IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 4, April 2021

DOI 10.17148/IJARCCE.2021.10464

Productivity Tracker Assistant

Prof. Vikrant A. Agaskar[1], Shubham Gargade[2], Rohan Jain[3]

Professor, Computer Department, Vidyavardhini's College of Engineering and Technology, Vasai, India[1]

Student, Computer Department, Vidyavardhini's College of Engineering and Technology, Vasai, India[2]

Student, Computer Department, Vidyavardhini's College of Engineering and Technology, Vasai, India[3]

Abstract: In today's world the time which people spend on device while working is to do productive work but many people get into the trap of wasting the time in using social media, entertainment, etc and after a few hours they recognise that the time has been wasted unconsciously by doing unproductive things.Since every work is moving towards the digital solution, which will lead many of the human existence to use the computer or mobile to get there work done. Hence it is necessary to have a monitor of activities that has been performed to avoid the unproductive time spent on their device. By taking this problem into consideration, this paper proposes Python, Deep Learning, Natural Language and Cloud Computing Frameworks for automating the tasks of classifying user activities into productive and nonproductive categories which has 13 subcategories which can be shown to the user in various statistical and tabular formats which will help user able to identify its activities as productive and non productive along with the time spent on the same. The user will be able to visualize its unproductive activities and can become conscious which will help the user to stop doing unwanted activities on desktop or mobile phone and start doing what the user actually wanted to do. For automating these tasks, along with the frameworks mentioned above, we have built a Desktop Application which will act as a productivity assistant for the user on a Desktop but the same can be built on an Android Application as well.

Keywords: Machine Learning, Productivity Assistant, Cloud Computing, Natural Language Processing, Cosine Similarity Algorithm, Desktop Application, Android Application, Electron, Algorithmia, Firebase, Text Classification, Voice Assistant, Website Classification, Chart.js.

I. INTRODUCTION

In this modern world of mobile phones and desktops, most of the people find these digital devices as a source of entertainment and sometimes unconsciously spend lots of time watching unnecessary youtube or facebook videos or play games for hours and then recognize that they have spent their time unproductively. To overcome this pitfall, we have proposed an idea of monitoring user activities and showing the reports to the user after classifying these activities and for that we have developed a Desktop Application for tracking user activities on Desktop, but the same can be achieved on an Android Application as well.

II. OBJECTIVE

1. Monitoring activities that have been performed, this project will help them to identify the productive and unproductive time they spent on the desktop.
2. The activities of the user along with the time spent on each activity will be tracked and the tasks will be classified into various classes using machine learning techniques, and reported to the user as productive or unproductive tasks.
3. We will be using a Text Classifier model which will process natural language and will provide the tracking details by voice assistant or text based assistant.
4. A desktop application is made through which the user will be able to understand the activities performed.

III. RESEARCH METHODOLOGY

1. We have researched methods for website classification, and we have found that it can be done in two ways, i. URL classification and ii. Web page classification.
2. In URL classification the url of the website is taken and prediction of the url is done into various classes using a machine learning model. URL classification is preferable when (i) when speed is crucial,(ii) to enable content filtering before an (objectionable) web page is downloaded, (iii) when a page's content is hidden in images,(iv) to annotate hyperlinks in a personalized
3. web browser, without fetching the target page, and (v) when a focused crawler wants to infer the topic of a target page before devoting bandwidth to download it
4. URL classification is faster than other classification but the efficiency and accuracy for many cases will be less because only website url cannot identify the main objective of the webpage.For eg: amazon.com is the url for e-commerce

website but when url fetches this it may recognize it as a amazon forest, hence there can be many such cases, to avoid this we are using web-page classification, in this method we will be actually fetching the content of the website and parsing its html content, by this we will be getting the entire motive of the web page and this content is passed to the machine learning model for classification.Web page classification though take more time to fetch the content but the accuracy is higher in many cases than url classification.

