User Profiling for Computer System

Submitted By

Rohan Upadhyay

16MCEC28

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

User Profiling for Computer System

Major Project

Submitted in fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Rohan Upadhyay

(16MCEC28)

Guided By

Dr. Sharada Valiveti

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

Certificate

This is to certify that the major project entitled **"User Profiling for Computer System"** submitted by **Rohan Upadhyay(16MCEC28)**, towards the fulfillment of the requirements for the award of degree of Master of Technology in Computer Engineering (Computer Science and Engineering) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-II, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. Sharada Valiveti
Guide & Associate Professor,
Institute of Technology,
Nirma University, Ahmedabad.

Dr Priyanka Sharma
Professor,
Coordinator M.Tech - CSE,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor and Head,
CE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Rohan Upadhyay**, **16MCEC28.**, give undertaking that the Major Project entitled "**User Profiling for Computer System**" submitted by me, towards the fulfillment of the requirements for the degree of Master of Technology in **Computer Science and Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:

Place:

Endorsed by

Guide Name

(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Dr. Sharada Valiveti**, Associate Professor, Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for her valuable guidance and continual encouragement throughout this work. The appreciation and continual support she has imparted has been a great motivation to me in reaching a higher goal. Her guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Rohan Upadhyay

16MCEC28

Abstract

With the advancing technology, the use of technology for malpractices is ever increasing. It is very easy to pretend to be someone else by forging their identity or through illegal use of authentication credentials. Hence, it is always recommended to have a self-detecting system which is capable of verifying the genuineness and authenticity of the user. This project work is part of the Fraud Detection System (FDS), in which the authenticity of user is verified in real-time. Then after, this information can be used to leverage the capabilities of the system dynamically, so that the illegal user does not harm the system or have access to some confidential information. In this research work, user profiling based verification method which can verify the identity of the user in real time is proposed. In this approach, a novel method of collecting the data from the user in real time is suggested. This data is trained using recurrent neural network. With the help of this trained model, specific user's identity can be verified in real time. The verification is carried out in real-time as the user uses the computer system. Deploying a suitable mechanism to then react based on the output of the user profiling system makes the system immune to identity theft based attacks.

Abbreviations

DL	Deep Learning
FDS	Fraud Detection System
GPU	Graphical Processing Unit
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
SVM	Support Vector Machine

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 User Profiling

User profiling is the process of classifying users based on the dynamic profiles created. Users can be profiled into various categories or they can be identified individually based on their interaction with the computer system. This interaction is purely dependent on the characteristics of the individual using the system. Profiling requires sets of data for a particular user known as profiles. This profile is then compared with a larger dataset to check which category the user belongs to or which particular user it is. To profile the behaviour of a user, we need more than just data about the user. We need to know what the user is currently doing. We need to know what actions the user performs, how the user performs them and based on the actions, we classify the behaviour of the user. Figure 1.1 demonstrates the process of user profiling:

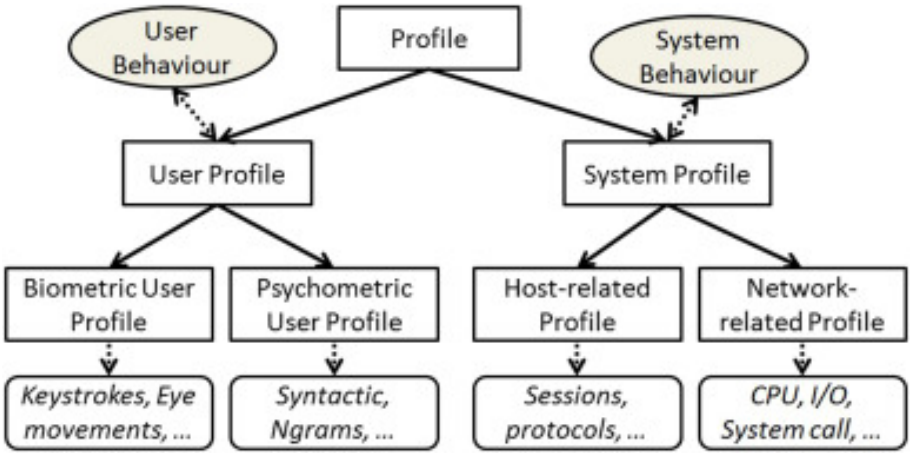

Figure 1.1: Example of User Profiling [1]

Problem identification

The process of user profiling begins with understanding the problem. Objectives need to be identified before and the related scope of work must be defined.

Data collection

The next step is to collect the necessary information required for profiling which is related to the application.

Data preparation

This step involves pre-processing the data so that it can be efficiently explored for user profiling.

Profile creation

The profile is created using data mining or Machine Learning (ML) techniques.

Application

Finally, the profile is used for matching users or other entities with the available profiles and decisions are made based on the access capabilities and rights of the user.

1.2 Applications of User Profiling

User profiling is used in a variety of applications now a days. Few of the areas of user profiling are discussed below:

Recommender Systems

A recommender system is a system that predicts the preferences of the user and the opinion of the user on a particular topic. Recommender systems are used to give suggestions to the user based on the previous choices of the user. The applications of recommender systems include recommending items to the user which the user is more likely to buy in e-commerce related applications.

