**UI Automation using Image Processing**

Karunakar Gonnabathula

**-----------------------------------------------------------------ABSTRACT------------------------------------------------------------**

**UI automation using image processing. Can say as an automation approach with human perception.**

**I have used a deep learning models to identify and classify the screen as a human.**

**Keywords: Deep Learning, Image Processing, Automation.**

**-**-------------------------------------------------------------------------------------------------------------------------------

Date of Submission: JULY 12, 2019

**-**-------------------------------------------------------------------------------------------------------------------------------

**Abbreviations:**

ML: Machine Learning,

AI: Artificial Intelligence,

OD- module: Object detection Module,

C-module: Classifier Module,

NLTK – Natural Language Toolkit

# INTRODUCTION

What does a human do to perform actions on UI? In general, the human brain captures an image of the screen

Process it and then identifies the necessary action that need to be performed. All these happens in some Nano seconds. Similar processor can be achieved through a machine learning model. Our model needs to be the brain of the process to identify and classify the objects to perform actions. To understand what exactly I mean let’s go through the use cases we picked up.

# Use Case -1 (Login to some web portal)

What a human does to logon to some web portal. He starts opening his browser in his desktop, enter the URL and proceed logging in by entering the credentials. What can be the tasks of the human brain to perform in this process.

1. Capture the screen and identify the browser. Go and click on it to open, wait for the browser to open.
2. Proceed entering the URL and wait for the page to load
3. Identify the inputs. fill them with the appropriate values and proceed to login.

**Start**  elseIf Chrome Opens Returns the coordinates of the fields

Returns the Chrome co-ordinates If page loads then

**W**e can see it as a two-layer architecture and there’s an alternate communication between the two layers. As there is between the hand and brain of a human. We can name ours layers as ML-model (brain) and Python script (Hands) of the process.

# Modules in ML-model (Brain) Our ML-model has two modules 1. OBJECT DETECTION MODULE 2. CLASSIFICATION MODULE Object Detection Module: - This unit of our model accepts the captured screen, process it and returns back the elements required for action.

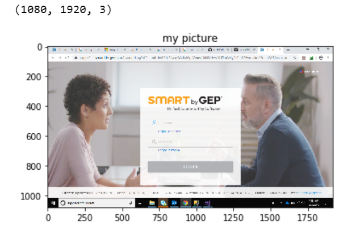
**Array of**

**Co-ordinates of Detected Objects**

**Image**

**OD-MODULE**

**Example:**

****

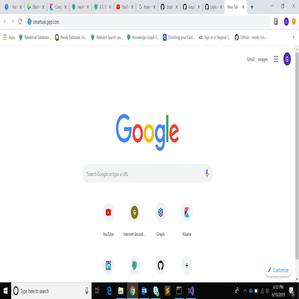
****

**Classification Module: -** This unit of the module accepts the captured screen and classifies the screen to desired classes

**Image Returns the class**

**C-MODULE**

**Example: -**

****

**Loading page**

# Modules in Python-script (Hands)

Our Python scripting is divided to sub modules.

1.Automation Design

**Automation Design: -**

Contains the functions need to be triggered and called through parameters for keyboard and mouse events including signalers and communication objects to ML model.

**Now let’s discuss in brief the functioning and flow in use case-1**

**Start**  If Chrome Opens Returns the coordinates of the fields

Returns the Chrome co-ordinates If page loads then

# In this process as you start the

# Process flow module knows to open chrome. this triggers the automation module, and this captures a screen shot and passes it to the OD-module for chrome co-ordinates. Migrates to chrome icon and double click on it.

1. Now the process flow needs to proceed entering URL if chrome opens up. So, it triggers automation module to check if the chrome status. This in turn communicates with C-model for verification.
2. Then it proceeds for filling the form. This triggers automation module to get the co-ordinates of the input fields of the form from OD-model
3. Then it is followed by filling the inputs and submitting the form.

