IMPUDENCE DETECTION IN ONLINE FORUMS

A MINI PROJECT REPORT SUBMITTED IN FULLFILLMENT OF THE

REQUIREMENTS FOR THE COMPLETION OF MACHINE LEARNING.

(FIELD: MACHINE LEARNING)

Hari Mitlapati(N150147), Mithipati Harshitha(N150161), Nadakuditi Prathyusha(N150374), Goriparthi Vijayalakshmi(N150935).

(DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING)

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, Nuzvid- 521202

ANDHRA PRADESH, KRISHNA, INDIA.

CERTIFICATE

IT IS CERIFIED THAT THE WORK CONTAINED IN THIS MINI PROJECT ENTITLED ‘IMPUDENCE DETECTION IN ONLINE FORUMS’ BY ‘**Hari Mitlapati(N150147), Mithipati Harshitha(N150161), Nadakuditi Prathyusha(N150374), Goriparthi Vijayalakshmi(N150935).** HAVE BEEN CARRIED OUT UNDER OUR SUPERVISION.

,

PROJECT GUIDE

N.RAMA KRISHNA,

FACULTY,

RGUKT- NUZVID

INDEX

I. ABSTRACT

The aim of this project is to detect the comments that are considered as insulting to other participants of blog/forum conversation. For base word conversion we used lemmatization technique, text cleaning, for feature extraction tf-idf is used and for feature selection, we use CHI-SQUARE and we proposed this new pattern for impudence detection with the use of supervised learning like logistic regression and SVM for training and testing. Evaluation on the insult dataset gives considerable accuracy of 74 percent.

1. NTRODUCTION

1.1. WHAT IS IMPUDENCE DETECTION

Now days, most people are connected across the world through online blogs, news groups and social network sites to express their opinions, share and receive knowledge. This communication involves people from many different cultures, communities and different parts of world. However, sometimes people may use language that is considered inappropriate by others and may hurt others feelings. Besides, it may lead to frustration among the users searching for particular information on some specific site because some people take it as fun to use personal attacking and insulting messages. While the Terms of Service for few sites prohibit posting content that is unlawful, abusive and harassing, users' posts are only partially filtered for some particular collection of offensive words. Also, while some other provide users with the facility to mark a comment as insulting/ inappropriate, they are prone to collusion and are highly misused. There is no existing classifier that identifies insult speech directed towards a participant of the conversation. We aim at building such a classifier in this project**.**

1.2. MOTIVATION

Coming to the negative content in the social platforms always hurts the users' feelings and majorly effects the new user participation. As, there exists many social platforms now a days, people tend to use all the socializing websites that results in excessive increase of data which makes difficult to the humans to handle manually. This is the dominant reason to develop the insult detection models.

2. LITERATURE SURVEY

2.1. OVER VIEW OF THE PROJECT

In the present days people are using various online forums, blogs, social networking sites, newsgroups as source of their networking, sharing and receiving knowledge. Sometimes, some users may make comments that are considered as unpleasant by other users and may hurt their feelings. Besides, it is a barrier to user participant and prevents new users from participating. Also, sometimes you are looking for some information on some site and find insults then it leads to frustration. So, to avoid such incidents a proper interface that screens the offensive comments before posting in the social media platforms is very much needed. Also, it is not possible to have a human moderator to review the comments before posting because of the increasing amount of online data. Hence, we need an automatic classifier that will detect the insulting comments. Our aim is to focus on comments that are insulting to other participants of the blog/ forum conversation. However, the comments containing insults but are targeted to a non-participant of conversation (like a celebrity etc.) are not marked as insults. Insults are of many types like: Taunts, squalid language, slurs and racism which are aimed at attacking the other person. However, some insults are aimed at abuse or embarrassing the reader (not an attack) like crude language, disguise provocative words, disguise, sarcasm, innuendo (indirect reference). We are aimed at detecting comments that are intended to be obviously insulting to other participants and when the comment is detected to be an insult then the user is not allowed to comment, else the user is allowed to post the comment without any restriction

We obtained our annotated dataset from the social media websites. The training dataset consists of 10,000 comments which are labelled as positive and negative. This problem is treated as binary classification problem and various supervised learning tools like Logistic regression, SVM have been used. The implementations have been done using Scikit learn Toolkit which is a natural language toolkit in Python.

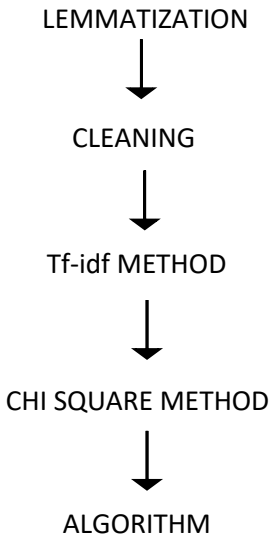
3. ANALYSIS

3.1. EXISTING SYSTEM

Various attempts have been made to classify the insulting comments in online forums, In the existed system of insult detection**,** there exists lot of methods yet it produces an outcome with 65 % accuracy. the methods used are Removal of unwanted strings, Stemming, Correction of some common words, Tokenizing, Counting, Tf-idf Score, Skip grams, Second-person feature, especially there exists some false positives and negatives in the previous model.

3.2. PROPOSED SYSTEM

The proposed detection model aims to validate the comments and segregate into either offensive or pleasant. Here is the basic philosophy of the work. The following sub-sections discuss each step in-detail.



NORMALIZATION

The raw data obtained from internet can’t be used directly for learning models. Some pre-processing of data is necessary and we need to convert it to the input format of various machine learning algorithms. However, this pre-processing should not lead to loss of information so depending on the task; necessary pre-processing steps should be identified.

LEMMATIZATION

lemmatization involves reducing the words to their root or stem form like “wondered” to “wonder”, “disgusting” to “disgust” etc. The first step of this process involves normalization where the base word conversion for every text is done with the reference of parts of speech. This is necessary because otherwise it results in unnecessary increase in the number of features.

TEXT CLEANING

Cleaning tends to be used for removing some encoding parts like \\xa0, \\xc2, \\n, etc. and some unwanted html tags, punctuation marks, single letters and wide spaces. These put bias on the results if not removed. For example: In the sentence “Wow!! Worst performance ever”. The exclamatory marks (“!!”) are removed in-order to get a proper outcome from cleaning. Single letters refer to some unwanted letters like when an (’) removed from a word, then the letter becomes a single useless word with no meaning such things are avoided. For example: From the word ‘your’s’ if (’) is removed then both the word ‘your’ and the letter ‘s’ are separated results in making ‘s’ as a single letter.

FEATURE EXTRACTION

The number of occurrences of each feature may not be a good feature to be used directly. For example, a word like 'the' occurs in almost all the text strings. We might wrongly consider it as important due to the high frequency of occurrence. So, we need to reduce its importance. This is achieved by using few methods. The frequency calculation is of two categories (1) TERM FREQUENCY- where the frequency of a word is found in single document file (2) INVERSE DOCUMENT FREQUENCY – where the frequency of a word is found in all document files.