Security

Another important application of user profiling is security. The system tries to verify the identity of the user to make sure that the system is not in the wrong hands. The profile of the user is created using the usage of the user. Afterwards, this profile is compared to the usage of the user whose identity needs to be verified

1.3 Problem Statement

With the increasing use of technology for hacking and identity theft, it is very easy for a hacker to pretend to be someone else by forging the identity. Hence, it is very important to find other, more reliable ways for verifying the identity of the user. One good way of verifying the identity of the user is making the system learn the behaviour of different users. Based on the behaviour, the user can be verified by the system even if the identity is forged.

1.3.1 Objective

On the completion of this project, a new method for data collection for user will be established and a neural network model will be created which will be able to train using the collected data. Finally, the model will be able to identify the user in while the user uses the system.

1.3.2 Scope of Work and Assumptions

The scope of this project includes various programming libraries that can interact directly with the system and get monitor data from input devices connected to the system like keyboard and mouse. Knowledge of various deep learning algorithms will also be required to decide the suitable algorithm for this project. Furthermore knowledge of threading will be necessary as the final program will be expected to perform multiple tasks simultaneously

Chapter 2

Literature Survey

This chapter covers the study and work related to user profiling and other tools that have been used for this work. The most difficult problem faced by the users in user profiling is the detailed dataset collection and the format of this data. Next step is to choose the technique for profiling the user i.e. the choice of machine learning algorithm to use.
This section contain the survey of papers related to user profiling used in this research.

2.1 Related Work

The list of publications are described in table. The column publication indicates where the publication is taken.

Publication	Title
Conference: IISA 2017, At Larnaca, Cyprus	Emotions and User Interactions with Keyboard and Mouse
ICCCI 2015, At Madrid, Spain, Volume: 7 th	Keystroke Data Classification for Computer User Profiling and Verification
International Journal of Computer Applications (0975 8887), December 2014	User Profiling-A Short Review
International Journal of Advance Foundation and Research in Computer (IJAFRC) Volume 1, Issue 1, Jan 2014.	User Profiling Trends, Techniques and Applications
Book: Advanced Computing and Systems for Security: Volume 1	Computer User Profiling Based on Keystroke Analysis
JOURNAL OF MEDICAL INFORMATICS and TECHNOLOGIES Vol. 23/2014, ISSN 1642-6037	User Profiling Based on Multiple Aspects of Activity in a Computer System

Papers for study on Recurrent Neural Network are listed below. The following papers were surveyed to get deeper knowledge of RNNs to figure out which neural network algorithm is best suited for this project.

Publication	Title
ArXiv	A Critical Review of Recurrent Neural Networks for Sequence Learning
Neural Computation 9(8):1735-1780, 1997	LONG SHORT-TERM MEMORY
IJCAI-17	What to Do Next: Modeling User Behaviors by Time-LSTM

2.2 Method of Analysis And Training/Testing Method

The list of the methods to analyse the data and the training/test method is described in this section. Many different ways for analyzing the data to create profiles has been proposed. One of the proposed ideas was using an SVM for classification. The research paper proposed which data needs to be collected. This work primarily focuses on the personalized characteristics of an individual like the time between key press and key release, and time between two consecutive key presses [2]. Another helpful method for acquiring data mentioned in the papers surveyed was getting the mouse coordinates at every mouse event like click or scroll[3].

Chapter 3

Proposed Method

Many researchers have worked on various techniques available for user profiling as mentioned in chapter 2. A novel architecture to perform an efficient user profile is as mentioned in Figure 3.1. At first, data will collected from the system and data from various users will be stored. Then, the data from various users will be combined together. After that, the data will be pre-processed so that it can be efficiently analyzed. Using the pre-processed data, user profiles will be created. After the profiles are created, user will be verified in real time and based on the verification further decisions will be taken.

