRODUCTION

## INTRODUCTION

### PROJECT SCOPE

The convolutional layers are likely effective at capturing local patterns and features,whilethe Bi-Directional Gated Recurrent Unit brings in the ability to understand context and dependencies over longer sequences. The Capsule Network might add another layer of abstraction, helping to model hierarchical relationships among different elements in thedata.

The project's scope probably involves data preprocessing, model training, and evaluation.

#### Data Preprocessing:

* Gather a diverse dataset containing examples of hate speech and non-hate speech.
* Preprocess the data, including cleaning, tokenization, and handling any imbalance in the dataset.

#### Model Training:

* Split the dataset into training, validation, and test sets.
* Train the model using the training set, adjusting hyperparameters as needed.
* Monitor and prevent overfitting through regularization techniques.

#### Evaluation:

* Assess the model's performance on the validation set.
* Fine-tune the model based on validation results.
* Evaluate the final model on the test set to measure its generalization to unseendata.

### PROJECT PURPOSE

The primary purpose of this research project is about Hate speech. Hate speech is asignificant problem in online platforms, impacting individuals and communities. The project aims to contribute to creating safer online spaces by automatically identifying andmitigating hate speech. By developing an effective hate speech detection model, it is working towards fostering inclusive and respectful online conversations. This is particularly crucial in today's digital age where social media plays a central role in communication. The primarypurpose is to protect users from harmful content. Hate speech can lead to real-world consequences, and the project seeks to create a tool that acts as a preventive measure against such harm. Online platforms often struggle with content moderation at scale.

An automated hate speechdetection system can assist in this process,making it more efficient and timely. The combination of Convolutional and Bi-DirectionalGated RecurrentUnit with a Capsule Network represents an innovative approach. It showcases the integration of diverse neural network architectures to enhance the model's ability to understand and identify hate speech. The project also highlights the importance of ethical AI development. Ensuring that the model is fair, unbiased, and interpretable is crucial in the context of hate speech detection, where misclassifications can have severe consequences. Providing users with tools to filter out or report hate speech empowers them to take control of their online experience. It gives them the means to create a more positive and respectful online environment.

### PROJECT FEATURES

The main features of this project will involve several key features. The model incorporates a combination of Convolutional layers, Bi-Directional Gated Recurrent Unit,and Capsule Network. This diverse architecture allows the model to capture various aspectsof language, including local patterns, contextual information, and hierarchical relationships. Bi-Directional Gated Recurrent Unit contributes to the model's ability to understand context in both forward and backward directions. This is crucial for grasping the meaning of words in a sentence and capturing dependencies over longer sequences.

Convolutional layers are designed to identify local patterns and features within the input data. This is particularly effective for capturing specific linguistic cues and expressions associated with hate speech. The Capsule Network introduces a hierarchical feature representation, allowing the model to understand the relationships between different elements in the data. This can enhance the model's ability to recognize complex patterns associated with hate speech.

The use of Bi-Directional Gated Recurrent Unit enables the model to adapt to varyingsequence lengths, making it robust to different lengths of text inputs commonly found in social media platforms. The model may includefeatures or strategies to address ethical considerations, such as minimizing biases and ensuring fairness in the detection of hate speech. The preprocessing stage likely involvesadvanced tokenization techniques to break down text into meaningful units. Embeddings, such as word embeddings or contextual embeddings, may be used to represent words in arich vector space.

# SYSTEM ANALYSIS

## SYSTEM ANALYSIS

### SYSTEM ANALYSIS

**Input Layer:**The system takes text input, typically in the form of sentences or paragraphs, from various online sources.

**Tokenization:** The input text undergoes tokenization to break it down into individual words or sub-word units.

**Embeddings:** Words are represented as embeddings, which could be pre-trained word embeddings or contextual embeddings, capturing semantic relationships.

**Convolutional Layers:** Convolutional layers analyze the input text to detect localpatterns and features, capturing specific linguistic cues associated with hate speech.

**Bi-Directional Gated Recurrent Unit (Bi-GRU):** The Bi-GRU processes the tokenized and embedded input in both forward and backward directions. This helps the model understand the context of words and dependencies over longer sequences.

**Capsule Network:**The Capsule Network introduces a hierarchical feature representation, allowing the model to understand relationships between different elements in the data. This is crucial for capturing complex patterns associated with hate speech.

**Output Layer:** Hate Speech Probability: The integrated features are fed into the output layer, which produces a probability score indicating the likelihood of the input containing hate speech.

### PROBLEM DEFINITION

The problem definition of HCOVBI-CAPS: Hate Speech Detection Using Convolutional and Bi-Directional Gated Recurrent Unit With Capsule Network involves addressing the challenge of automatically identifying and classifying hate speech in textualdata. The primary goal is to develop a robust and accurate hate speech detection model using a combination of Convolutional layers, Bi-Directional Gated Recurrent Unit (Bi- GRU), and Capsule Network. Hate speech detection is a binary classification problem where the model needs to determine whether a given piece of text contains hate speech or not.

The model is trained and evaluated on a diverse dataset that includes examples of hate speech and non-hate speech. The dataset should cover various forms of hate speech found in online platforms. The input data consists of text snippets, such as social media posts, comments, or any form of textual content where hate speech may be present. Hate speech can be nuanced and context-dependent. The model needs to capture both explicit and implicit forms of hate speech, considering variations in language, cultural references, and evolving online communication trends. The problem involves designing a neural network architecture (HCOVBI-CAPS) that leverages Convolutional layers for localpattern recognition, Bi-GRU for contextual understanding, and Capsule Network for hierarchical feature representation.

.

### EXISTING SYSTEM

Warner and Hirschberg used unigram, part of speech, and other template- based features in one of the early approaches to tackle the hate speech problem. Theauthors further trained the SVMlight model using linear kernel and evaluated it overtwo datasets from Yahoo and the American Jews Congress websites toclassify the hate from non-hate content. In another approach, Kwok and Wand usedunigram features and further trained Naive Bayes classifier to segregate the racist tweets from ordinary ones with an accuracy of 76%. They experimentally concludedthat bigram, trigram, and sentiment improve model performance. In another approachbased on n- gram, Burnap and Williams employed various n-gram features and trained three machine learning models: Bayesian logistic regression, SVM, and voted ensemble classifiers.They further evaluated the trained models over the crawled Twitter datasetand reported that voted ensemble classifiers show the best performance. Djuric et al. utilized paragraph2vec language model for the joint modeling of comments and words collected from the Yahoo Financial website.