Tf-idf METHOD

Tf-idf stands for: “Term frequency x Inverse document Frequency” and it measures the ratio of the number of times the token occurs in a particular text string to the fraction of the documents in which the token occurs. After constructing a key word set using n-grams, skip grams and special rule, using this list of features; we find the tf-idf score of each feature. The item in the tf-idf vector corresponds to the tf-idf weight of the feature (keyword) at the same index. A keyword’s term frequency is the number of times the word appears in the title. Its

document frequency is the number of titles the word appears over the total number of titles. The keyword's tf-idf weight is its term frequency divided by its document frequency. Then, the tf-idf vector is normalized. The tf-idf vector represents the occurrence of the important keywords a user is interested in. In this model, each document is first represented as a term-frequency vector in the term space:

$$d_{itf} = (tf_{1j} + tf_{2j} + \ldots\ldots\ldots\ldots .. tf_{nj}) \qquad j=1,2,3,\ldots\ldots D$$

where 'tf_{ij}' is the frequency of the ith term in document, n is the total number of the selected vocabulary, and D is the total number of documents in the collection. Next, we weight each term based on its inverse document frequency (IDF) and obtain a tf-idf vector for each document: $d_j = (tf_{1j} * id_{f1}\ ,\ tf_{2j} * id_{f2}\ ,\ldots\ldots\ldots,\ tf_{nj} * id_{fn}\ )$ $\quad j=1,2,3,\ldots\ldots\ldots\ldots D$

BEST FEATURE SELECTION

The features generated are of the order of a hundred thousand features which is too big a number to be handled efficiently by algorithms like SVM and Logistic Regression. So, we need to select a few best features out of our set of features. We apply a statistical test known as "Chi Squared Test" to our feature matrix to select best k number of features where k is the parameter.

CHI-SQUARE TEST

This test finds if a pair of categorical variables on a data is statistically dependent or independent. In this case, the pair of variables is: The label of the sentence: "insult or not" and the occurrence or non-occurrence of a feature". The features which score high in this test are the important features which we keep while we discard the remaining features. To understand this measure better, consider the following example. Imagine a sample of our data that looks like this:

TEXT STRING	**FEATURE-1: 'YOU'**	**FEATURE-2: 'THE'**	**FEATURE-3: 'YOU ARE'**	**FEATURE-4: 'FOOL'**	**LABEL**
1	PRESENT	PRESENT	NOT PRESENT	PRESENT	INSULT
2	PRESENT	PRESENT	NOT PRESENT	NOT PRESENT	NOT INSULT
3	NOT PRESENT	PRESENT	NOT PRESENT	PRESENT	INSULT
4	PRESENT	NOT PRESENT	PRESENT	NOT PRESENT	INSULT

Had the presence of 'you' and 'being insult' been independent, the expected number of rows in which both of these happen would be:

E(‘you present’, insult) = (N(‘you present’)*N(‘insult’))/N

where N ('you present', 'insult') is the number of rows which have the feature 'you' and are labeled as 'insult' and N is the total number of rows.

$\chi^2 = \sum (Observed(i,j) - Expected(i,j))^2 / Expected(i,j)$

where i= {'yes present', 'yes not present'} and j= {'insult', 'not insult'} This is a measure of dependency in the two categorical variables. Higher this value, more related is the two variables. Lesser the value, more independent are the variables

CLASSIFICATION

After we have constructed the features and extracted the top features, we apply machine learning algorithms to learn models. We use SVM and Logistic regression algorithms on our feature data. Finally, we combine the results of both the algorithms to get the final results. The results are obtained as probabilities which indicate the probability of a comment being an insult.

3.3. FUNCTIONAL REQUIREMENTS

The detection model should consist the following properties in order to state that the model is working with efficient accuracy.

- The input we give in text format should be accepted by the model.
- The output should be returned with in 2 seconds predicting whether the comment is offensive or not.

3.4. NON-FUNCTIONAL REQUIREMENTS

- **USER INTERFACE:** It is generally understood to mean how the information in the model is presented to the user.

- **USABILITY:** The interface should provide easy access to the user for accurate performance of the model.

- **DATA ENTRY:** It is concerned with how the data is entered and validated.it allows the user to enter data via user input methods.

- **RESULT GENERATION:** The model shall have a result feature that will allow by user to generate a result showing the information.

3.5. HARDWARE REQUIREMENTS

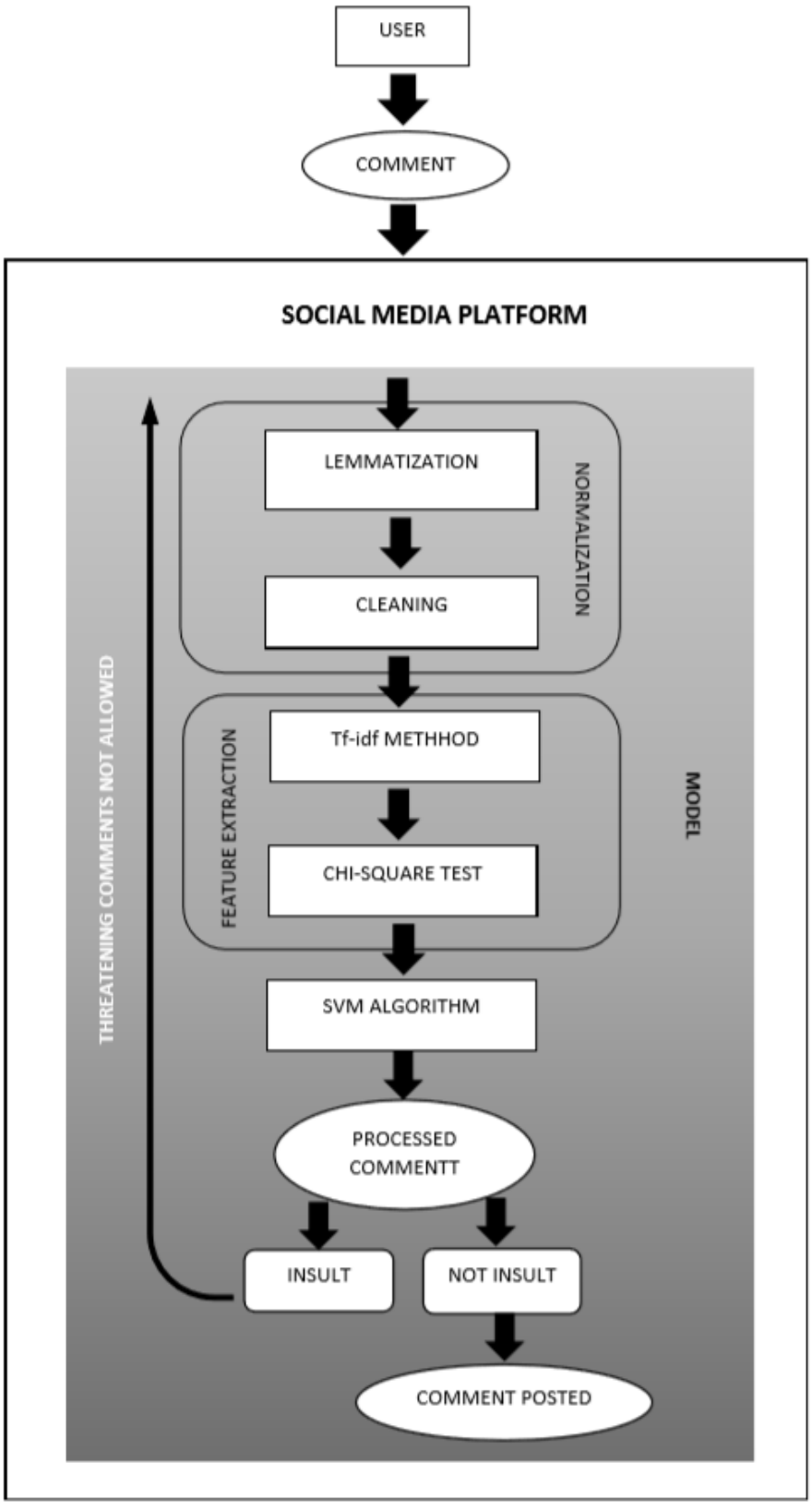
- PROCESSOR : INTEL i5
- RAM : 32 GB
- HARD DISK SPACE : 8 GB