# Control Layer

**Process Flow Design: -**

Scripting includes the step by step flow. A third layer above all to control the process of the use case

**Open chrome 🡪 Enter URL 🡪 enter credentials 🡪 submit the form**

# Basic Tree Structure: -

Mouse, Keyboard Events

Trained Classifiers

Trained detectors

It’s a three-layer architecture. This has been to for the scalability of deployment. Each component of the layer communicates and triggers the components in other layers as explained in above 3 sections

**Use case -1 Tree Structure: -**

1. Chrome Detector
2. Form input Detector

Mouse, Keyboard Events

1. Browser loading Classifier
2. Form Page Loading Classifier

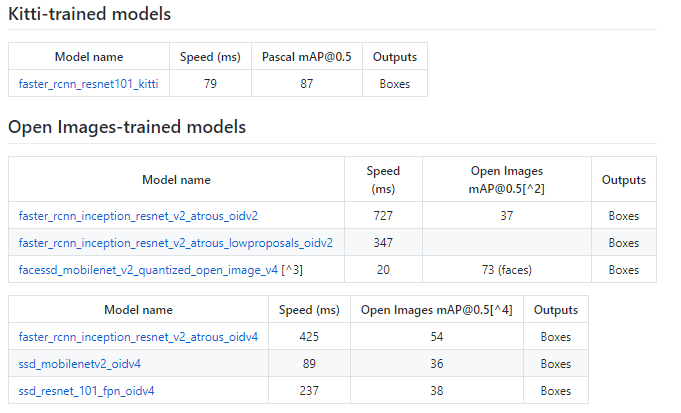
Now let’s go further knowing about the models and libraries used in the OD, C modules.

# OD – Module: -

We have used Google’s TensorFlow Object detection API in our model (Current stable version 2.0)

The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.

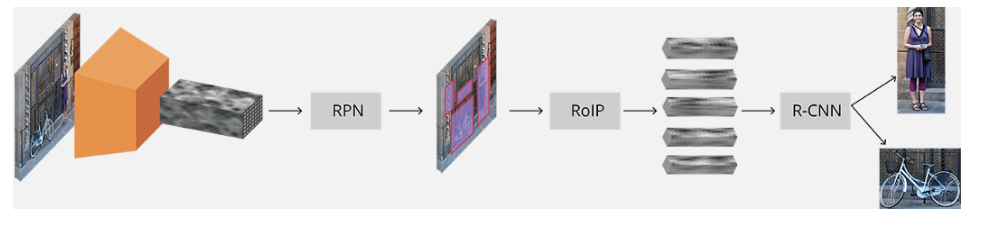
They  provide a collection of detection models pre-trained on the [COCO dataset](http://mscoco.org/), the [Kitti dataset](http://www.cvlibs.net/datasets/kitti/), the [Open Images dataset](https://github.com/openimages/dataset). These models can be useful for out-of-the-box inference if you are interested in categories already in those datasets. They are also useful for initializing your models when training on novel datasets.





After few researches and based on the available environment, we concluded using the Faster -RCNN-Inception-V2-COCO model give slower detection but with more accuracy

**Architecture of Faster -RCNN-Inception-V2-COCO**



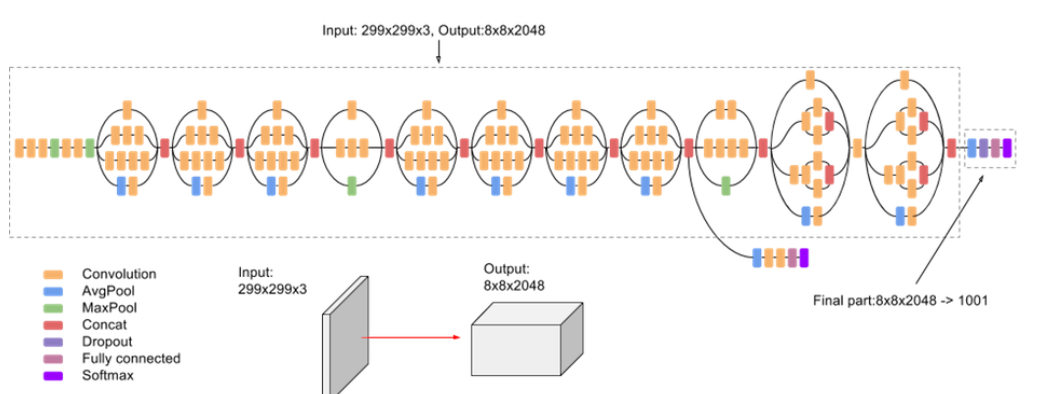
Further details: - <https://medium.com/object-detection-using-tensorflow-and-coco-pre/object-detection-using-tensorflow-and-coco-pre-trained-models-5d8386019a8>

# C-Module: - we used TensorFlow-hub pre-trained models for image classification. The model we picked for classification is Inception V3

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, con-cats, dropouts, and fully connected layers. Batch norm is used extensively throughout the model and applied to activation inputs. Loss is computed via SoftMax.