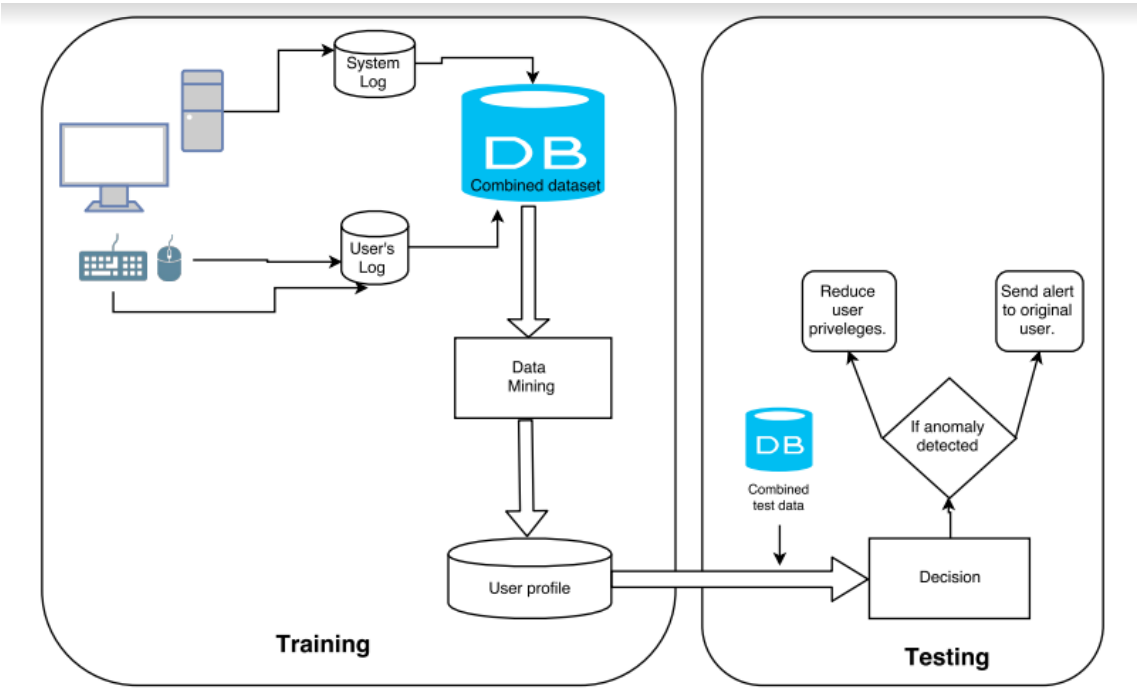

Figure 3.1: Proposed Architecture

3.1 Acquiring data

Data has been acquired from keyboard and mouse inputs. Every time, the system receives an input, a new sample is created. Table 3.1 shows the list of profiles stored for each individual.

Table 3.1: Fields of extracted profiles

Name	Description
charc	Number of character keys pressed on the keyboard at a time.
lClick	1 if left mouse button is pressed, 0 otherwise.
rClick	1 if right mouse button is pressed, 0 otherwise.
mClick	1 if middle mouse button is pressed, 0 otherwise.
space	1 if spacebar is pressed, 0 otherwise.
backSpace	1 if backSpace is pressed, 0 otherwise.
delete	1 if delete is pressed, 0 otherwise.
ctrl	1 if control is pressed, 0 otherwise.
alt	1 if alt is pressed, 0 otherwise.
capsLock	1 if capsLock is pressed, 0 otherwise.
shift	1 if shift is pressed, 0 otherwise.
tab	1 if tab is pressed, 0 otherwise.
numLock	1 if numLock is pressed, 0 otherwise.
enter	1 if enter is pressed, 0 otherwise.
mouseX	X coordinate of mouse.
mouseY	Y coordinate of mouse.
scrollX	1 if mouse is scrolled in x axis
scrollY	1 if mouse is scrolled in y axis
timeDiff	Time difference between two events

3.2 Long Short Term Memory

The Long Short Term Memory (LSTM) variant of recurrent neural network for training is proposed. This is because RNNs have have a memory which can store a sequence of data before which it gives the output. LSTM is an improved version of RNN which replaces nodes with memory cells. Figure 3.2 shows the architecture of an LSTM cell.

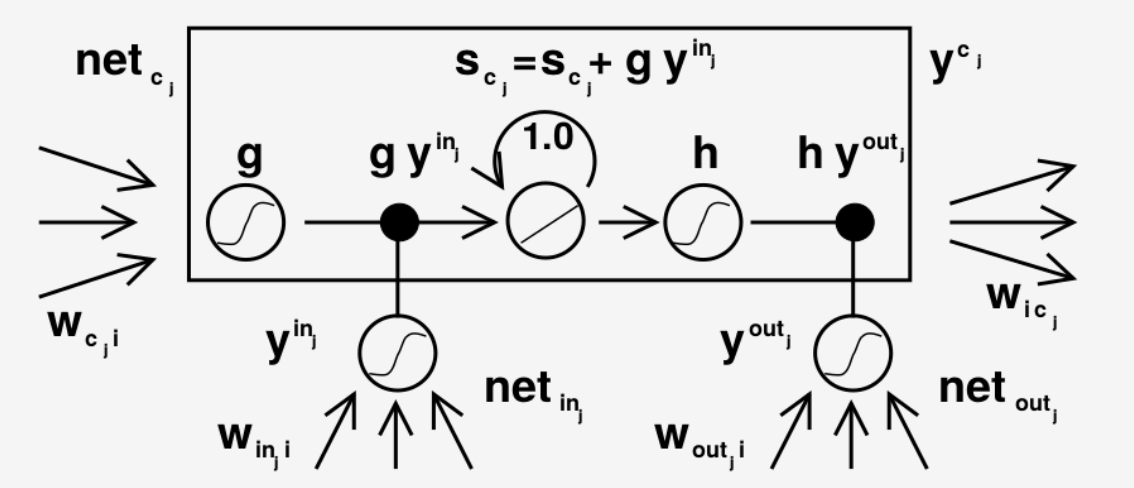

Figure 3.2: LSTM

Each cell of LSTM contains gates which decide which data to keep. Both new data and the old data pass through the gates. The gate here is an analog gate which means it can contain any value between 0 and 1. LSTM solves the vanishing gradient problem of RNNs. [4]. An LSTM with a memory of 250 timesteps as in any smaller timestep is used here. Otherwise, meaningful information might be lost.