5. For building an application, electron is used in which we can use the actual web technologies like HTML,CSS, Javascript to build a Desktop Application. For tracking python is used and the application is connected with the backend using npm python-shell module.The main process will be the electrons main.js and visible renderer process will be the frontend part i.e electron application, and the application will also include hidden renderer processes, for dynamic updates and fetching data of tracking in the background.Also the python scripts will run as a background process in electron for dynamically storing user tracking activities in the background. The communication between visible processes or between a visible process and a hidden process is done via the main process. The actual details of tracking will be stored dynamically on the firebase real time database.Similarly for showing results, reports and tracking details to the user, firebase listeners are used which will listen to the changes in the database and dynamically update the respective changes in the application.

6. Python is used as a backend technology for tracking user activities using which it will give the required information of a particular software or a website along with the time spent on that active activity. This software or website activity will then be given to the machine learning text classifier model which will classify this activity into 13 predefined categories out of which the one with the highest value will be selected as the classified category for that activity(application or website), and this classes are further classified into Productive and Unproductive category. Once this activity is classified into the required category, then that activity along with its information and time spent is stored in the Firebase Realtime Database which gets dynamically updated on the user end in the form of tables and/or statistical data.

7. The user can also ask questions to the voice assistant or a text assistant in the form of speech or text respectively, about the tracked activities. The voice and text assistant is not yet added to the application as it is an additional feature which has yet to be compiled successfully as per the requirements of the application.

IV. SYSTEM DESIGN

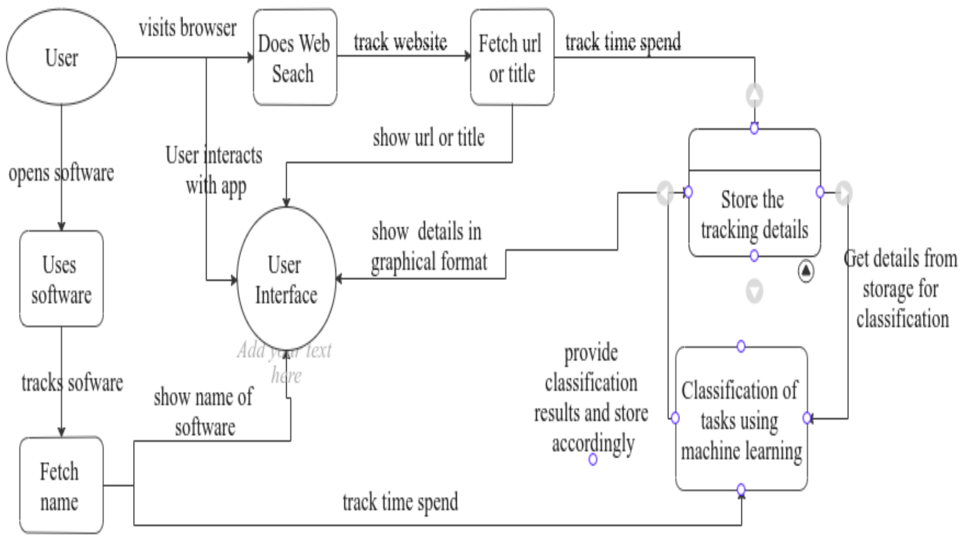


Fig. 1 Application System Design

V. PROPOSED ALGORITHM AND IMPLEMENTATION

1. We have used a deep learning pretrained embedding model for training and classification purpose. Pretrained embedding model is a key to achieve higher accuracies as it is already trained on millions of dataset. The embedding model is available on tensorflow hub which is a repository of trained machine learning models. The embedding model normalizes the text data and converts it into embeddings which is a numerical vectorized format of text which helps in training the weights of the model and perform inferences on an actual input.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 4, April 2021