3.6. SOFTWARE REQUIREMENTS

- OPERATING SYSTEM : WINDOWS 8, WINDOWS 9, WINDOWS 10
- MINI FRAMEWORK : FLASK
- TOOLS : SPYDER IN ANACONDA DISTRIBUTION

4. DESIGN

4.1. SYSTEM ARCHITECTURE DIAGRAM



4.2. UML DIAGRAMS

4.2.1. USE CASE DIAGRAM

The use case diagrams in this system analyses the system requirements, High-level visual software designing, Capturing the functionalities of a system, making a model from a basic idea behind the system, Forward and reverse engineering of a system using various test cases. Use cases are intended to convey desired functionality that is carried out in the model.

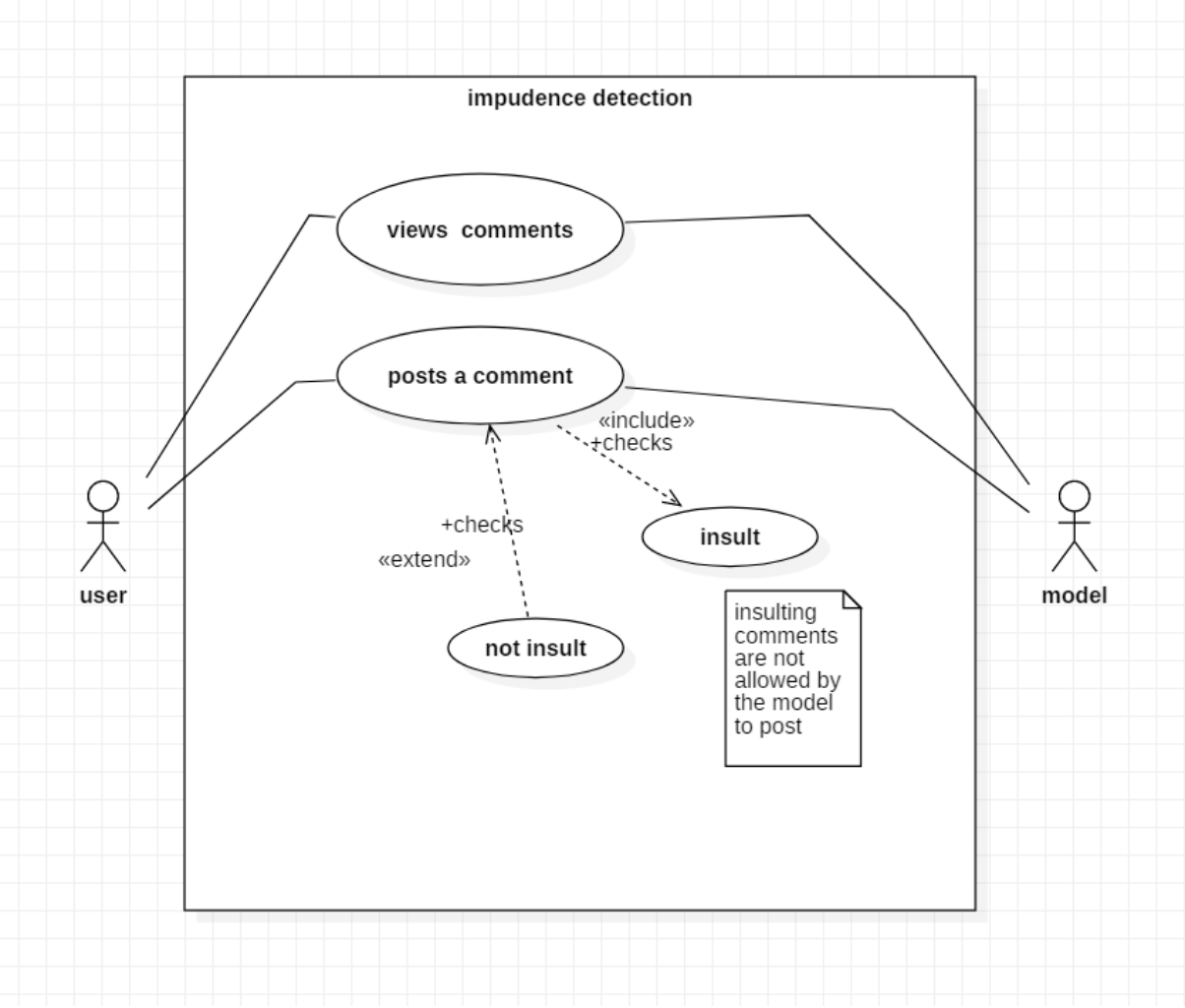


Fig (4.2.1)

4.2.2. CLASS DIAGRAM

The Class Diagrams Illustrate data models for even very complex information systems. Here it provides an overview of how the model is structured before studying the actual code. This can easily reduce the maintenance time and also helps for better understanding of general schematics of the model. Allows drawing detailed charts which highlights code required to be programmed which is very helpful for developers.