Chapter 4

Implementation Results

Data is collected three users as they used their windows systems. From the collected data, 50000 samples from each user are used. These 50000 samples are then pre-processed into sequences of length 250. The final dataset consists of batches of 250 sequences of data which is then divided into multiple files. Each file contains a batch of 1024 such sequences. The data in each file are in the form of a three dimensional matrix of the shape [batchsize=1024,timesteps=250,features=19]. The file is stored in the form of a numpy array as a '.npy' file.

The flow of this work starts with the 'data_collect.py' file which is used to collect real time usage data from the user. Refer figures 4.1 and 4.2,

```python
import os
#import win32gui
import threading
from threading import Timer
from threading import Thread
import pynput
from pynput import mouse
from pynput.mouse import Button
from pynput.keyboard import Key
from pynput import keyboard
import math
import time
#os.system("tasklist")
#print(win32gui.GetDoubleClickTime())
i=1
if not os.path.exists(os.path.dirname(r"data/")):
    os.makedirs(os.path.dirname(r"data/"))
while os.path.exists(r"data/"+os.environ['USERNAME']+"_data_"+str(i)+".txt"):
    i+=1
f=open(r"data/"+os.environ['USERNAME']+"_data_"+str(i)+".txt","a")

stime=time.time()
ltime=0
mData={}
tempData={}
mData['user']=os.environ['USERNAME']
mData['charc']=0
mData['lClick']=0
mData['rClick']=0
mData['mClick']=0
mData['sTime']=time.time()-stime
```

Figure 4.1: Code for Data Collection 1

```
43    mData['mouseY']=pynput.mouse.Controller().position[1]
44    mData['scrollX']=0
45    mData['scrollY']=0
46    mData['timeDiff']=0
47
48    iloop=1
49
50    keyPrsd=[]
51  ⊞ def on_click(x, y, button, pressed):⋯
86
87  ⊞ def on_scroll(x, y, dx, dy):⋯
111
112 ⊞ def on_move(x,y):⋯
128
129 ⊞ def recMousList():⋯
137
138 ⊞ def recMousPos():⋯
157
158 ⊞ def on_press(key):⋯
239
240 ⊞ def on_release(key):⋯
320
321 ⊞ def recKeybPress():⋯
327
328   t0=threading.Thread(target=recMousList,args=())
329   t1=threading.Thread(target=recKeybPress,args=())
330   #t2=threading.Thread(target=recMousPos,args=())
331   t0.start()
332   t1.start()
333   #t2.start()
334   Thread.join(t0)
335   Thread.join(t1)
336   #Thread.join(t2)
```

Figure 4.2: Code for Data Collection 2

The format of the collected data is that of a python dictionary and it can be better understood from the figure 4.3.


```
{'user': 'rohan', 'charc': 0, 'lClick': 0, 'rClick': 0, 'mClick': 0, 'sTime': 0.0, 'space': 0, 'backSpace': 0, 'delete': 0,
'ctrl': 0, 'alt': 0, 'capsLock': 0, 'shift': 0, 'tab': 0, 'numLock': 0, 'enter': 0, 'mouseX': 633, 'mouseY': 629,
'scrollX': 0, 'scrollY': 0, 'timeDiff': 0}
{'user': 'rohan', 'charc': 0, 'lClick': 0, 'rClick': 0, 'mClick': 0, 'sTime': 0.015575408935546875, 'space': 0,
'backSpace': 0, 'delete': 0, 'ctrl': 0, 'alt': 0, 'capsLock': 0, 'shift': 0, 'tab': 0, 'numLock': 0, 'enter': 0, 'mouseX':
641, 'mouseY': 618, 'scrollX': 0, 'scrollY': 0, 'timeDiff': 0.015575408935546875}
{'user': 'rohan', 'charc': 0, 'lClick': 0, 'rClick': 0, 'mClick': 0, 'sTime': 0.015575408935546875, 'space': 0,
'backSpace': 0, 'delete': 0, 'ctrl': 0, 'alt': 0, 'capsLock': 0, 'shift': 0, 'tab': 0, 'numLock': 0, 'enter': 0, 'mouseX':
651, 'mouseY': 608, 'scrollX': 0, 'scrollY': 0, 'timeDiff': 0.0}
{'user': 'rohan', 'charc': 0, 'lClick': 0, 'rClick': 0, 'mClick': 0, 'sTime': 0.03689908981323242, 'space': 0, 'backSpace':
0, 'delete': 0, 'ctrl': 0, 'alt': 0, 'capsLock': 0, 'shift': 0, 'tab': 0, 'numLock': 0, 'enter': 0, 'mouseX': 663,
'mouseY': 600, 'scrollX': 0, 'scrollY': 0, 'timeDiff': 0.021323680877685547}
{'user': 'rohan', 'charc': 0, 'lClick': 0, 'rClick': 0, 'mClick': 0, 'sTime': 0.04489898681640625, 'space': 0, 'backSpace':
0, 'delete': 0, 'ctrl': 0, 'alt': 0, 'capsLock': 0, 'shift': 0, 'tab': 0, 'numLock': 0, 'enter': 0, 'mouseX': 672,
'mouseY': 594, 'scrollX': 0, 'scrollY': 0, 'timeDiff': 0.007999897003173828}
{'user': 'rohan', 'charc': 0, 'lClick': 0, 'rClick': 0, 'mClick': 0, 'sTime': 0.05289888381958008, 'space': 0, 'backSpace':
0, 'delete': 0, 'ctrl': 0, 'alt': 0, 'capsLock': 0, 'shift': 0, 'tab': 0, 'numLock': 0, 'enter': 0, 'mouseX': 681,
'mouseY': 590, 'scrollX': 0, 'scrollY': 0, 'timeDiff': 0.007999897003173828}
```