DOI 10.17148/IJARCCE.2021.10464

2. For knowing which activity of the user has to be classified, we have used an algorithm which will track the current active window of the user which can be the software window or a website window. These can be achieved using a python packages like uiautomation, win32gui, win32process, and psutil which will help get the current active window of the user and get the information of that window to classify that into productive and unproductive categories, we can also calculate time tracked on the window simultaneously and lastly store everything in firebase using python wrapper of firebase called Pyrebase, and show it to the user in the Frontend using Firebase web.
3. For software classification, we can get the name of that software along with other details using above python packages required for classification.
4. For website classification, we have used web page categorization which generally takes the whole html document as an input and performs predictions based on this input. But in our application, instead of downloading the whole html document which is time consuming, we have just used the title and description of the website for categorization which will decrease the processing time of the machine learning model while classification. But for getting only the title and description of the website, there is a time consuming and an inefficient process of downloading the whole html document and parsing the document for fetching title and description. So, to overcome this inefficient process, instead of downloading the whole html document, we can download it in chunks of bytes using python html parsers to stream the html document in chunks. Once the title and description is fetched, then there is no need to download the further web page which decreases the download and fetching time of the application.
5. Once we get the required text data to be classified, the data is sent to the cloud platform named Algorithmia which is an Algorithm as a Service(AAAS) which is a cloud service to write algorithms(mainly ML Algorithms) which can be called from a remote application. The ML model for classifying software apps and websites is running on this cloud service and takes the data for classification as an input and sends the predicted response in the form of JSON.
6. This predicted output along with the time spent is stored in Firebase using Firebase Realtime Database. We are using Python to store data in Firebase. Pyrebase is a Python wrapper for firebase. Firebase Web is used to fetch data in the frontend
7. Using Firebase Event Listeners, we can trigger events in a remote application whenever a new data is added in the firebase and hence can show the updated reports to the user without any refresh needed from the user end.
8. The data can be presented to the user in various ways depending on the application developer. The formats to show the reports that we have used are tabular and statistical formats. To show the data in statistical format, we have used Chart.js.
9. The important data of the users activities can be stored in the users tracking history and can be shown to the user so that the user can get the intuition of how productive or unproductive he/she was in the past.
10. For building a Jarvis Assistant for the application, which acts as both a Voice and a Text based Assistant, we have used Google Natural Language Processing API to convert speech to text. The api is available in the python speech_recognition package. So it can be used on successfully installing the speech_recognition package. Further, our assistant works as a Question Answering System which takes input as a question from the user which can be a question asked to know the information like time spent or how productive the user is related to tracked activities. First this question which is in the form of speech is converted to text using google speech recognition api. Once the text is generated, it is given to an Algorithm which will match the input question to the list of questions stored in the database and the answer to the corresponding matched question will be provided to the user after fetching the required answer from firebase. For the input question validation and matching, we have used the Cosine Similarity Algorithm which matches an input string with the given list of strings and produces a list of probabilities for each string in the database. The one with the highest probability is selected as the question asked by the user and the corresponding answer is provided to the user. Similarly for text assistant, the input can be in the form of text which requires only question validation and question matching to be performed.