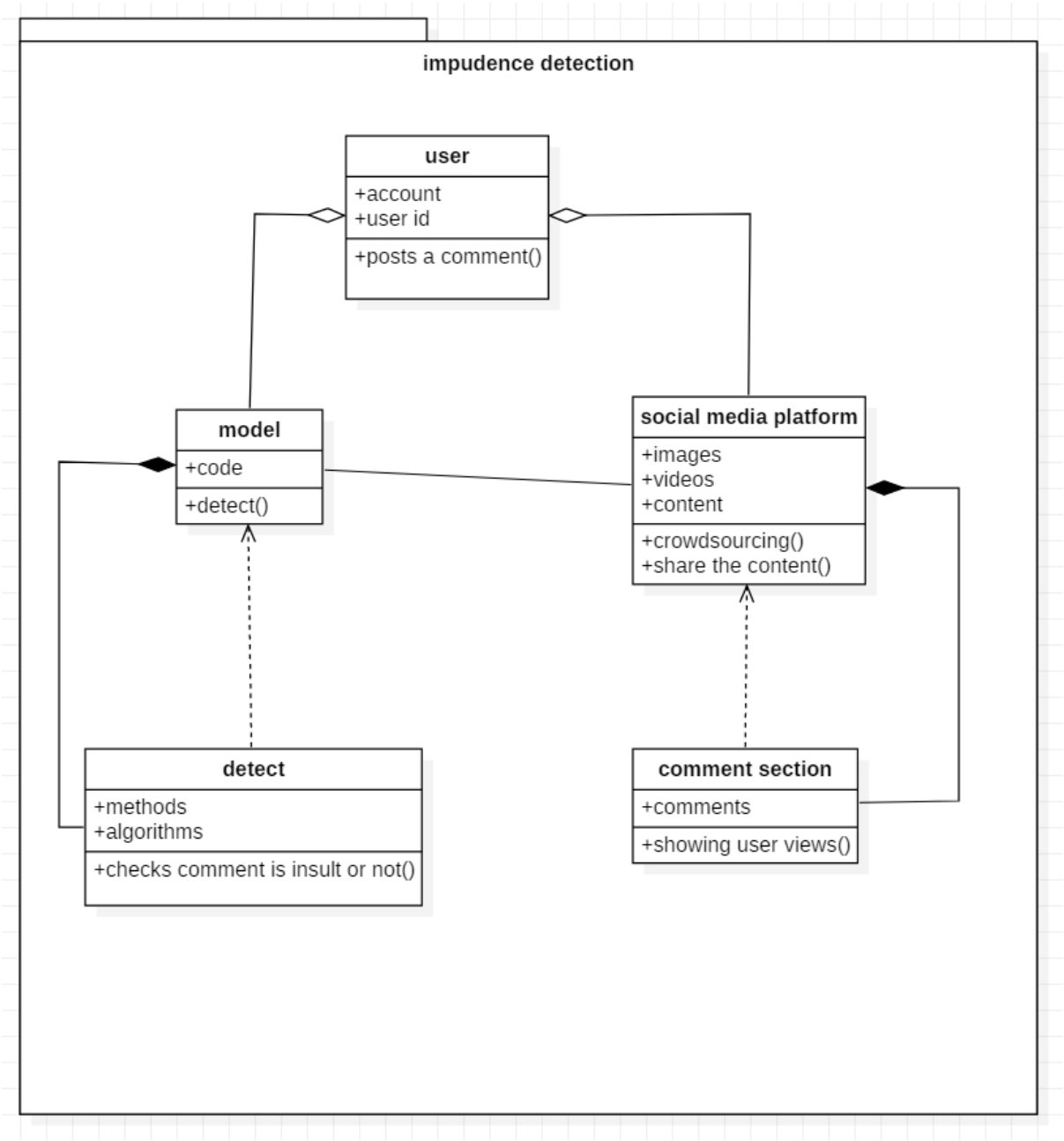


Fig (4.2.2)

4.2.3. SEQUENCE DIAGRAM

Sequence diagrams here models the interaction between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed. In simpler words, here the sequence diagram shows different parts of a system work in a 'sequence' to get the desired result.

The two possibilities of object interaction sequences are shown below

CASE 1: The sequence interaction of the objects in the model to process a comment posted by the user which is resulting to be an insult.

CASE 2: The sequence interaction of the objects in the model to process a comment posted by the user which is resulting to be not an insult.

CASE 1

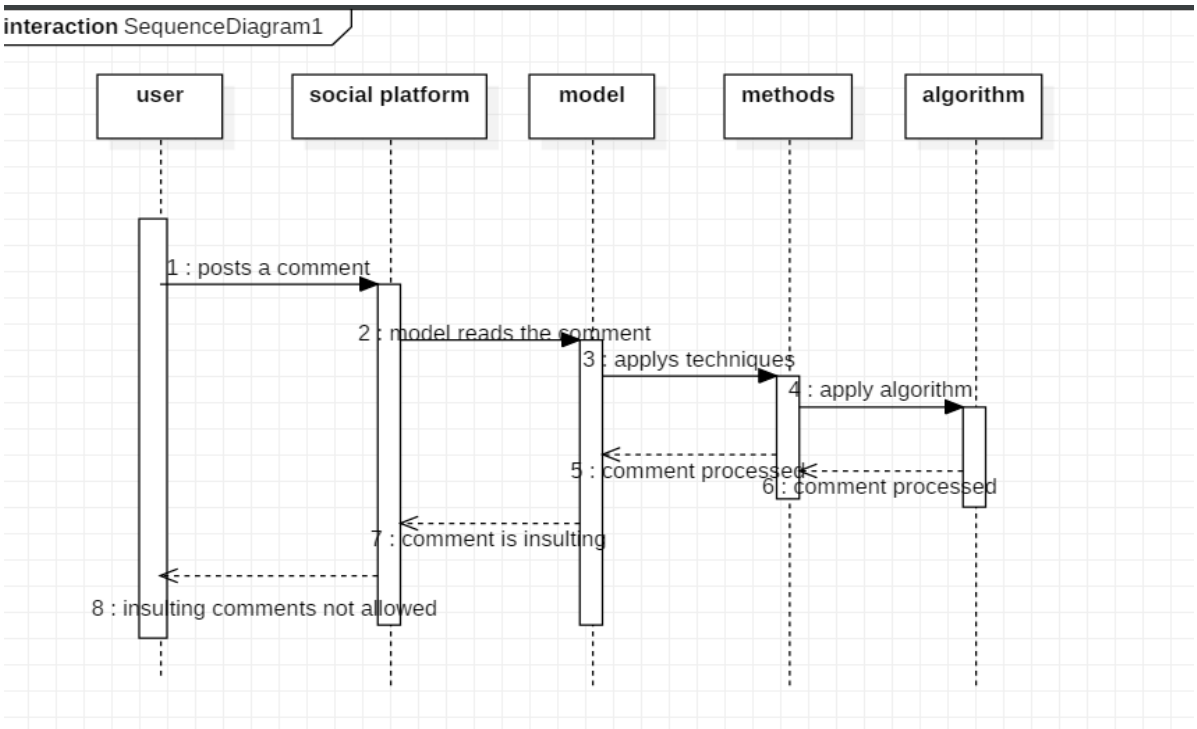


Fig (4.2.3(a))