Figure 4.3: Sample Data

Next the collected data from various users is preprocessed using the 'preprocess.py' script. Refer figures 4.4 and 4.5. This script converts the collected data into a three dimensional matrix since, this is the form in which Tensorflow takes input. The three dimensional matrix is stored as '.NPY' files which are binary files.

```
import csv
import os
import ast
import re
import numpy as np
if not os.path.exists(os.path.dirname(r"final/")):
    os.makedirs(os.path.dirname(r"final/"))
if not os.path.exists(os.path.dirname(r"final/in/")):
    os.makedirs(os.path.dirname(r"final/in/"))
if not os.path.exists(os.path.dirname(r"final/out/")):
    os.makedirs(os.path.dirname(r"final/out/"))
if not os.path.exists(os.path.dirname(r"final/out/")):
    os.makedirs(os.path.dirname(r"final/out/"))
if not os.path.exists(os.path.dirname(r"final/test/")):
    os.makedirs(os.path.dirname(r"final/test/"))
users=[]

for filename in os.listdir('data/'):
    users.append(''.join(re.split("(_)", filename)[0:-4]))
users=sorted(np.unique(users))
#print(users)
#print(users.index('admin'))
filei=1
#pfi=open("final/in/pp_in_"+str(filei)+".txt","a")
#pfo=open("final/out/pp_out_"+str(filei)+".txt","a")
pfu=open("final/users.txt","a")
pfu.write(str(users))
timesteps=250
finalIn=[]
finalOut=[]
count=0
for filename in os.listdir('data/'):
    #print(filename)
    cOut=''.join(re.split("(_)", filename)[0:-4])
    #print(cOut)
```

Figure 4.4: Code for Preprocessing 1

```
    tempFile=[]
    for line in lines:
        dLine=ast.literal_eval(line)
        tempLine=[]
        for key,value in sorted(dLine.items()):
            tempLine.append(value)
        tempLine.pop(13)
        tempLine.pop(-1)
        tempFile.append(tempLine)
    for i in range(len(tempFile)-timesteps+1):
        outS=list(np.zeros(len(users)))
        #print(cOut)
        outS[users.index(cOut)]=1
        #print(outS)
        finalIn.append(tempFile[i:i+250])
        finalOut.append(outS)
        count+=1
        if count==1024:
            np.save("final/in/pp_in_"+str(filei),finalIn)
            np.save("final/out/pp_out_"+str(filei),finalOut)
            #pfi.write(str(finalIn))
            #pfo.write(str(finalOut))
            count=0
            finalIn=[]
            finalOut=[]
            #pfi.close()
            #pfo.close()
            filei+=1
            #pfi=open("final/in/pp_in_"+str(filei)+".txt","a")
            #pfo=open("final/out/pp_out_"+str(filei)+".txt","a")
#print(finalIn)
#print(finalOut)
if count!=0:
    np.save("final/in/pp_in_"+str(filei),finalIn)
    np.save("final/out/pp_out_"+str(filei),finalOut)
```

Figure 4.5: Code for Preprocessing 2

After this, the preproccesed data is fed into the 'train.py' file to train the LSTM model. Refer figures 4.6 4.7. The model consists of three layers of LSTM having 75 cells in each layer. cost is calculated using cross entropy and the cost is reduced using Adam Optimizer.

```
import tensorflow as tf
from tensorflow.contrib import rnn
import numpy as np
import ast
import os
#tf.reset_default_graph()

f3=open('final/users.txt','r')
#f4=open('final/test/pp_in.txt','r')
#tx=ast.literal_eval(f4.readline())
tx=np.load('final/test/pp_in.npy')
#f5=open('final/test/pp_out.txt','r')
#ty=ast.literal_eval(f5.readline())
ty=np.load('final/test/pp_out.npy')
users=ast.literal_eval(f3.readline())
num_classes = len(users) #output classes
users=tf.identity(users,name="users")

#learning_rate = 0.0001
training_steps = 1000
display_step = 1

num_input = 19 # number of features
timesteps = 250 # timesteps
num_hidden = 75 # cells in hidden layer
hidden_layers=2

X=tf.placeholder("float",[None,timesteps,num_input],name='input')
Y=tf.placeholder("float",[None,num_classes],name='output')

weights = {
    'out': tf.Variable(tf.random_normal([num_hidden, num_classes]))
}
biases = {
    'out': tf.Variable(tf.random_normal([num_classes]))
```