VI. FLOW CHARTS

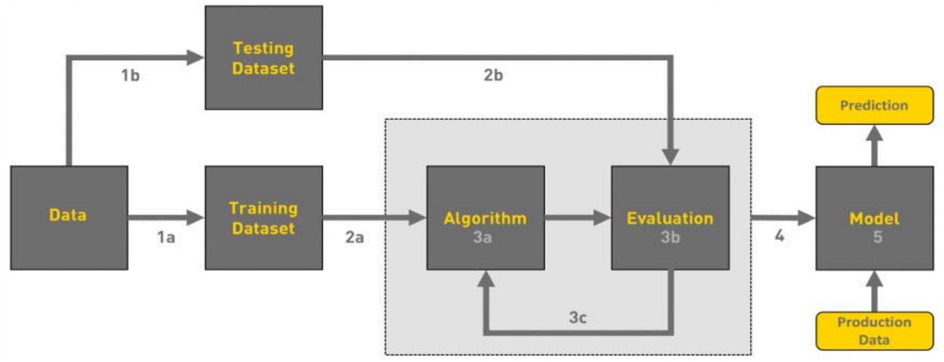


Fig. 2 General ML Workflow

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 4, April 2021

DOI 10.17148/IJARCCE.2021.10464

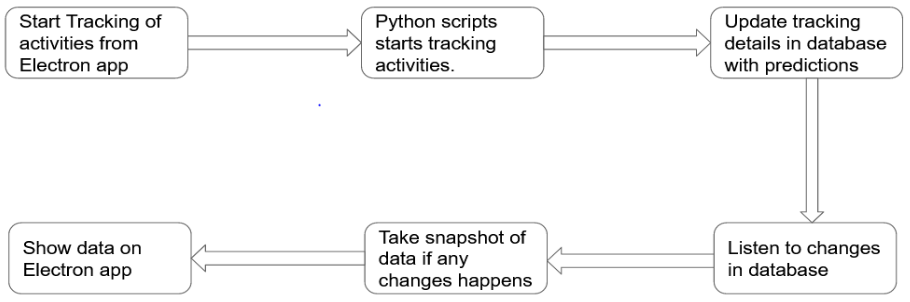

Start Tracking of activities from Electron app
Python scripts starts tracking activities.
Update tracking details in database with predictions
Listen to changes in database
Take snapshot of data if any changes happens
Show data on Electron app

Fig. 3 Electron application with Firebase Realtime Database

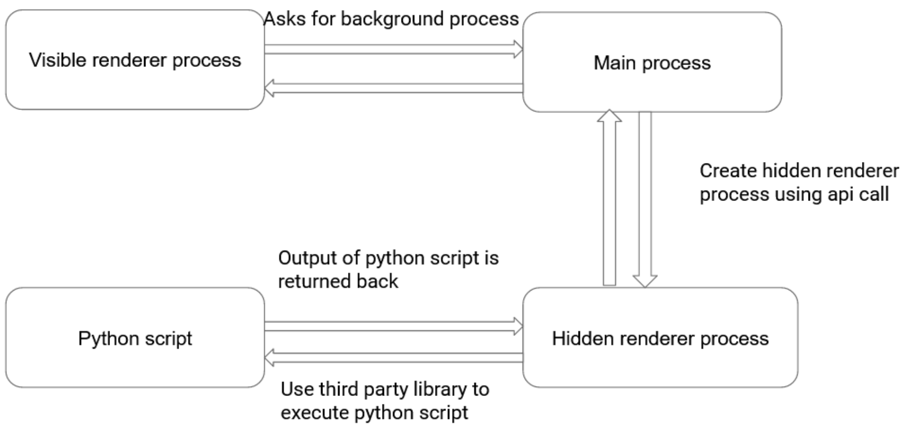

Visible renderer process
Asks for background process
Main process
Create hidden renderer process using api call
Output of python script is returned back
Python script
Hidden renderer process
Use third party library to execute python script