In a popular approach, Waseem and Hovy open-sourced a benchmarked dataset of 16k tweets containing hate speech. The authors further used 1\_4-gram features to train the logistic regression classifier to segregate the hate and ordinary tweets. The best model shows performance with an F1-score of 73:89. They also used location and gender features and gender with n-gram reports the best performance. The various categories of hateful content, such as hate, offensive, abusive, have subtle differences; however, it is understudied. Davidson et al.

Investigated the difference between hate, abusive, spam, and genuine content andused unigram, bigram, POS tag- based n-grams, Flesch–Kincaid Grade Level, FleschReading Ease scores, sentiment score, and various linguistic features to train logisticregression classifier to segregatethem. Malmasi and Zampieri presented a similar approach using character and word n-gram features to train linear SVM classifier to classify the hate, offensive, andordinary contents.

#### DISADVANTAGES OF EXISTING SYSTEM

Following are the disadvantages of existing system:

* The existing literature has no universally accepted definition of hate speech, and even OSNs do not have a consensus.
* An existing system doesn’t Filtering of Twitter-related markers and symbols, such as hashtags, URLs, mentions, and retweets.

### PROPOSED SYSTEM

* Introduce a novel deep neural network model, HCovBi-Caps, by integrating the BiGRU, Convolutional layer and Capsule network to incorporate the contextual information at different orientations for hate speech detection.
* Perform the comparative evaluation of HCovBi-Caps over two benchmark datasets to establish its efficacy.
* Investigate the impact of different hyper-parameters values on the efficacy of HCovBi- Caps performance to observe the best hyper-parameters values.

#### 2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

* The HCovBi-Caps applies the convolutional layer over the embedding vector to extract the spatial features. The proposed model uses the one-dimensional convolutional operation because the input embedding vector is a row vector.
* HCovBi-Caps is a best contribution to the collaborative effort going around the world to eradicate the hateful and anti-social content from OSNs.

### FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis:

* Economic Feasibility
* Technical Feasibility
* Social Feasibility

#### ECONOMIC FEASIBILITY

This study is carried out to check the economic impact that the system will have onthe organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

#### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed systemmust have a modest requirement, as only minimal or null changes are required for implementing this system.

#### SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

### HARDWARE & SOFTWARE REQUIREMENTS

#### HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

* + Processor : Pentium –IV
  + RAM : 4 GB or more
  + Hard Disk : 20 GB or more

#### SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

* + Operating system : Windows 7 Ultimate or later.
  + Coding Language : Python.
  + Front-End : Python.
  + Back-End : Django-ORM
  + Designing : Html, css, javascript.
  + Data Base **:** MySQL (WAMP Server).

# ARCHITECTURE

## ARCHITECTURE:

### PROJECT ARCHITECTURE:

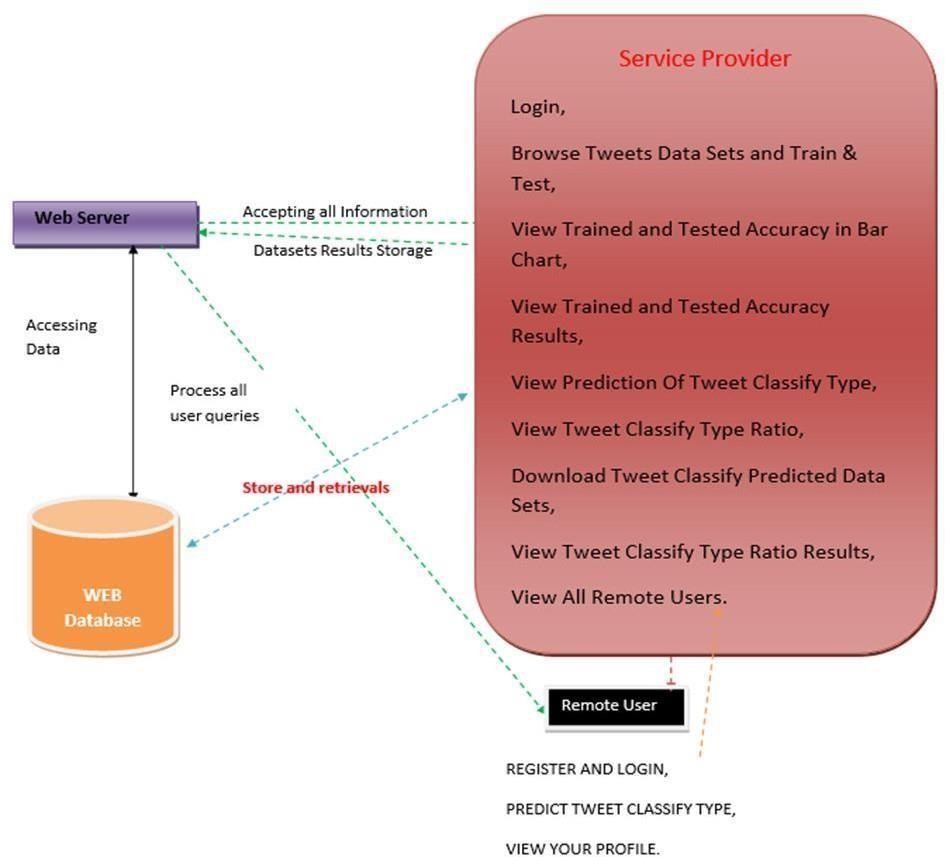


Figure 1: Architecture of HCOVBI-CAPS : Hate Speech Detection Using Convolutional and Bi- Directional Gated Recurrent Unit With Capsule

### DESCRIPTION:

HCOVBI-CAPS is a hate speech detection model that utilizes a combination of Convolutional Neural Networks (CNNs), Bi-directional Gated Recurrent Units (Bi- GRUs), and Capsule Networks (CapsNets) architecture.

CNNs are widely used in image processing tasks but have also proven effective in natural language processing (NLP) tasks, particularly for text classification. They are adept at capturing local patterns and features within the input data. In the context of hate speech detection, CNNs can extract relevant features from textual data, such as word embeddings or character-level embeddings.

GRUs are a type of recurrent neural network (RNN) that are efficient in capturing sequential patterns in data. By utilizing a bi-directional architecture, information from both past and future context of a word or a sequence can be incorporated into the model's representation. This helps in understanding the context and semantics of the text, which is crucial in identifying hate speech.