Figure 4.6: Code for Training 1

```
layer = rnn.BasicLSTMCell(num_hidden,forget_bias=1.0)
cell = rnn.MultiRNNCell([rnn.BasicLSTMCell(num_hidden) for _ in range(hidden_layers)])
#x=tf.unstack(X,axis=1)

output,state=tf.nn.dynamic_rnn(cell,X,dtype=tf.float32)

prediction=tf.matmul(output[:,-1],weights['out']) +biases['out']
prediction=tf.identity(prediction,name="prediction")
SM_prediction=tf.nn.softmax(prediction)
loss=tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=prediction,labels=Y))
optimizer=tf.train.AdamOptimizer()
train=optimizer.minimize(loss)
accuracy=tf.reduce_mean((Y*SM_prediction) + (1-Y)*(1-SM_prediction))

saver=tf.train.Saver()

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)

    saver.restore(sess,"checkpoint/here.ckpt")

    print("0",format(sess.run(accuracy, feed_dict={X: tx,Y:ty})))
    for i in range(training_steps):
        epocAcc=[]
        for j in range(len(os.listdir('final/in'))):
            #f1=open('final/in/pp_in_'+str(j+1)+'.txt','r')
            #f2=open('final/out/pp_out_'+str(j+1)+'.txt','r')
            #x=f1.readline()
            #x=ast.literal_eval(x)
            #y=f2.readline()
            #y=ast.literal_eval(y)
            x=np.load('final/in/pp_in_'+str(j+1)+'.npy')
            y=np.load('final/out/pp_out_'+str(j+1)+'.npy')
```

Figure 4.7: Code for training 2

```
            y=np.load('final/out/pp_out_'+str(j+1)+'.npy')
            for k in range(1):
                sess.run(train, feed_dict={X: x,Y:y})
            iterAcc=sess.run(accuracy, feed_dict={X: x,Y:y})
            print("Iteration:",j+1,format(iterAcc))
            epocAcc.append(iterAcc)
        epocAcc=np.array(epocAcc)
        printpred=sess.run(SM_prediction, feed_dict={X: tx})
        npp=np.array(printpred)
        print('\n')
        print(np.mean(npp[0:251,:],axis=0))
        print(np.mean(npp[251:502,:],axis=0))
        print(np.mean(npp[502:753,:],axis=0))
        print("\nEpoch:",i+1,format(sess.run(accuracy, feed_dict={X: tx,Y:ty})),np.mean(epocAcc),"\n\n\n")
        if i%1==0:
            saver.save(sess,"checkpoint/here.ckpt")
            #tf.saved_model.simple_save(sess,"model/",inputs={"X":X},outputs={"prediction":prediction})
    saver.save(sess,"checkpoint/here.ckpt")
    #tf.saved_model.simple_save(sess,"model/",inputs={"X":X},outputs={"prediction":prediction})


    #ans=sess.run(output, feed_dict={X: x})
    printpred=sess.run(SM_prediction, feed_dict={X: tx})
    npp=np.array(printpred)
    print(np.mean(npp[0:251,:],axis=0))
    print(np.mean(npp[251:502,:],axis=0))
    print(np.mean(npp[502:753,:],axis=0))
    print(sess.run(accuracy, feed_dict={X: tx,Y:ty}))
```

Figure 4.8: Code for Training 3

Finally the trained model is run using 'run_model.py'. Refer figures 4.9, 4.10, 4.11, 4.12, 4.13 and 4.14

```
import tensorflow as tf
from tensorflow.contrib import rnn
import numpy as np
import os
import win32gui
import threading
from threading import Timer
from threading import Thread
import pynput
from pynput import mouse
from pynput.mouse import Button
from pynput.keyboard import Key
from pynput import keyboard
import math
import time

cData=[]
cData.append([])



stime=time.time()
ltime=0
mData={}
tempData={}
#mData['user']=os.environ['USERNAME']

mData['charc']=0
mData['lClick']=0
mData['rClick']=0
mData['mClick']=0
#mData['sTime']=time.time()-stime
mData['space']=0
mData['backSpace']=0
mData['delete']=0
```

Figure 4.9: Code for Final Prediction 1

```
iloop=1

keyPrsd=[]
def on_click(x, y, button, pressed):
    global ltime
    global stime
    global iloop
    #print (button)
    if pressed:
        if button==Button.left:
            mData['lClick']=1
        if button==Button.right:
            mData['rClick']=1
        if button==Button.middle:
            mData['mClick']=1
    if not pressed:
        if button==Button.left:
            mData['lClick']=0
        if button==Button.right:
            mData['rClick']=0
        if button==Button.middle:
            mData['mClick']=0
    #print(mData['lClick'],mData['mClick'],mData['rClick'])
    mData['mouseX']=pynput.mouse.Controller().position[0]
    mData['mouseY']=pynput.mouse.Controller().position[1]
    #mData['sTime']=time.time()-stime
    if ltime==0:
        mData['timeDiff']=0
    else:
        mData['timeDiff']=time.time()-ltime
    ltime=time.time()
    cData[0].append(list(mData.values()))