Fig. 4 Electron Application with Python scripts running in background

VII. WORKING

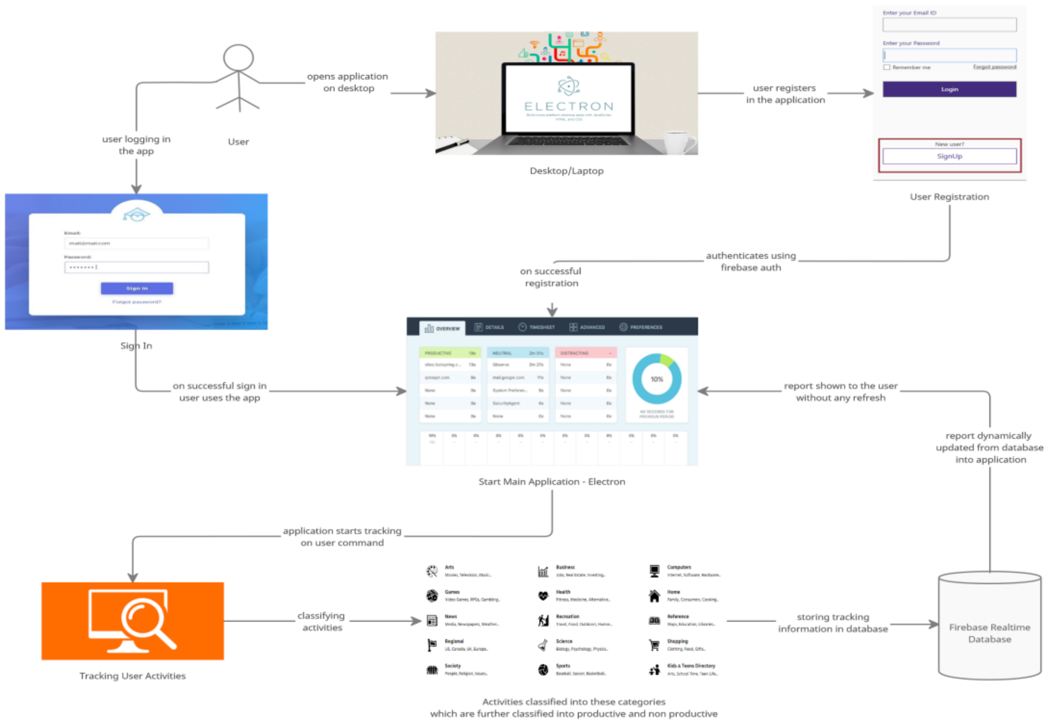

User
opens application on desktop
Desktop/Laptop
user registers in the application
User Registration
user logging in the app
Sign In
authenticates using firebase auth
on successful registration
on successful sign in user uses the app
Start Main Application - Electron
report shown to the user without any refresh
report dynamically updated from database into application
application starts tracking on user command
Tracking User Activities
classifying activities
storing tracking information in database
Firebase Realtime Database
Activities classified into these categories which are further classified into productive and non productive

Fig. 5 Detailed Working of Application

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 4, April 2021

DOI 10.17148/IJARCCE.2021.10464

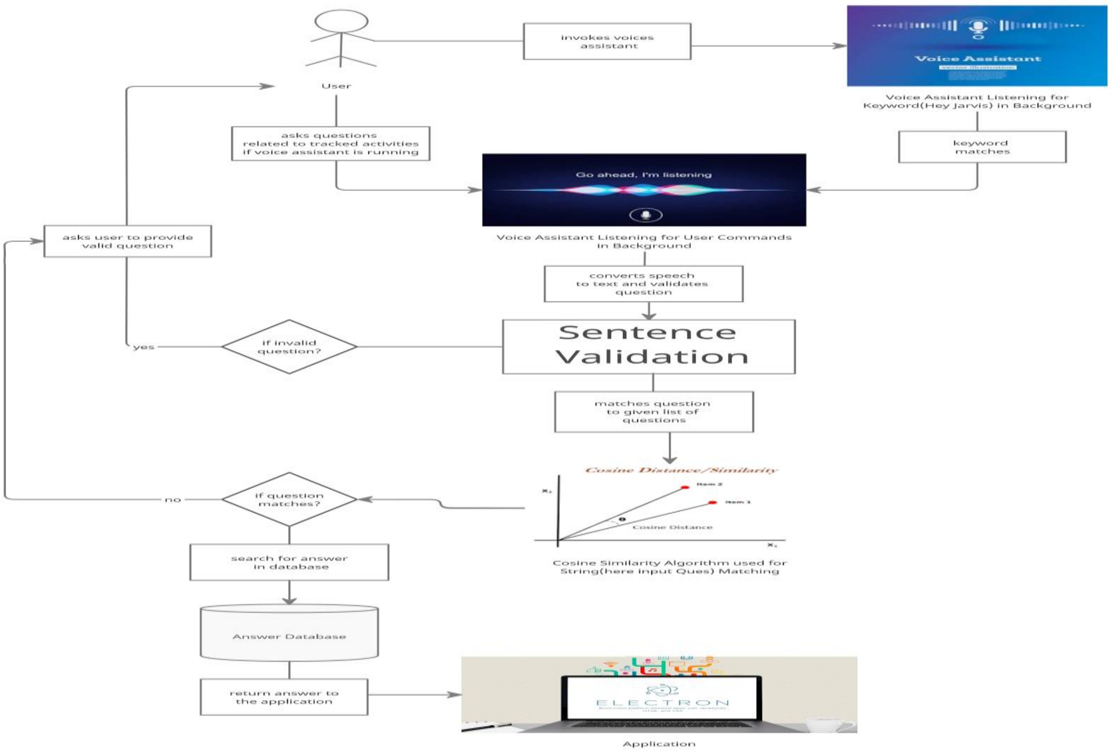
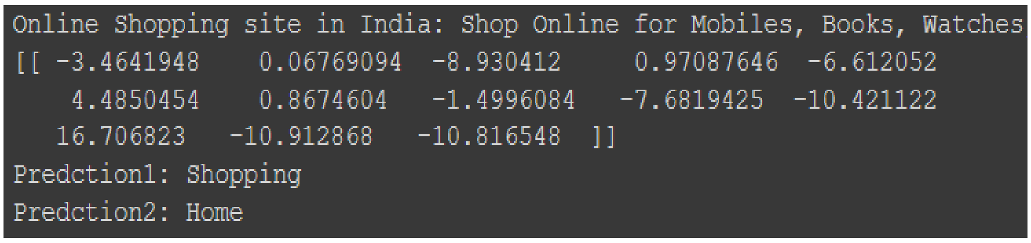