**A high-level diagram of the model is shown below:**



To know more: - <https://cloud.google.com/tpu/docs/inception-v3-advanced>

# Automating Functions: -

We used **Keyboard** and **PyAutoGUI** libraries of python to automate the keyboard and mouse events.

PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.

We have several functions that are triggered by the python scripts

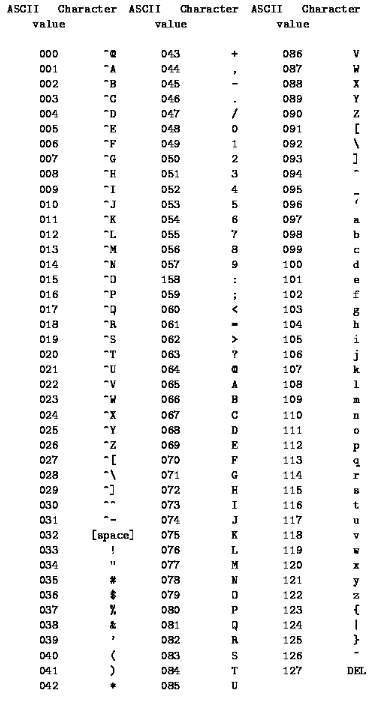
1. Move Mouse: - move\_mouse (X, Y) moves the mouse to specified x, y co-ordinates.
2. Click Mouse: - click\_mouse (X, Y) clicks specified x, y co-ordinates.
3. Double Click: - double\_click\_mouse (X, Y)
4. The hot keys are handled by keyboard.press(“hot keys”).
5. Hit specific key: - keyboard.press(“key”)

Keys include all the keyboard events for example page down/up, enter, space etc.

1. Enter keys: - type (String)

Type function works on keyboard.press\_and\_release(“char”)

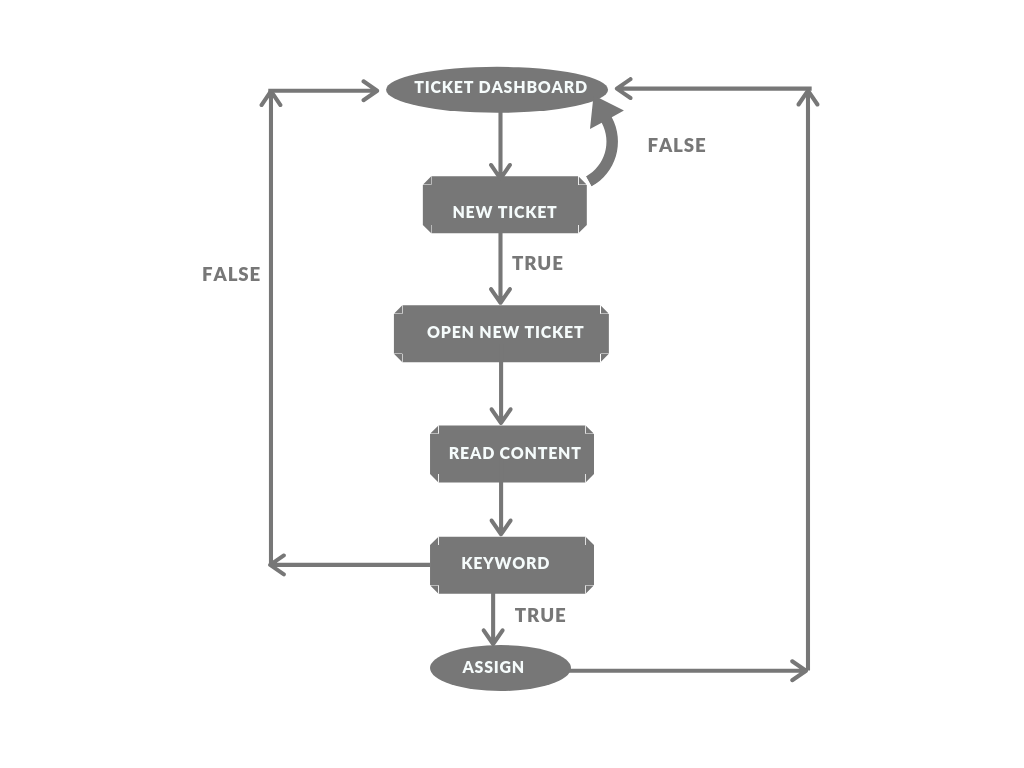
They are segregated based on the ASC11 values.