CapsNets are a relatively new architecture in deep learning, inspired by the human visual system. They aim to overcome some limitations of traditional CNNs, particularly in handling hierarchical relationships among features. CapsNets excel in capturing spatial hierarchies and pose relationships within visual data. In the context of hate speech detection, CapsNets can potentially learn hierarchical representations of text, capturing nuanced relationships between words or phrases.

### USE CASE DIAGRAM

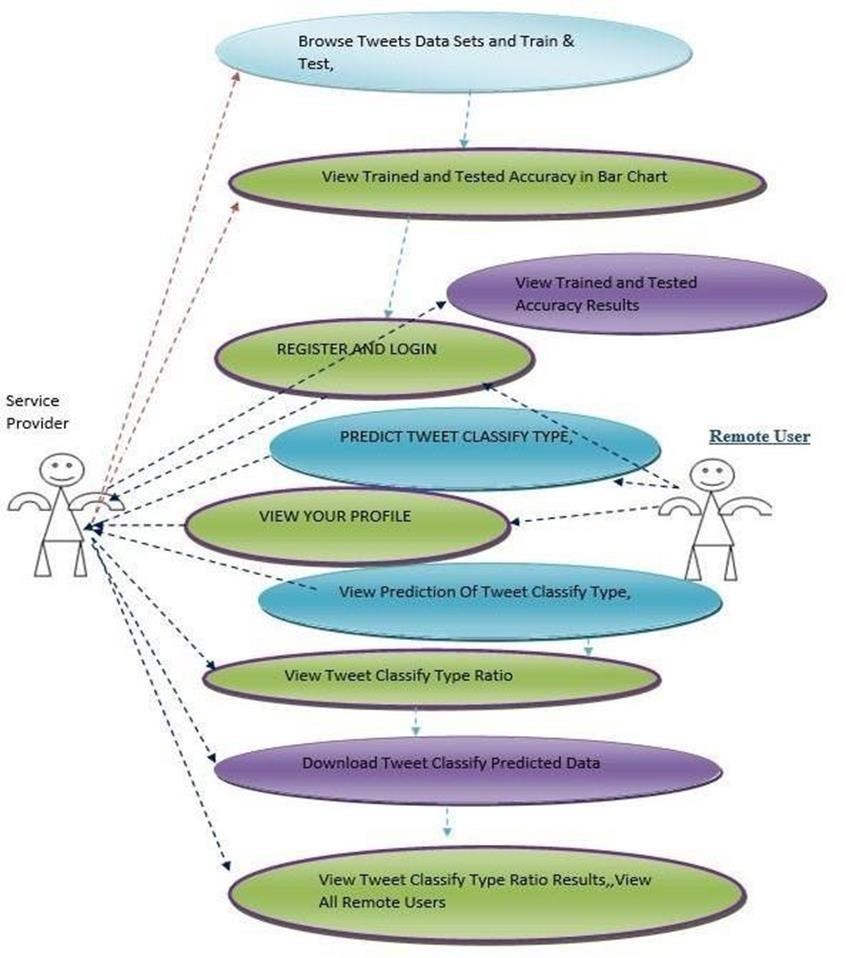
In the use case diagram, we have basically one actor who is the user in the trained model. A use case diagram is a graphical depiction of a user's possible interactions with asystem. A use case diagram shows various use cases and different types of usersthe systemhas. The use cases are represented by either circles or ellipses. The actors are often shownas stick figures.

Figure 3.2: Use Case Diagram for HCOVBI-CAPS : Hate Speech Detection Using Convolutional and Bi- Directional Gated Recurrent Unit With Capsule Network

### CLASS DIAGRAM

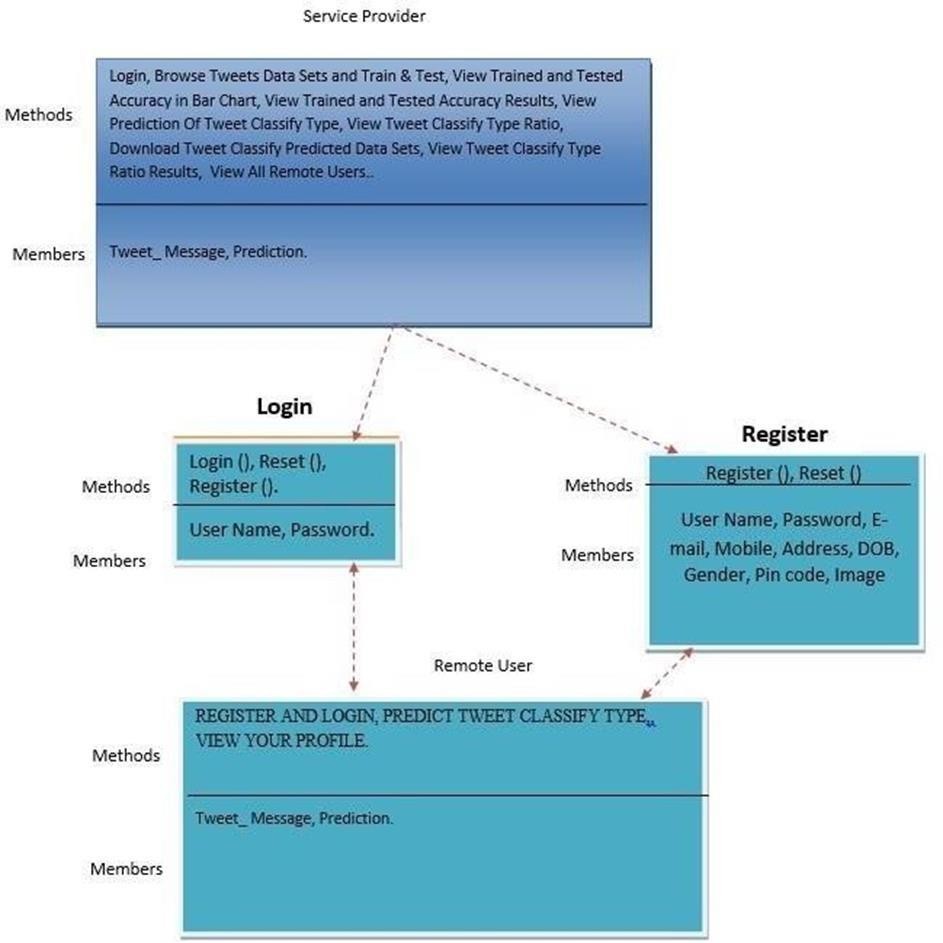
Class diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among objects.