CASE 2

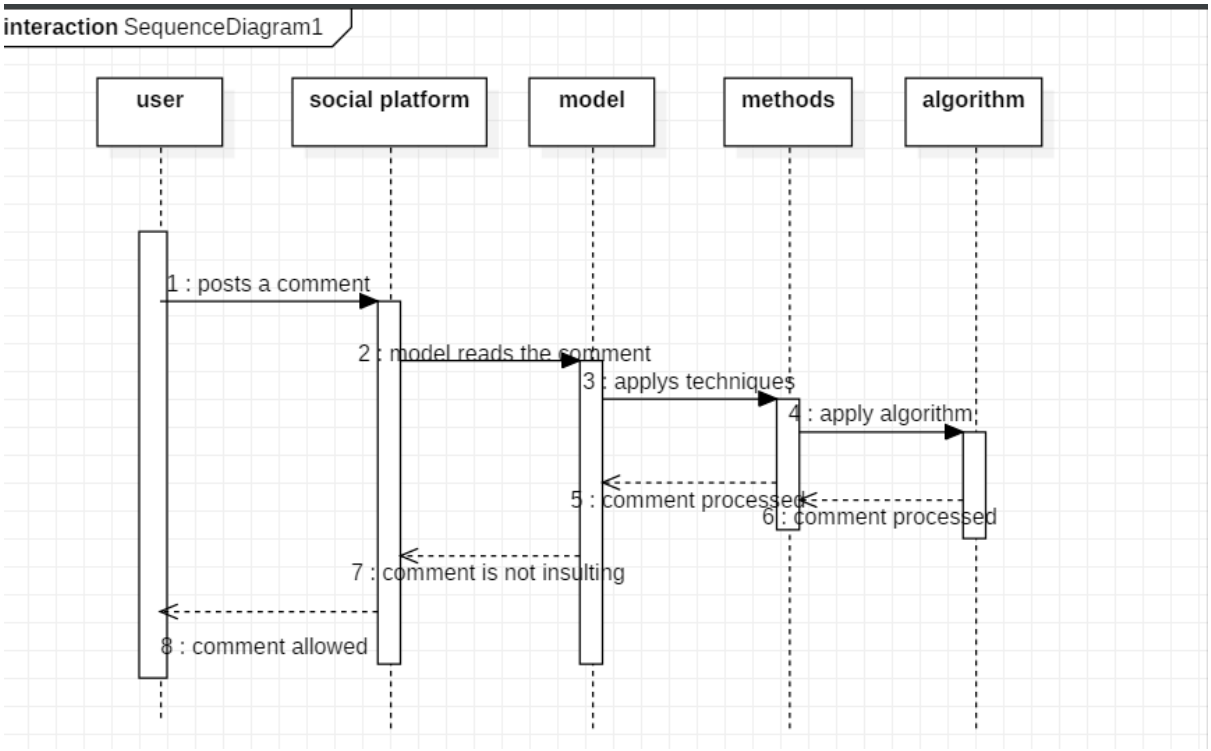


Fig (4.2.3(b))

4.2.4. COLLOBORATION DIAGRAM

The colloboration diagram visualizes the interactive behaviour of the system. Visualizing the interaction is a difficult task. Hence, the solution is to use different types of models to capture the different aspects of the interaction. Sequence and collaboration diagrams are used to capture the dynamic nature but from a different angle. Here, it captures the dynamic behaviour of the model, describes the message flow, structural organization of the objects and interaction among them.

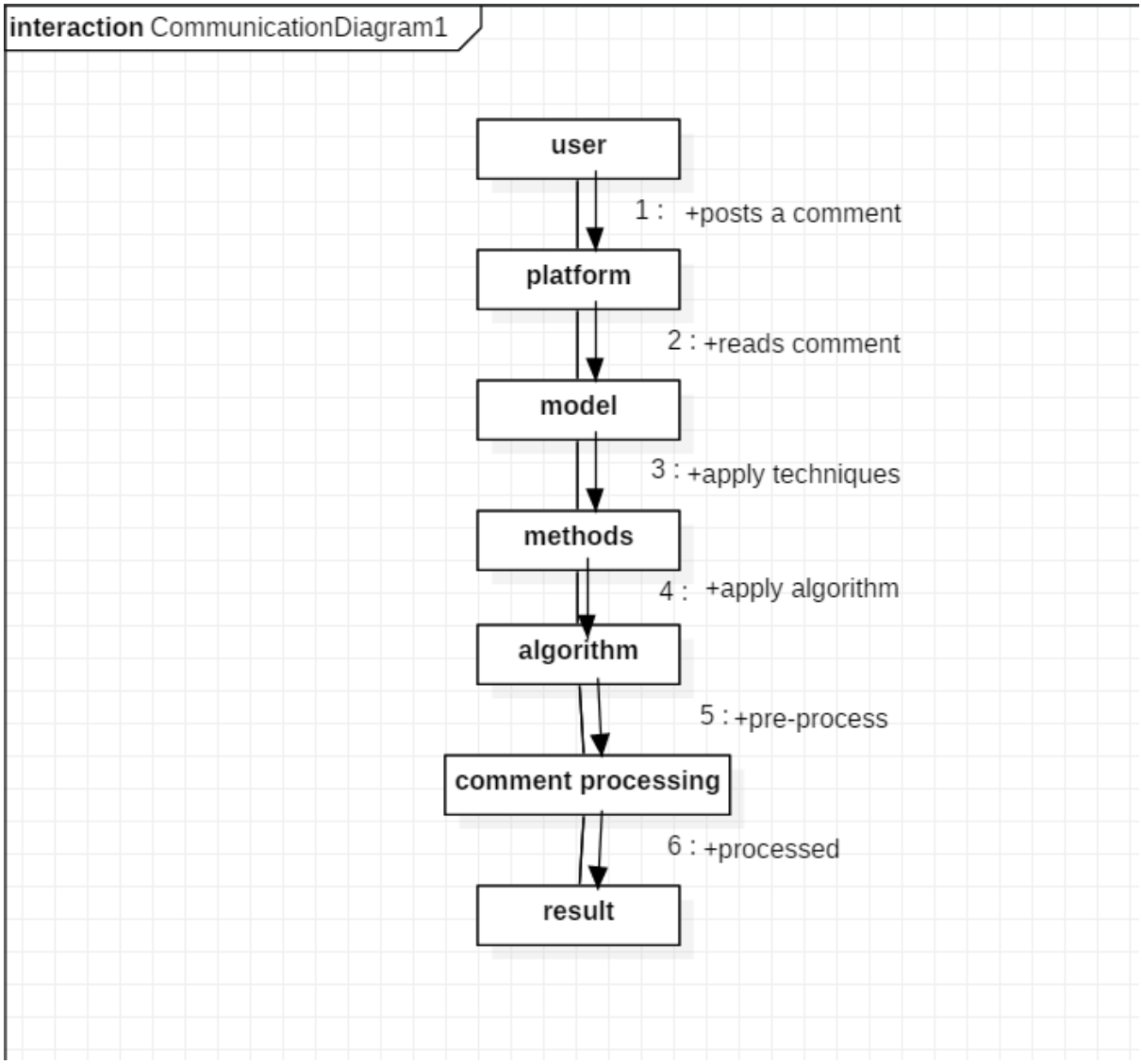


Fig (4.2.4)

4.2.5. STATE CHART DIAGRAM

State chart diagram describes the flow of control from one state to another state in this model. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination. State chart diagrams are also used for forward and reverse engineering of a system. It models the reactive system. Here it models the dynamic aspect of a system, models the life time of a reactive system, describe different states of an object during its life time and Define a state machine to model the states of an object.