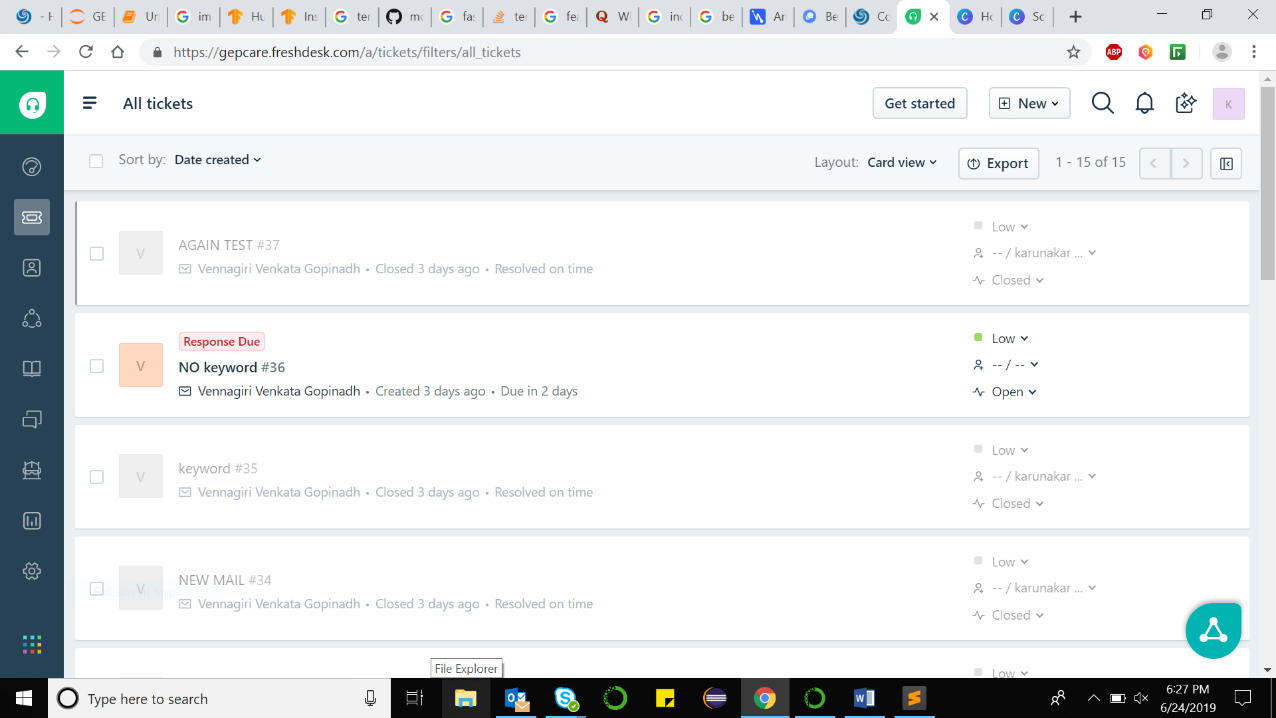
Hit the keys along with “Shift” Key Hit the single key.

# Use case -2: - Lets proceed through our second use case. Use Case – In the fresh desk dashboard we need to take the actions of the new tickets received. Say we are having a series of tickets in the dashboard. And each time there’s a new ticket, we need to open the ticket, read the content and if necessary assign it to some group or person.

**Layer 1 Architecture**

****

Here we are assuming that the Freshdesk portal is already logged in and we are currently on tickets dashboard.



The above is the tickets dashboard of the Freshdesk. Now from here we need to keep waiting for new ticket to come and once we get the ticket take the necessary action defined in flow.