```

Figure 4.10: Code for Final Prediction 2

```
94  def on_scroll(x, y, dx, dy):
95      global stime
96      global iloop
97      global ltime
98      #print('Scrolled {0}'.format((x, y)))
99      #print(dx,dy)
100     #mData['sTime']=time.time()-stime
101     if ltime==0:
102         mData['timeDiff']=0
103     else:
104         mData['timeDiff']=time.time()-ltime
105     ltime=time.time()
106     mData['scrollX']=dx
107     mData['scrollY']=dy
108     mData['mouseX']=x
109     mData['mouseY']=y
110     cData[0].append(list(mData.values()))
111     mData['scrollX']=0
112     mData['scrollY']=0
113
114     if iloop==0:
115         return False
116
117 def on_move(x,y):
118     global iloop
119     global ltime
120     mData['mouseX']=x
121     mData['mouseY']=y
122     #mData['sTime']=time.time()-stime
123     if ltime==0:
124         mData['timeDiff']=0
125     else:
126         mData['timeDiff']=time.time()-ltime
```

Figure 4.11: Code for Final Prediction 3

```
def recMousList():
    global mlistener
    # Collect events until released
    with mouse.Listener(
            on_click=on_click,
            on_scroll=on_scroll,
            on_move=on_move) as mlistener:
        mlistener.join()


def recMousPos():
    global iloop
    global stime
    global ltime
    while iloop:
        tempData=mData.copy()
        mData['mouseX']=pynput.mouse.Controller().position[0]
        mData['mouseY']=pynput.mouse.Controller().position[1]
        if tempData!=mData:
            #mData['sTime']=time.time()-stime
            if ltime==0:
                mData['timeDiff']=0
            else:
                mData['timeDiff']=time.time()-ltime
            ltime=time.time()
            cData[0].append(list(mData.values()))

    return False


def on_press(key):
    global stime
    global ltime
```

Figure 4.12: Code for Final Prediction 4

```
146 def recMousPos():
147     global iloop
148     global stime
149     global ltime
150     while iloop:
151         tempData=mData.copy()
152         mData['mouseX']=pynput.mouse.Controller().position[0]
153         mData['mouseY']=pynput.mouse.Controller().position[1]
154         if tempData!=mData:
155             #mData['sTime']=time.time()-stime
156             if ltime==0:
157                 mData['timeDiff']=0
158             else:
159                 mData['timeDiff']=time.time()-ltime
160             ltime=time.time()
161             cData[0].append(list(mData.values()))
162 
163     return False
164 
165 
166 def on_press(key):
167     global stime
168     global ltime
169 
170     if isinstance(key,pynput.keyboard._win32.KeyCode) and key not in keyPrsd: ···
184     else:
185         if key not in keyPrsd: ···
236 
237 
238 
239 def on_release(key): ···
308 
```

Figure 4.13: Code for Final Prediction 5

```
316
317 def predictModel():
318     global iloop
319     with tf.Session(graph=tf.Graph()) as sess:
320         tf.saved_model.loader.load(sess,[tf.saved_model.tag_constants.SERVING],"model/")
321         t_X=tf.get_default_graph().get_tensor_by_name('input:0')
322         t_prediction=tf.get_default_graph().get_tensor_by_name('prediction:0')
323         t_users=tf.get_default_graph().get_tensor_by_name('users:0')
324         f_prediction=tf.nn.softmax(t_prediction)
325         users_list=sess.run(t_users)
326         for i in range(len(users_list)):
327             users_list[i]=users_list[i].decode('utf-8')
328         outp=[]
329         while iloop:
330             if len(cData[0])>=250:
331                 tempcData=[]
332                 tempcData.append([])
333                 for i in range(len(range(250))):
334                     tempcData[0].append(cData[0][i])
335                 #print(users_list)
336                 outp.append(sess.run(f_prediction,feed_dict={t_X:tempcData}))
337                 cData[0].pop(0)
338             if len(outp)>=1000:
339                 npoutp=np.array(outp)
340                 npm=np.mean(npoutp[0:1000,:],axis=0)
341                 print(npm)
342                 print(users_list[npm.argmax()])
343                 outp.pop(0)
344 t0=threading.Thread(target=recMousList,args=())
345 t1=threading.Thread(target=recKeybPress,args=())
346 t2=threading.Thread(target=predictModel,args=())
347 t0.start()
348 t1.start()
349 t2.start()
```

Figure 4.14: Code for Final Prediction 6

4.1 Tools and Technology

Programming Language:- Python

Library/ Platform:- Tensorflow, Numpy, Pynput, Threading

IDE:- VS Code

4.1.1 Tensorflow

Tensorflow is an opensource deep learning framework maintained by Google. It supports parallelism and supports GPUs of various manufacturers. Tensorflow can be considered as a library for data flow programming. It is widely used for building neural network models. It is highly optimized and though Python is the mostly used as a language for working with Tensorflow, it is actually built on C++ which is very vast.Tensorflow supports a wide range of complex mathematical operations, which makes coding of engineering tasks a lot easier.