Figure 3.3: Class Diagram for HCOVBI-CAPS : Hate Speech Detection Using Convolutional and Bi- Directional Gated Recurrent Unit With Capsule Network

### SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depictsthe objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

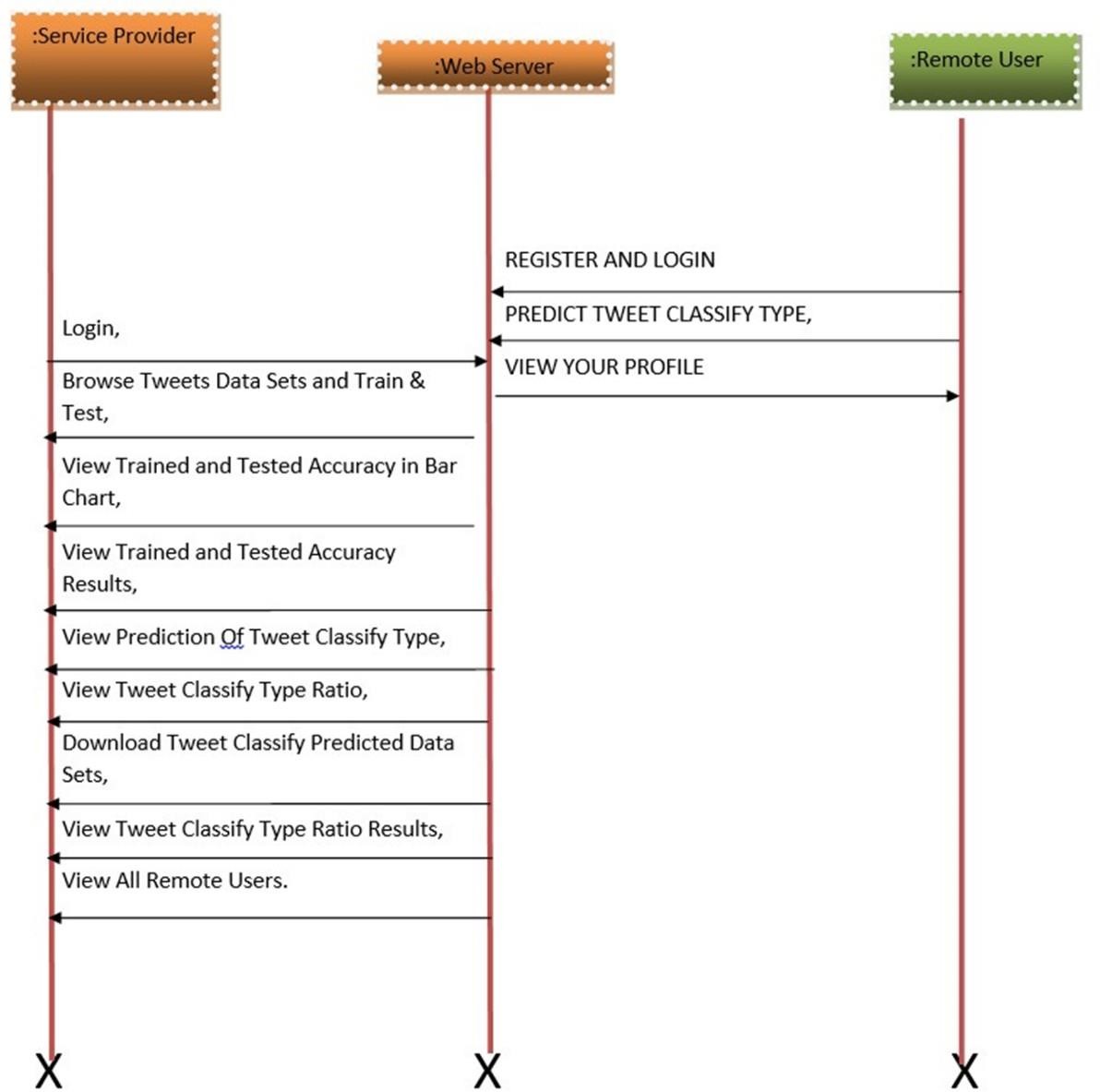


Figure 3.4: Sequence Diagram for HCOVBI-CAPS : Hate Speech Detection Using Convolutional and Bi- Directional Gated Recurrent Unit With Capsule Network

### DATA FLOW DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activitiesand actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more datastores.

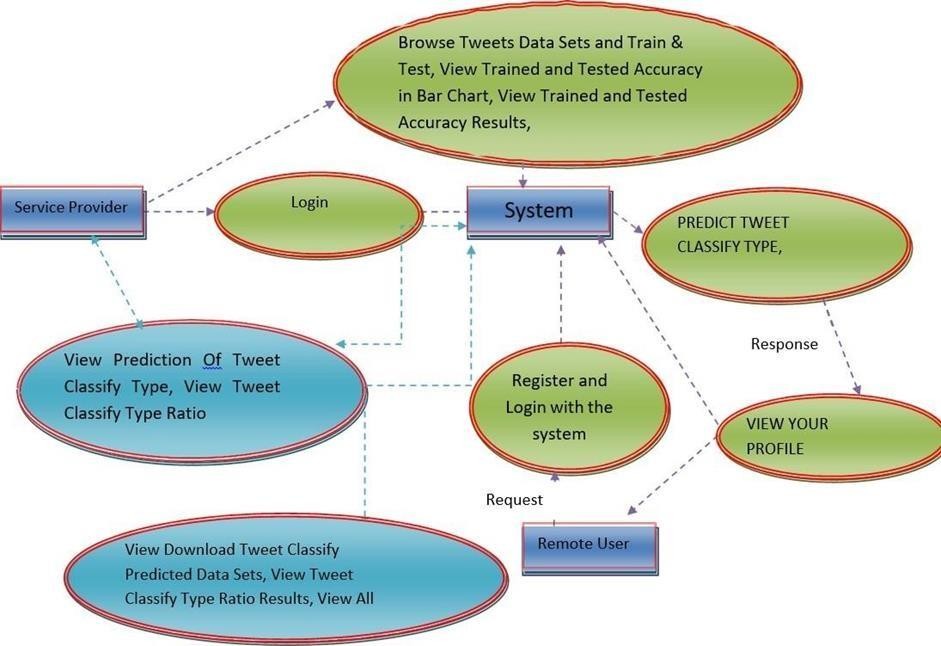


Figure 3.5: Activity Diagram for HCOVBI-CAPS : Hate Speech Detection Using Convolutional and Bi- Directional Gated Recurrent Unit With Capsule Network