**Start**  Returns the coordinates co-ordinates of mail box

of the new ticket

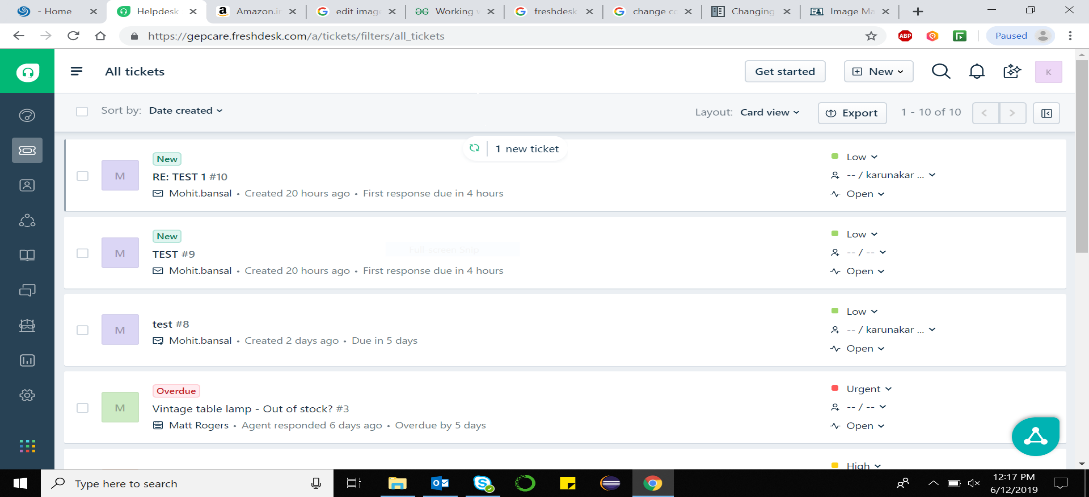
Go to properties scroll down and assign

Returns the new-ticket co-ordinates of text in mail

prompt co-ordinates ticket link

Let’s explain the above process in steps.

1. As it we keep waiting for the new ticket to come. That means we feed continuous screen shots to our OD-model. And once the ticket is identified we go click on it for the new ticket.

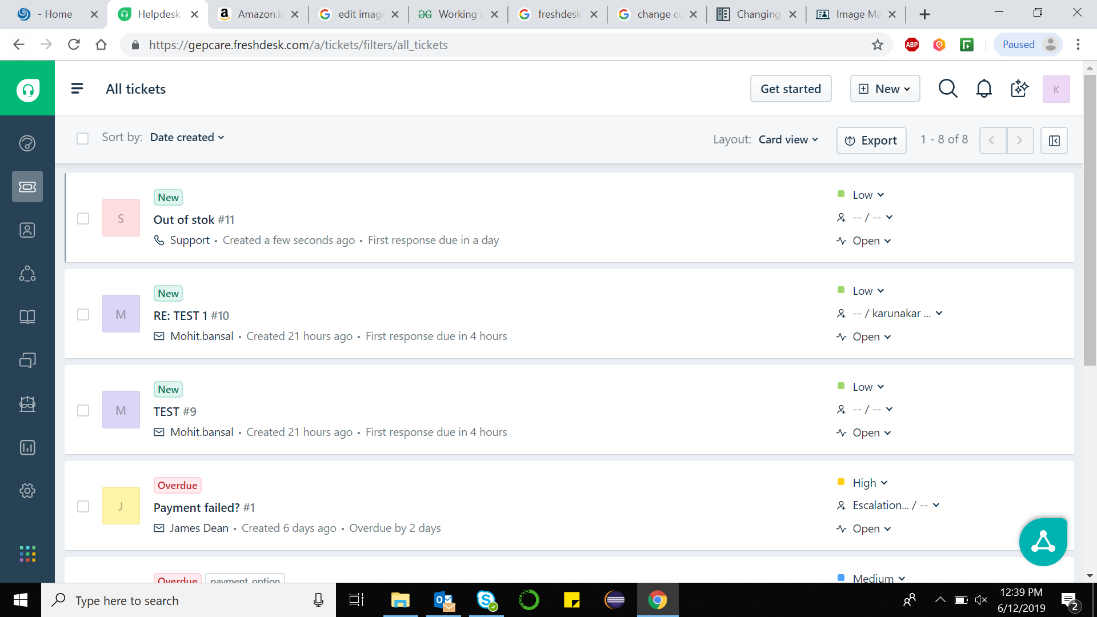


Identified New Ticket

1. The above step ensures the new ticket to be on our dashboard now. Now we need to proceed by opening the new ticket.

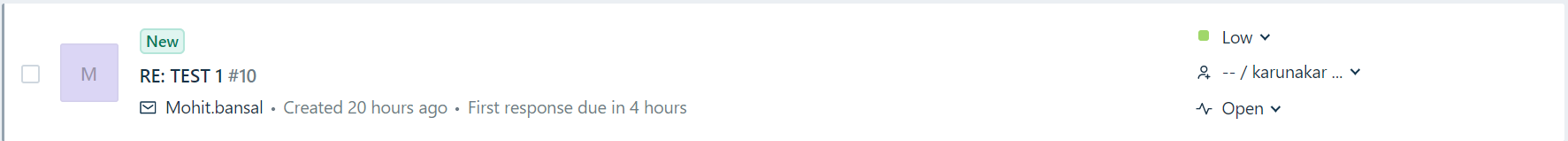
This make us look for the link to open up the new ticket.