User
invokes voices assistant
Voice Assistant
Voice Assistant Listening for Keyword(Hey Jarvis) in Background
keyword matches
asks questions related to tracked activities if voice assistant is running
Go ahead, I'm listening
Voice Assistant Listening for User Commands in Background
asks user to provide valid question
converts speech to text and validates question
yes
if invalid question?
Sentence Validation
matches question to given list of questions
Cosine Distance/Similarity
Item 2
Item 1
Cosine Distance
Cosine Similarity Algorithm used for String(here input Ques) Matching
no
if question matches?
search for answer in database
Answer Database
return answer to the application
ELECTRON
Application

Fig. 6 Detailed Working Of Jarvis Assistant

VIII. RESULT

1. The user can monitor its activities for the current day and can get the productivity reports in various statistical formats for all the tracked activities for the current day and can also get reports of the previous seven days that were tracked.
2. The user will be able to start and stop the tracking at any point of time.
3. The application can be used by any type of user who wants to know about the daily activity reports performed on software or website on a desktop, or a user who unconsciously spends time on unproductive activities and wants to use its time efficiently while using the desktop.
4. The application will provide separate reports for software activities, website activities and overall total tracking activities.
5. The user will get the time spent on each website or software along with its title or name, the time spent can also be shown in terms of category, for eg: time spent in playing games, or time spent in business category.



```
Online Shopping site in India: Shop Online for Mobiles, Books, Watches
[[ -3.4641948    0.06769094  -8.930412     0.97087646  -6.612052
    4.4850454    0.8674604   -1.4996084   -7.6819425  -10.421122
   16.706823   -10.912868   -10.816548   ]]
Predction1: Shopping
Predction2: Home
```