# IMPLEMENTATION

## IMPLEMENTATION

### SAMPLE CODE

import os import sys import nltk import re

import pandas as pd import string import datetime import xlwt

import numpy as np import datetime import openpyxl

from django.db.models import Count from django.db.models import Q

from django.shortcuts import render, redirect, get\_object\_or\_404 from django.contrib import admin

from django.apps import AppConfig from django.db import models

from django.test import TestCase from nltk.corpus import stopwords

from sklearn.feature\_extraction.text import CountVectorizer from nltk.stem.wordnet import WordNetLemmatizer

from sklearn.ensemble import RandomForestClassifier from wordcloud import WordCloud, STOPWORDS

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

from sklearn.tree import DecisionTreeClassifier from sklearn.ensemble import VotingClassifier

from Remote\_User.models import ClientRegister\_Model,hate\_classify\_prediction,detection\_ratio,detection\_accuracy from django.test import TestCase

from django.apps import AppConfig from django.contrib import admin

from django.db.models import Count, Avg from django.shortcuts import render, redirect from django.db.models import Count

from django.db.models import Q

from django.http import HttpResponse from nltk.corpus import stopwords

from sklearn.feature\_extraction.text import CountVectorizer from nltk.stem.wordnet import WordNetLemmatizer import pandas as pd

from wordcloud import WordCloud, STOPWORDS

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report from sklearn.metrics import accuracy\_score

from sklearn.metrics import f1\_score

from sklearn.tree import DecisionTreeClassifier from sklearn.ensemble import VotingClassifier

from sklearn.ensemble import RandomForestClassifier from Remote\_User.models import

ClientRegister\_Model,hate\_classify\_prediction,detection\_ratio,detection\_accuracy from django import forms

from Remote\_User.models import ClientRegister\_Model

class ClientSiteConfig(AppConfig): name = 'Remote\_User'

class ClientRegister\_Form(forms.ModelForm):

password = forms.CharField(widget=forms.PasswordInput()) email = forms.EmailField(required=True)

class Meta:

model = ClientRegister\_Model

fields = ("username","email","password","phoneno","country","state","city")

class ClientRegister\_Model(models.Model): username = models.CharField(max\_length=30) email = models.EmailField(max\_length=30) password = models.CharField(max\_length=10) phoneno = models.CharField(max\_length=10) country = models.CharField(max\_length=30) state = models.CharField(max\_length=30)

city = models.CharField(max\_length=30)

class hate\_classify\_prediction(models.Model): Tweet\_Message= models.CharField(max\_length=30000) Prediction= models.CharField(max\_length=300)

class detection\_accuracy(models.Model): names = models.CharField(max\_length=300) ratio = models.CharField(max\_length=300)

class detection\_ratio(models.Model):

names = models.CharField(max\_length=300) ratio = models.CharField(max\_length=300)

class ResearchSiteConfig(AppConfig): name = 'Service\_Provider'

def login(request):

if request.method == "POST" and 'submit1' in request.POST: username = request.POST.get('username')

password = request.POST.get('password')

try:

enter = ClientRegister\_Model.objects.get(username=username,password=password) request.session["userid"] = enter.id

return redirect('ViewYourProfile') except:

pass

return render(request,'RUser/login.html') def Add\_DataSet\_Details(request):

return render(request, 'RUser/Add\_DataSet\_Details.html', {"excel\_data": ''})

def Register1(request):

if request.method == "POST":

username = request.POST.get('username') email = request.POST.get('email') password = request.POST.get('password') phoneno = request.POST.get('phoneno') country = request.POST.get('country') state = request.POST.get('state')

city = request.POST.get('city') ClientRegister\_Model.objects.create(username=username, email=email,

password=password, phoneno=phoneno,

country=country, state=state, city=city)

return render(request, 'RUser/Register1.html') else:

return render(request,'RUser/Register1.html')

def ViewYourProfile(request): userid = request.session['userid']

obj = ClientRegister\_Model.objects.get(id= userid)

return render(request,'RUser/ViewYourProfile.html',{'object':obj})

def Predict\_TweetClassify\_Type(request): if request.method == "POST":

Tweet\_Message = request.POST.get('Tweet\_Message') if request.method == "POST":

Tweet\_Message = request.POST.get('Tweet\_Message') dataset = pd.read\_csv('Tweet\_Datasets.csv')

# Preprocess Data

def process\_tweet(tweet):

return " ".join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])", " ", tweet.lower()).split())

dataset.rename(columns={'class': 'label', 'tweet': 'review'}, inplace=True) def apply\_results(label):

if (label == 0):

return 0 # Hate Speech elif (label == 1):

return 1 # Offensive Speech elif (label == 2):

return 2 # Non Offensive Speech

dataset['results'] = dataset['label'].apply(apply\_results) dataset.drop(['label'], axis=1, inplace=True)

results = dataset['results'].value\_counts() cv = CountVectorizer()

dataset["review"] = dataset['review'].apply(process\_tweet) x = dataset["review"]

y = dataset["results"]

x = cv.fit\_transform(x) print("Tweet") print(x)

print("Label") print(y)

models = []

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.20)

X\_train.shape, X\_test.shape, y\_train.shape

print("Naive Bayes")

from sklearn.naive\_bayes import MultinomialNB NB = MultinomialNB()

NB.fit(X\_train, y\_train) predict\_nb = NB.predict(X\_test)

naivebayes = accuracy\_score(y\_test, predict\_nb) \* 100 print(naivebayes)

print(confusion\_matrix(y\_test, predict\_nb)) print(classification\_report(y\_test, predict\_nb)) models.append(('naive\_bayes', NB))