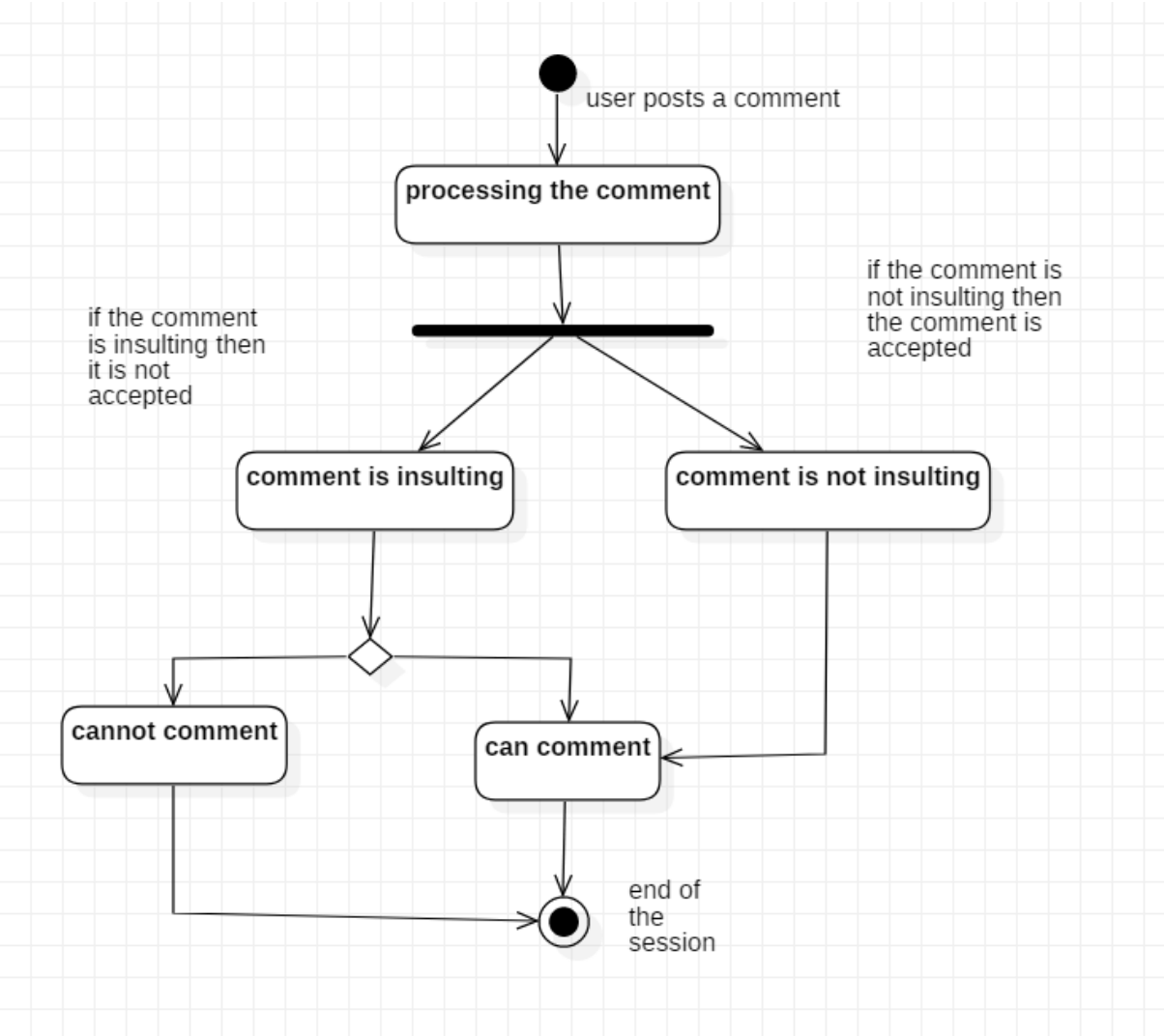


Fig (4.2.5)

4.2.6. ACTIVITY DIAGRAM

It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not.

especially It is to Draw the activity flow of a system, Describe the sequence from one activity to another, Describe the parallel, branched and concurrent flow of the system.