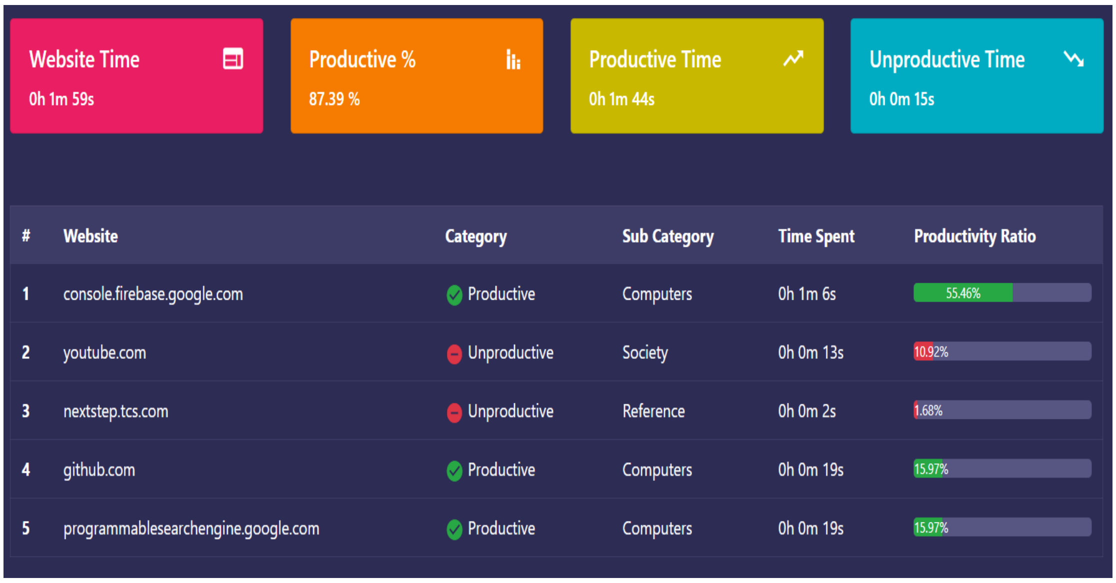
Fig. 7 Model Prediction Of Amazon website on Google Colab

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 4, April 2021

DOI 10.17148/IJARCCE.2021.10464



Website Time
0h 1m 59s

Productive %
87.39 %

Productive Time
0h 1m 44s

Unproductive Time
0h 0m 15s

#	Website	Category	Sub Category	Time Spent	Productivity Ratio
1	console.firebase.google.com	✓ Productive	Computers	0h 1m 6s	55.46%
2	youtube.com	● Unproductive	Society	0h 0m 13s	10.92%
3	nextstep.tcs.com	● Unproductive	Reference	0h 0m 2s	1.68%
4	github.com	✓ Productive	Computers	0h 0m 19s	15.97%
5	programmablesearchengine.google.com	✓ Productive	Computers	0h 0m 19s	15.97%