# SVM Model print("SVM")

from sklearn import svm lin\_clf = svm.LinearSVC() lin\_clf.fit(X\_train, y\_train)

predict\_svm = lin\_clf.predict(X\_test)

svm\_acc = accuracy\_score(y\_test, predict\_svm) \* 100 print(svm\_acc)

print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, predict\_svm)) print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, predict\_svm)) models.append(('svm', lin\_clf))

print("Logistic Regression")

from sklearn.linear\_model import LogisticRegression

reg = LogisticRegression(random\_state=0, solver='lbfgs').fit(X\_train, y\_train) y\_pred = reg.predict(X\_test)

print("ACCURACY")

print(accuracy\_score(y\_test, y\_pred) \* 100) print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, y\_pred)) print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, y\_pred)) models.append(('logistic', reg))

print("Decision Tree Classifier") dtc = DecisionTreeClassifier() dtc.fit(X\_train, y\_train) dtcpredict = dtc.predict(X\_test) print("ACCURACY")

print(accuracy\_score(y\_test, dtcpredict) \* 100) print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, dtcpredict)) print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, dtcpredict))

classifier = VotingClassifier(models) classifier.fit(X\_train, y\_train) y\_pred = classifier.predict(X\_test) tweet\_data = [Tweet\_Message]

vector1 = cv.transform(tweet\_data).toarray() predict\_text = classifier.predict(vector1) pred = str(predict\_text).replace("[", "") pred1 = pred.replace("]", "")

prediction = int(pred1) if prediction == 0:

val = 'Hate Speech' elif prediction == 1:

val = 'Offensive Speech' elif prediction == 2:

val = 'Non Offensive Speech' print(val)

print(pred1)

hate\_classify\_prediction.objects.create(Tweet\_Message=Tweet\_Message, Prediction=val) return render(request, 'RUser/Predict\_TweetClassify\_Type.html',{'objs': val})

return render(request, 'RUser/Predict\_TweetClassify\_Type.html')

def serviceproviderlogin(request): if request.method == "POST":

admin = request.POST.get('username') password = request.POST.get('password')

if admin == "Admin" and password =="Admin": detection\_accuracy.objects.all().delete() return redirect('View\_Remote\_Users')

return render(request,'SProvider/serviceproviderlogin.html')

def View\_TweetClassify\_Type\_Ratio(request): detection\_ratio.objects.all().delete()

rratio = ""

kword = 'Hate Speech' print(kword)

obj = hate\_classify\_prediction.objects.all().filter(Q(Prediction=kword)) obj1 = hate\_classify\_prediction.objects.all()

count = obj.count(); count1 = obj1.count();

ratio = (count / count1) \* 100 if ratio != 0:

detection\_ratio.objects.create(names=kword, ratio=ratio)

ratio1 = ""

kword1 = 'Offensive Speech' print(kword1)

obj1 = hate\_classify\_prediction.objects.all().filter(Q(Prediction=kword1)) obj11 = hate\_classify\_prediction.objects.all()

count1 = obj1.count(); count11 = obj11.count();

ratio1 = (count1 / count11) \* 100 if ratio1 != 0:

detection\_ratio.objects.create(names=kword1, ratio=ratio1)

ratio1a = ""

kword1a = 'Non Offensive Speech' print(kword1a)

obj1a = hate\_classify\_prediction.objects.all().filter(Q(Prediction=kword1)) obj11a = hate\_classify\_prediction.objects.all()

count1a = obj1a.count(); count11a = obj11a.count();

ratio1a = (count1a / count11a) \* 100

if ratio1a != 0:

detection\_ratio.objects.create(names=kword1a, ratio=ratio1a)

obj = detection\_ratio.objects.all()

return render(request, 'SProvider/View\_TweetClassify\_Type\_Ratio.html', {'objs': obj})

def View\_Remote\_Users(request): obj=ClientRegister\_Model.objects.all()

return render(request,'SProvider/View\_Remote\_Users.html',{'objects':obj})

def ViewTrendings(request): topic =

hate\_classify\_prediction.objects.values('topics').annotate(dcount=Count('topics')).order\_by('- dcount')

return render(request,'SProvider/ViewTrendings.html',{'objects':topic})

def charts(request,chart\_type):

chart1 = detection\_ratio.objects.values('names').annotate(dcount=Avg('ratio'))

return render(request,"SProvider/charts.html", {'form':chart1, 'chart\_type':chart\_type})

def charts1(request,chart\_type):

chart1 = detection\_accuracy.objects.values('names').annotate(dcount=Avg('ratio')) return render(request,"SProvider/charts1.html", {'form':chart1, 'chart\_type':chart\_type})

def View\_Prediction\_Of\_TweetClassify\_Type(request): obj =hate\_classify\_prediction.objects.all()

return render(request, 'SProvider/View\_Prediction\_Of\_TweetClassify\_Type.html',

{'list\_objects': obj})

def likeschart(request,like\_chart):

charts =detection\_accuracy.objects.values('names').annotate(dcount=Avg('ratio')) return render(request,"SProvider/likeschart.html", {'form':charts, 'like\_chart':like\_chart})

def Download\_Trained\_DataSets(request):

response = HttpResponse(content\_type='application/ms-excel') # decide file name

response['Content-Disposition'] = 'attachment; filename="Predicted\_Data.xls"' # creating workbook

wb = xlwt.Workbook(encoding='utf-8') # adding sheet

ws = wb.add\_sheet("sheet1") # Sheet header, first row row\_num = 0

font\_style = xlwt.XFStyle() # headers are bold font\_style.font.bold = True

# writer = csv.writer(response)

obj = hate\_classify\_prediction.objects.all() data = obj # dummy method to fetch data. for my\_row in data:

row\_num = row\_num + 1

ws.write(row\_num, 0, my\_row.Tweet\_Message, font\_style) ws.write(row\_num, 1, my\_row.Prediction, font\_style)

wb.save(response) return response

def train\_model(request): detection\_accuracy.objects.all().delete() dataset = pd.read\_csv('Tweet\_Datasets.csv')

# Preprocess Data

def process\_tweet(tweet):

return " ".join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])", " ", tweet.lower()).split()) dataset.rename(columns={'class': 'label', 'tweet': 'review'}, inplace=True)

def apply\_results(label): if (label == 0):

return 0 # Hate Speech elif (label == 1):

return 1 # Offensive Speech elif (label == 2):

return 2 # Non Offensive Speech

dataset['results'] = dataset['label'].apply(apply\_results) dataset.drop(['label'], axis=1, inplace=True)

results = dataset['results'].value\_counts() cv = CountVectorizer()

dataset["review"] = dataset['review'].apply(process\_tweet) x = dataset["review"]

y = dataset["results"]

cv = CountVectorizer(lowercase=False, strip\_accents='unicode', ngram\_range=(1, 1)) print(x)

print("Y") print(y)

x = cv.fit\_transform(x)

models = []

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.20)