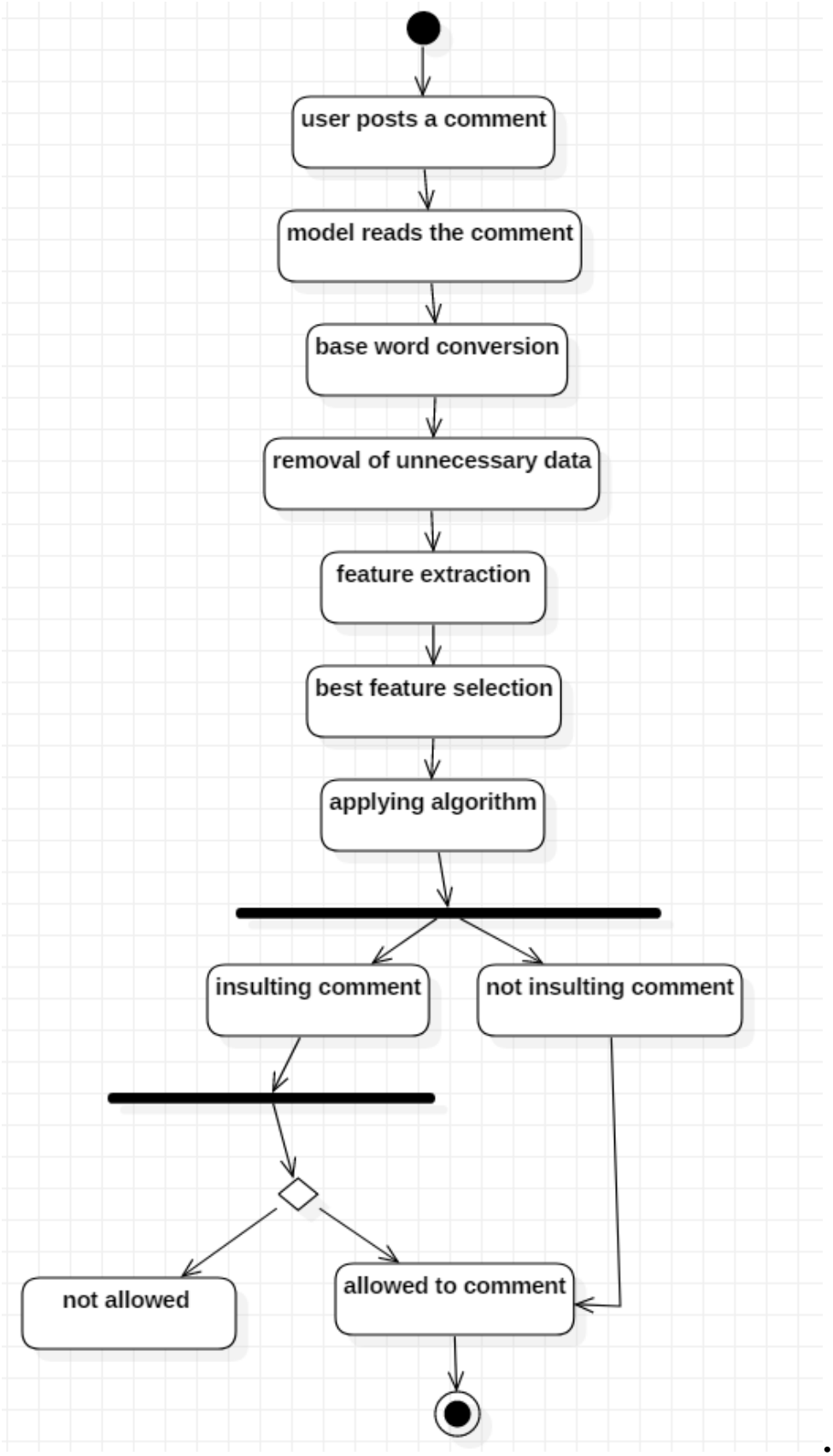

user posts a comment
model reads the comment
base word conversion
removal of unnecessary data
feature extraction
best feature selection
applying algorithm
insulting comment
not insulting comment
not allowed
allowed to comment

Fig (4.2.6)

5. IMPLEMENTATION

5.1. MODULE DESCRPTION

Turning now to the module description, the main modules used in detection model are FLASK - micro framework, Natural language tool kit (NLTK), Scikit Learn, these modules has been applied to yield a proper easy-to-use interface of detection model. The light weight web application back-end framework FLASK, has a remarkable feature of presenting a new approach to the model as a web-based application which responds by stating whether the comment is an insulting or a non-insulting one. NLTK a python library that generates considerable interests in terms of processing the text like lemmatization, text cleaning which normalizes the text for feature extraction. NLTK requires Python 2.7, 3.5, 3.6, or 3.7. Scikit learn probably the most useful library for machine learning in python which seeks to address various algorithms like clustering and classification.

5.2. SOURCE CODE

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov  7 14:10:56 2019

@author: DELL
"""

# importing required modules
import csv
import re
```

```
import nltk
from nltk.corpus import stopwords,wordnet
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
#functions
def read():
    x,y=[],[]
    with open('data.csv', 'r') as csvFile:
        reader = csv.reader(x.replace('\0', '') for x in csvFile)
```

```
        for row in reader:
            x.append(row[2])
            y.append(row[1])
    csvFile.close()
    return x[1:5000],y[1:5000]
def lemmatize_verbs(x):
    word_tokens = word_tokenize(x)
    lemmatizer = WordNetLemmatizer()
    lemmas = []
    for w in word_tokens:
        lemma = lemmatizer.lemmatize(w, pos='v')
        lemmas.append(lemma)
    return ' '.join(lemmas)
def get_wordnet_pos(word):
    tag=nltk.pos_tag([word])[0][1][0].upper()
    tag_dict={'J':wordnet.ADJ,
              'N':wordnet.NOUN,
              'V':wordnet.VERB,
              'R':wordnet.ADV}
    return tag_dict.get(tag,wordnet.NOUN)
def preprocessing(x):
```

```
    l=[]
    for i in x:
        i=i.lower()
        i=re.sub(r'#[a-zA-Z]+','',i)
        i=re.sub(r'&amp;quot;','',i)
        i=re.sub(r'@\w+','',i)
        i=re.sub(r'https?://+S','',i)
        i = re.sub(r'\W', ' ', i)
        i=re.sub(r'\s+[a-zA-Z]\s+', ' ', i)
        i=re.sub(r'\s+', ' ', i, flags=re.I)
        l.append(i)
    return l


def tfi_idf():
    vectorizer = TfidfVectorizer (max_features=600, min_df=7, max_df=0.8,
stop_words=stopwords.words('english'))
    processed_features = vectorizer.fit_transform(trimout).toarray()
    #print(np.shape(processed_features))
    return processed_features,vectorizer
def chisquare(model,answer):
    X=model
    y=answer
```

```
    chi2_selector = SelectKBest(chi2, k=300)
    select=chi2_selector.fit_transform(X, y)
    return select,chi2_selector
def supportvec():
    clf = svm.SVC(kernel='linear')
    clf.fit(X_train, y_train)
    #v1,v2=prediction(clf)
    y_pred = clf.predict(X_test)
    return y_pred,clf
def logreg():
    logreg = LogisticRegression(solver='lbfgs')
    logreg.fit(X_train,y_train)
    y_pred=logreg.predict(X_test)
    return y_pred,logreg
def navie_bayes():
    clf = GaussianNB()
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    return y_pred,clf
def knn():
    model = KNeighborsClassifier(n_neighbors=3)
```

```
    model.fit(X_train,y_train)
    y_pred=model.predict(X_test)
    return y_pred,model
def random_forest():
    clf=RandomForestClassifier(n_estimators=100)
    clf.fit(X_train,y_train)
    y_pred=clf.predict(X_test)
    return y_pred,clf
def trim(t):
    li=[]
    for i in t:
        li.append(lemmatize_verbs(i))
    return li
def output(x,i):
    t1=''
    if label[i]:
            t1="Not Insult"
    else:
            t1="Insult"
    print("Sentence : ",data[i],"Original : ",t1,"Prediction : ",x,sep='\n')
def predict_sent(c,sen):
```

```
    y_pred = c.predict(sen)
    pre=''
    #rev=''
    if y_pred:
        pre='Not Insult'
    else:
        pre='Insult'
#    if ans[0]:
#        rev='Not Insult'
#    else:
#        rev='Insult'
    print("Prediction : ",pre)
def display(name,yp):
    print(name+" Accuracy:",metrics.accuracy_score(y_test, yp))
    print(name+" Precision:",metrics.precision_score(y_test, yp))
    print(name+" Recall:",metrics.recall_score(y_test, yp))
def ensemble():
    model1 = n2
    model2 = n
    model3 = n3
    model4 = n4
```

```
    model = VotingClassifier(estimators=[('lr', model1), ('svm',
model2),('navie',model3),('knn',model4)])
    model.fit(X_train,y_train)
    print(model.score(X_test,y_test))
if __name__=='__main__':


    data,label=read()
    clean_text=preprocessing(data)
    trimout=trim(clean_text)
    tfi_idf_model,vect=tfi_idf()
    tar=np.asarray(label)
    tar=tar.astype('int32')
    X_kbest,fix=chisquare(tfi_idf_model,tar)
    X_train, X_test, y_train, y_test = train_test_split(X_kbest, tar,
test_size=0.2,random_state=9)
    svm_target,n=supportvec()
    logreg_target,n2=logreg()
    navie_target,n3=navie_bayes()
    knn_target,n4=knn()
    rf_target,n5=random_forest()
    ensemble()
    while True:
```