Fig. 8.1 User Interface-Website Tracked Data-Tabular Format

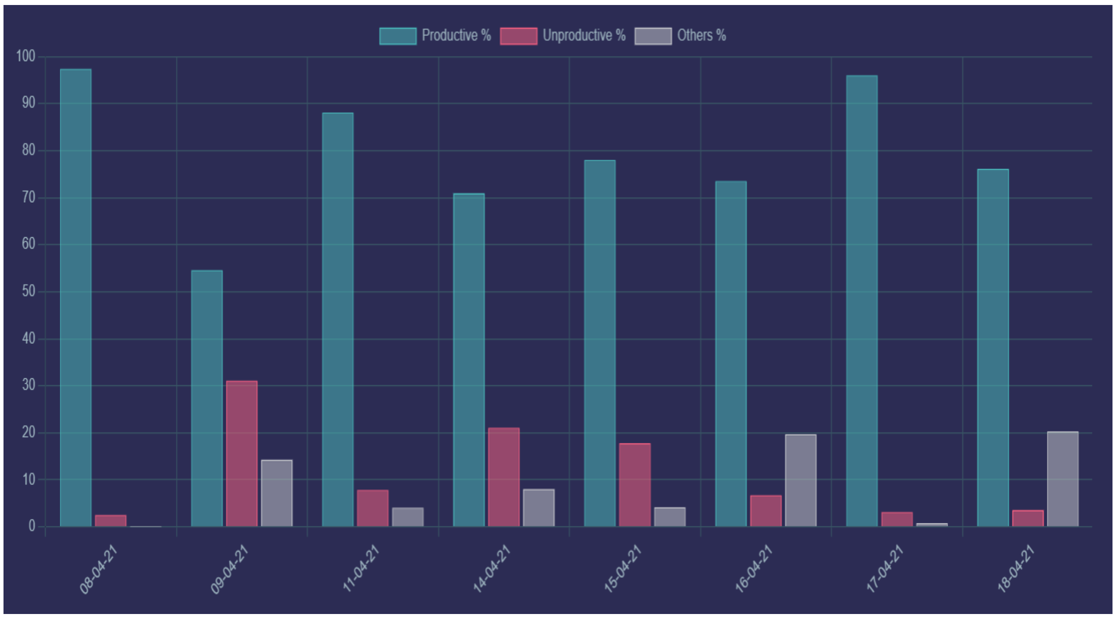


Fig. 8.2 User Interface-Past Data-Statistical Format

IX. GAP ANALYSIS

DESIRED SITUATION	CURRENT SITUATION
The activity classifier model will produce accurate results.	Currently the accuracy of the model is 80%.
The model has to be deployed on cloud for users and locally on docker for testing purposes.	The model is deployed on cloud and locally on docker successfully.
The user will get real-time updates of the tracked activities	The user is getting real-time updates of the tracked activities.

The user will also get productivity details of the past seven days.	The user is getting the details of the past seven
Apart from providing tracked activities information in various statistical formats, the user can also get that information using a voice or text assistant.	Currently there are some latency issues in the voice assistant, but is implemented and has to be embedded in the application.
The user can change any category(eg: Games, Health etc.) into productive or unproductive based on user needs.	Currently working on it.

X. BENEFITS TO THE SOCIETY

1. As in today's world most of the people spend their time on their desktop to do productive work, but while working there may be the case that the user will be distracted from the work and unconsciously do the unproductive work. Hence tracking the activities will show users how much productive or unproductive work is performed by them.
2. It will be highly beneficial for children and teenagers as they might get distracted many times in other unproductive tasks on the desktop.
3. As it tracks each and every activity on the desktop it might also be helpful for parents to monitor their children's activities.
4. Similarly the application if built for management it will be helpful for the organization to monitor the activities of their employees.

XI. FUTURE SCOPE

1. In future voice and text assistant also can be used to show the result of tracking.
2. The user will be allowed to choose the classes of tracking (arts, sports, education, health, science etc.) into productive or unproductive categories according to them.
3. The application is currently storing past productivity details till the last seven days, in future it can be upgraded to store history of several months.
4. In future it can also be made for tracking the activities on android devices.

REFERENCES

[1]. Web page classification: a survey of perspectives, gaps, and future directions,Mahdi Hashemi, 2020
[2]. Website Classification using Python. https://towardsdatascience.com/industrial-classification-of-websites-by-machine-learning-with-hands-on-python-3761b1b530f1, 13 July 2018
[3]. Url classification using DMOZ dataset, Utsav, 2019
[4]. Tensorflow Hub. https://www.tensorflow.org/hub
[5]. Electron Python BoilerPlate, Aakash Malik, 2019
[6]. Pyrebase. https://github.com/thisbejim/Pyrebase, James Childs, 2016
[7]. Electron. https://medium.com/heuristics/electron-react-python-part-1-introduction-b228ccf8e889
[8]. Firebase. https://firebase.google.com
[9]. Algorithmia. https://algorithmia.com
[10]. Docker. https://docker.com
[11]. PythonShell. www.npmjs.com/package/python-shell
[12]. Python Electron. www.techiediaries.com/python-electron-tutorial/?utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com
[13]. Chart.js. https://www.chartjs.org/
[14]. Speech Recognition Using Deep Neural Networks: A Systematic Review, Ali Bou Nassif; Ismail Shahin, 2019
[15]. Speech Recognition using Python. https://pypi.org/project/SpeechRecognition/
[16]. The Ultimate Guide To Speech Recognition With Python. https://realpython.com/python-speech-recognition/#author, David Amos
[17]. How to build your own AI personal assistant using Python. https://towardsdatascience.com/how-to-build-your-own-ai-personal-assistant-using-python-f57247b4494b, M.Mirthula, 2020
[18]. How to Compute the Similarity Between Two Text Documents? https://www.baeldung.com/cs/ml-similarities-in-text, Francesco Elia, 2020