4.1.2 Pynput

Pynput is the python library that allows the programmer to monitor and control the input devices connected to a computer. It can be used to monitor the position of the mouse, the clicks of the mouse or the keys pressed of the keyboard. It also allows the user to control the input devices like triggering a mouse click, moving a mouse to a particular location or getting key inputs from the keyboard.

4.1.3 Threading

The python library threading provide multiprogramming capabilities. This allow for multiple tasks to be run simultaneously. This library was particularly useful for this project as it allowed me to both monitor the computer inputs and perform other necessary operations. Due to this library, in this project data from multiple input sources like mouse and keyboard can simultaneously monitor . Also using this libary, the user can be verified in real time by collecting the usage data of the user and running the tensorflow model simultaneously to make predictions.

4.1.4 Numpy

Numpy is a Python library for scientific computing. It has a built-in array processing package and is one of the single greatest Python libraries. A lot of important python

libraries wouldn't be possible without Numpy. It very helpful for working on large multi-dimensional arrays and supports various functions for working on them. In this project, Numpy library ahs been used to store large three dimensional matrices to files which are stored as binary files hence they can be processed much master and much more efficiently than a csv files.

4.2 System Configuration

Operating System:- Ubuntu

OS Type:- 64-bit operating system

Processor:- Intel(R) Core(TM) i7

RAM:- 6 GB

Graphics:- Tesla K40

4.3 Training

The training was done with a two layer LSTM with each Layer consisting of 75 cells. The following is the accuracy on the testing set after each Epoch. (Note: 0th epoch means before the training has started.)

Table 4.1: Testing Accuracy

Epoch	Testing Accuracy
0	0.536982536315918
1	0.5560380220413208
2	0.5729376077651978
3	0.5874494910240173
4	0.5656393766403198
5	0.5760141611099243
6	0.59056156873703
7	0.580303430557251
8	0.6012371182441711
9	0.6617692112922668
10	0.6885388493537903
11	0.6527553796768188
12	0.7049592137336731
13	0.6984983086585999
14	0.7306012511253357
15	0.7366735339164734
16	0.7202929258346558

4.4 Graphical Result

Below we can see outputs of three batch of training data. Each row is a batch of training data with three classes. The rectangles denote the correct outputs for each batch. As we can see that the answer of the first batch is wrong the out output of the second and third batches is correct.

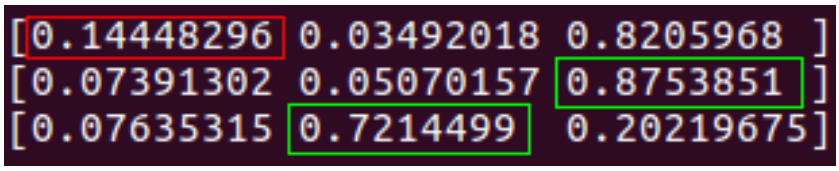

Figure 4.15: Graphical Output

Chapter 5

Conclusion and Future Work

The proposed model does give reasonable results but only if the usage of the users is not too similar to each other. If their usage is similar, model will be able to predict the user to a particular class. There are many ways the model can be improved, such as more features can be added like the time of the day or the time since the program has started. But any increase in the number of features will demand an increase in the training data set.

In the future, a better feature set will be implemented which will include data that will help in training the model better. Also, another feature will be added such that if the user cannot be verified by the model, the system security will be raised so that the unknown user cannot do much harm.

Bibliography

[1] J. Peng, K.-K. R. Choo, and H. Ashman, "User profiling in intrusion detection: A review," *Journal of Network and Computer Applications*, vol. 72, pp. 14 – 27, 2016.

[2] A. Pentel, "Emotions and user interactions with keyboard and mouse," 08 2017.

[3] T. Wesoowski and P. Kudacik, "User profiling based on multiple aspects of activity in a computer system," vol. 23, p. 121, 10 2014.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[5] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015.

[6] T. Wesoowski and P. Porwik, "Keystroke data classification for computer user profiling and verification," 09 2015.

[7] Y. L. B. W. Z. G. H. L. D. C. Yu Zhu, Hao Li, "What to do next: Modeling user behaviors by time-lstm," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 3602–3608, 2017.

[8] A. Cufoglu, "Article: User profiling - a short review," *International Journal of Computer Applications*, vol. 108, pp. 1–9, December 2014. Full text available.

[9] S. Kanoje, S. Girase, and D. Mukhopadhyay, "User profiling trends, techniques and applications," vol. 1, pp. 2348–4853, 11 2014.

[10] T. Wesoowski and P. Porwik, "Computer user profiling based on keystroke analysis," 01 2016.