```
option=input("Select : \n1.metrics\n2.user given samples\n3.exit\n")
if option=='3':
    break
elif option=='1':
    display('SVM',svm_target)
    display('Logreg',logreg_target)
    display('Navie Bayes',navie_target)
    display('Knn',knn_target)
    display('random forest',rf_target)
elif option=='2':
    review=input("enter sample :\n")
    #ans=np.asarray([int(input("0 for insult or 1 for not insult : "))])
    t11=[]
    t11.append(review)
    t2=preprocessing(t11)
    t1=lemmatize_verbs(t2[0])
    t12=[]
    t12.append(t1)
    t3=vect.transform(t12).toarray()
    t4=fix.transform(t3)
    predict_sent(n5,t4)
```

```
    #print(t2)
else:
    print('Invalid option')
```

6. TESTING

6.1. NEED FOR TESTING

We made an attempt of Testing the constructed model in the aim of finding error. To make the model perform well, it should be error free. So certain tests are carried out to make the model free from the errors. Successful testing removes all the errors from the detection model. Here, are the testing results.

6.2. TEST PROCESS

INPUT	OUTPUT	TEST RESULT
You are the worst person I have ever seen	It is an INSULT	SUCCESSFUL
I think this shoe suits you very well	It is NOT AN INSULT	SUCCESSFUL

7. EVALUATION AND RESULTS

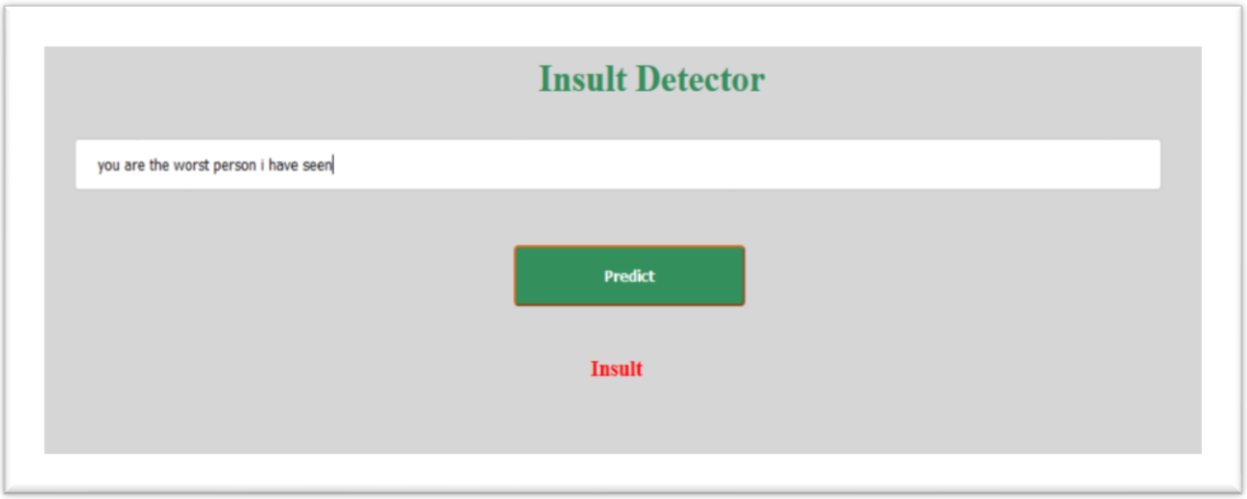
7.1. RESULTS

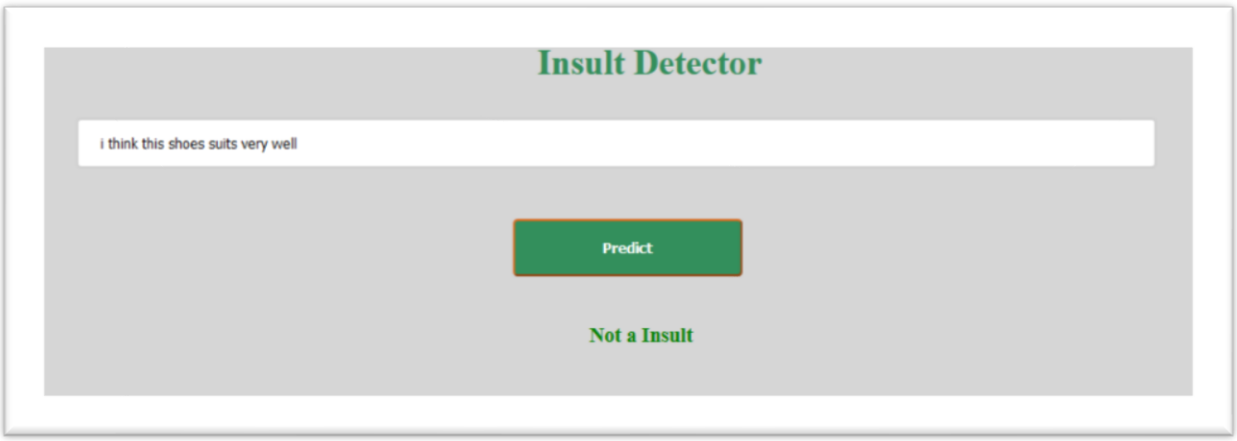
- Accuracy with logistic: 75%
- Accuracy with SVM: 74%
- Accuracy with logistic and SVM: 75%

As we observe, the following are the metrics and its values for the algorithms used:

ALGORITHM	ACCURACY	PRECISION	RECALL
SVM	74.95%	77%	27%
Logistic	75%	79%	28%

7.2. SCREEN SHOTS





8. CONCLUSION AND FUTURE WORK

Our findings would seem to show that the attempt of detecting comments intended to be insulting to a participant of conversation is achieved and For, this we observed the dataset and proposed the features to test the hypothesis. Machine learning algorithms SVM and Logistic Regression are used to train models. It is tested on a dataset and obtained a considerable gain in accuracy. However, still the current attempt of detection was limited when the user comments were like “y0u @re a F00l” instead of “you are a fool” Grammatical mistakes e.g. “What on earth a BIGGOT like you is doing walking on the face of earth?” Typography: s h i t (shit), People usually circumvent dictionary due to flexibility of conversation e.g. @$$hole (asshole), Wordplays e.g. kucf oyu, Sarcasm

e.g. “Sometimes I don’t know whether to laugh at you or pity you.”, Innuendo where the indirect mention to someone has been made e.g. “Only cowards, thieves, cheats and liars hide behind pseudonyms.”. however, this is not the focus of this study all these things are recovered in future work and we are intended to elaborate this work in native languages also.

9. ACLNOWLEGEMENT

We thank our faculty N.Rama Krishna for his valuable support throughout the project, guiding us from time to time and helping us get through various roadblocks. We are thankful to "Kaggle.com" online forums as whose discussions proved really helpful to us and open source "scikit learn" package in python.

10. REFERENCES

- Ellen Spertus, Smokey: Automatic recognition of hostile messages. In Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence,
- Altaf Mahmud, Kazi Zubair Ahmed, and Mumit Khan. Detecting flames and insults in text. In Proceedings of the Sixth International Conference on Natural Language Processing.
- For bad words file:
- https://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/
- For dataset:
- https://twitter.com/
- https://www.kaggle.com/datasets