X\_train.shape, X\_test.shape, y\_train.shape

print("Naive Bayes")

from sklearn.naive\_bayes import MultinomialNB NB = MultinomialNB()

NB.fit(X\_train, y\_train) predict\_nb = NB.predict(X\_test)

naivebayes = accuracy\_score(y\_test, predict\_nb) \* 100 print("ACCURACY")

print(naivebayes) print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, predict\_nb)) print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, predict\_nb)) detection\_accuracy.objects.create(names="Naive Bayes", ratio=naivebayes)

# SVM Model print("SVM")

from sklearn import svm

lin\_clf = svm.LinearSVC() lin\_clf.fit(X\_train, y\_train) predict\_svm = lin\_clf.predict(X\_test)

svm\_acc = accuracy\_score(y\_test, predict\_svm) \* 100

print("ACCURACY")

print(svm\_acc) print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, predict\_svm)) print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, predict\_svm)) detection\_accuracy.objects.create(names="SVM", ratio=svm\_acc)

print("Logistic Regression")

from sklearn.linear\_model import LogisticRegression

reg = LogisticRegression(random\_state=0, solver='lbfgs').fit(X\_train, y\_train) y\_pred = reg.predict(X\_test)

print("ACCURACY")

print(accuracy\_score(y\_test, y\_pred) \* 100) print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, y\_pred)) print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, y\_pred))

detection\_accuracy.objects.create(names="Logistic Regression", ratio=accuracy\_score(y\_test, y\_pred) \* 100)

print("Decision Tree Classifier") dtc = DecisionTreeClassifier() dtc.fit(X\_train, y\_train) dtcpredict = dtc.predict(X\_test) print("ACCURACY")

print(accuracy\_score(y\_test, dtcpredict) \* 100) print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, dtcpredict))

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, dtcpredict)) detection\_accuracy.objects.create(names="Decision Tree Classifier",

ratio=accuracy\_score(y\_test, dtcpredict) \* 100)

print("SGD Classifier")

from sklearn.linear\_model import SGDClassifier

sgd\_clf = SGDClassifier(loss='hinge', penalty='l2', random\_state=0) sgd\_clf.fit(X\_train, y\_train)

sgdpredict = sgd\_clf.predict(X\_test) print("ACCURACY")

print(accuracy\_score(y\_test, sgdpredict) \* 100) print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, sgdpredict)) print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, sgdpredict)) detection\_accuracy.objects.create(names="SGD Classifier", ratio=accuracy\_score(y\_test,

sgdpredict) \* 100)

labeled = 'Processed\_data.csv' dataset.to\_csv(labeled, index=False) dataset.to\_markdown

obj = detection\_accuracy.objects.all()

return render(request,'SProvider/train\_model.html', {'objs': obj})

def main():

"""Run administrative tasks.""" os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'hate\_classify.settings') try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc: raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you " "forget to activate a virtual environment?"

) from exc execute\_from\_command\_line(sys.argv)

if name == ' main ': main()

# SCREENSHOTS



Screenshot 5.1: User Login



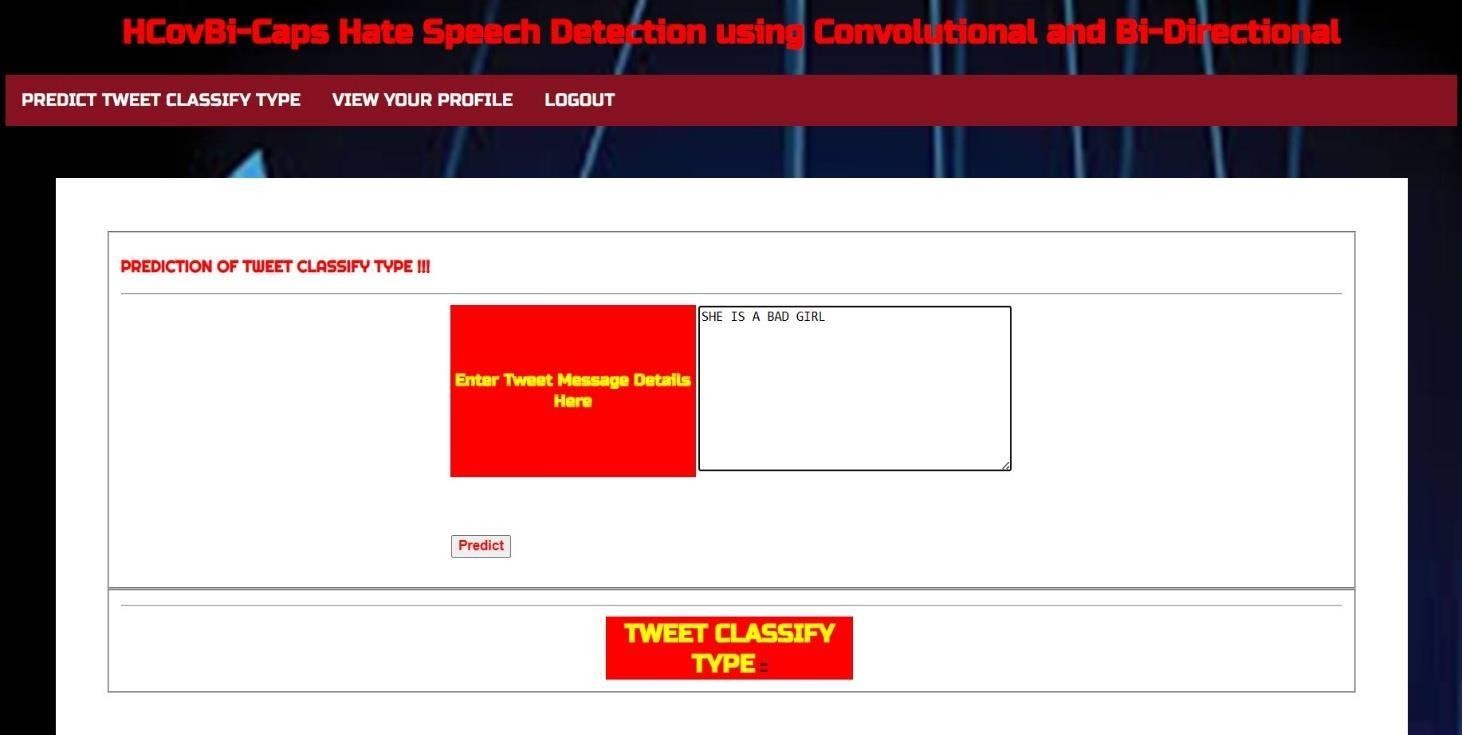
Screenshot 5.2: User Register



Screenshot 5.3: Service Provider Login



Screenshot 5.4: Details of users registered



Screenshot 5.5: Hate Speech Prediction



Screenshot 5.6: Hate Speech Result

# TESTING

## TESTING

### INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

### TYPES OF TESTING

#### UNIT TESTING

Unit testing involves the design of test cases that validate that the internal programlogic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit testsensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

#### INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine ifthey actually run as one program. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

#### FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input

: identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputsmust be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases.

#### WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of theinner workings, structure and language ofthe software, or at least its purpose. It is purpose.It is used to test areas that cannot be reached from a black box level.

#### BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

#### SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An exampleof system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### TEST CASES

**6.3.1 CLASSIFICATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Test case name | Purpose | Input | Output |
| 1. | User Register | If User registration successfully. | Pass | If already user email exist then it fails. |
| 2. | User Login | If Username and password is correct then it will get  valid page. | Pass | Unregistered Users will not logged in. |
| 3. | Service Provider Login | If registration successfully. | Pass | If entered wrong credentials then it fails. |
| 4. | Service Provider View User | Show our dataset | Pass | If Data set Not Available fail. |
| 5. | Admin can activate the register users | Admin can activate the register user id | Pass | If user id not found then it won’t login |
| 6. | Results | For our SVM models accuracy | Pass | If Accuracy Not Displayed fail |

# CONCLUSION

## CONCLUSION & FUTURE SCOPE

### CONCLUSION

In this Project, we presented a service framework called Hate Classify for hate speech detection on social media. The Hate Classify framework employs a crowed-sourcedapproach that permits the social media users to vote about any textual speech or content that is deemed inappropriate. To evaluate the performance in terms of classification, the CNNs were employedand experimental results demonstrate that the classification accuracy achieved through the CNN models, particularly the SCNN is significantly competitiveand even better than several state-of-the-art approaches. An important contribution of this article is that it presents the problem of hate speech classification as the multi label classification problem. The experimental results attained by employing the CNN approaches both for the multiclass classification and multi label classification are sufficiently encouraging and signify the feasibility of these approaches for hate speech classification on social media.

### FUTURE SCOPE

The proposed HCovBi-Caps model detects hateful content with different contextual orientations. However, we can further improve it in detecting hate content considering different contextual semantic. Further, HCovBi-Caps does not exploit the sentiment and users’ profile-related features, which may be effective. We can also evaluate HCovBi-Caps over more diverse datasets. The HCovBi-Caps model detects the hate propagated in text only. Therefore, it can be extended to a multi-model approach for hate speech detection. The extension of HCovBi-Caps to classify the hateful multi-lingual and code-mixed content is also another direction of research. The contextual information, which triggers controversy and hates on OSNs, will also be investigated.

# BIBLIOGRAPHY

## BIBLIOGRAPHY

### REFERENCES

1. F. Del Vigna, A. Cimino, F. Dell’Orletta, M. Petrocchi, andM. Tesconi, “Hate me, hate me not: Hate speech detection on Facebook,” in Proc. 1st Italian Conf.Cybersecurity, 2017, pp. 86–95.
2. T. Davidson, D. Warmsley, M. Macy, and I.Weber,“Automated hate speech detection and the problem of offensive language,” in Proc. 11th Int. AAAI Conf. Web Social Media, 2017, pp. 512–515.
3. W. Warner and J. Hirschberg, “Detecting hate speech on the world wide web,” in Proc. 2nd Workshop Lang. Social Media, 2012, pp. 19–26.
4. Y. Chen, Y. Zhou, S. Zhu, and H. Xu, “Detecting offensive language in social media to protect adolescent online safety,” in Proc. IEEE Int. Conf. Privacy, Secur., Risk Trust Int. Conf. Soc. Comput., 2012,pp. 71–80.
5. H. Hosseinmardi, S. A.Mattson, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra, “Analyzing labeled cyberbullying incidents on the instagram social network,” Social Inform., T. Y. Liu, C. N. Scollon, andW. Zhu, Eds., 2015, pp. 49–66.
6. Z. Waseem and D. Hovy, “Hateful symbols or hateful people? predictive features for hate speech detection on Twitter,” in Proc. NAACL Student Res. Workshop, 2016, pp. 88–93.
7. C. VanHee et al., “Detection and fine-grained classification of cyberbullying events,” in Proc. Int. Conf. Recent Adv. Natural Lang. Process., 2015,pp. 672–680.
8. J.-M. Xu, K.-S. Jun, X. Zhu, and A. Bellmore, “Learning from bullying traces in social media,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol., 2012, pp. 656–666.
9. G. Xiang, B. Fan, L. Wang, J. Hong, and C. Rose, “Detecting offensive tweets via topical feature discovery over a large scale Twitter corpus,” in Proc. 21st ACM Int. Conf. Inf. Knowl. Manage., 2012, pp. 1980–1984.
10. H. Zhong et al., “Content-driven detection of cyberbullying on the Instagram social network,” in Proc. Int. Joint Conf. Artif. Intell., 2016, pp. 3952–3958.
11. N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, “Hate speech detection with comment embeddings,” in Proc. 24th Int. Conf. World Wide Web, 2015, pp. 29–30.
12. P. Burnap and M. L. Williams, “Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making,” Policy Internet, vol. 7, no. 2, pp. 223–242, 2015.
13. C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, “Abusive language detection in online user content,” in Proc. 25th Int. Conf. World Wide Web, 2016, pp. 145–153.
14. Y. Mehdad and J. Tetreault, “Do characters abuse more than words?” in Proc. 17th Annu. Meeting Special Interest Group Discourse Dialogue, 2016, pp. 299–303.
15. Y. Kim, “Convolutional neural networks for sentence classification,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2014, pp. 1746–1751.
16. W. Yin and H. Sch€utze, “Attentive convolution: Equipping CNNs with RNN- style attention mechanisms,” Trans. Assoc. Comput. Linguistics, vol. 6, pp. 687–702, 2018.
17. M. S. Sorower, “A literature survey on algorithms for multi-label learning,” Oregon State Univ., Corvallis, vol. 18, pp. 1–25, 2010

### GITHUB LINK

<https://github.com/Anoohyapasala/Hate-Speech-Detection>