Identifying the new tickets and links of mail are two exclusive processes of brain. Here we need open the link associated with the new ticket to open the ticket. So, we need to first identify the new ticket, crop the image to new tickets bounding box and feed it to our OD-model to detect the link in it. It a simple logic for us to use migration of co-ordinates as the OD-model for detecting the link gives the co-ordinates with respect to the cropped image.

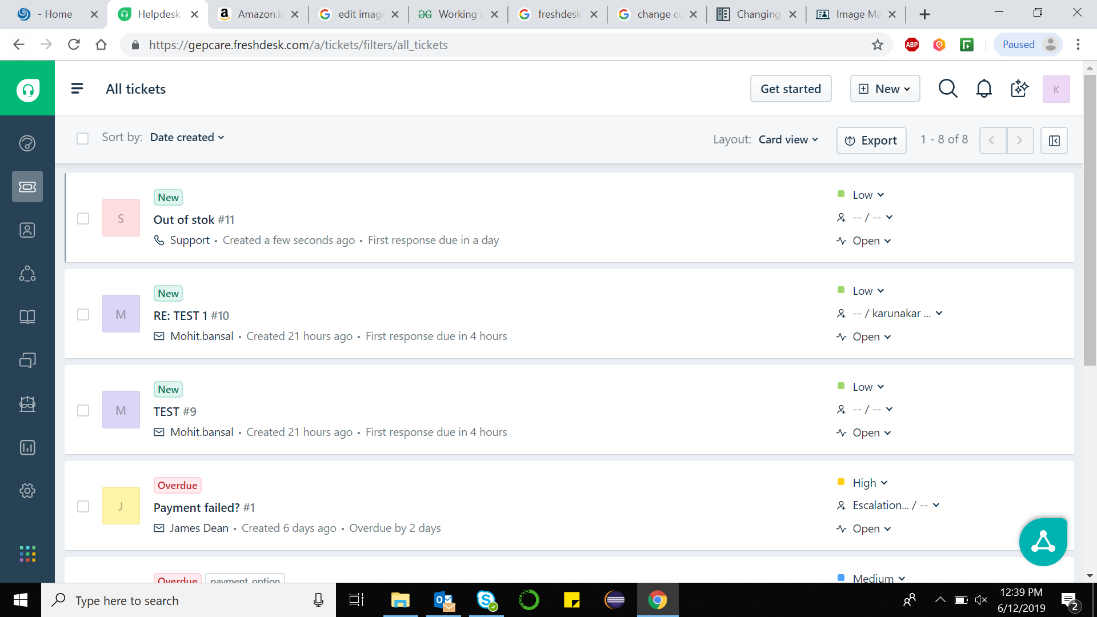


Identified New Ticket is cropped and fed to Od-model

Link



**Co-ordinate Shifting**

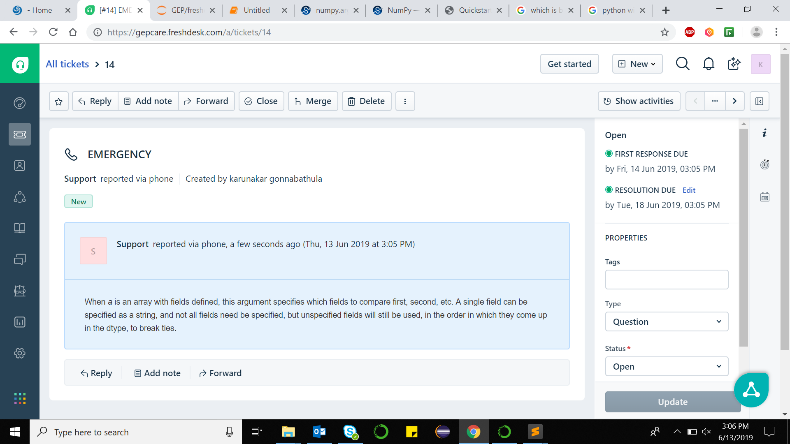


Link co-ordinates are returned w.r.t x2, y2.

**2**

**2**

1. Now proceeding to extract the content in ticket detect the content box, crop it and feed it to the tesseract model designed.



**Text**

**Tesseract**

Tesseract model takes the image as an input and pulls out text content in it. We will be discussing about tesseract in later sections.

1. Now the extracted text is tokenized using NLTK and fed to trained NLP model to classify the content. This leads to the action performed accordingly.

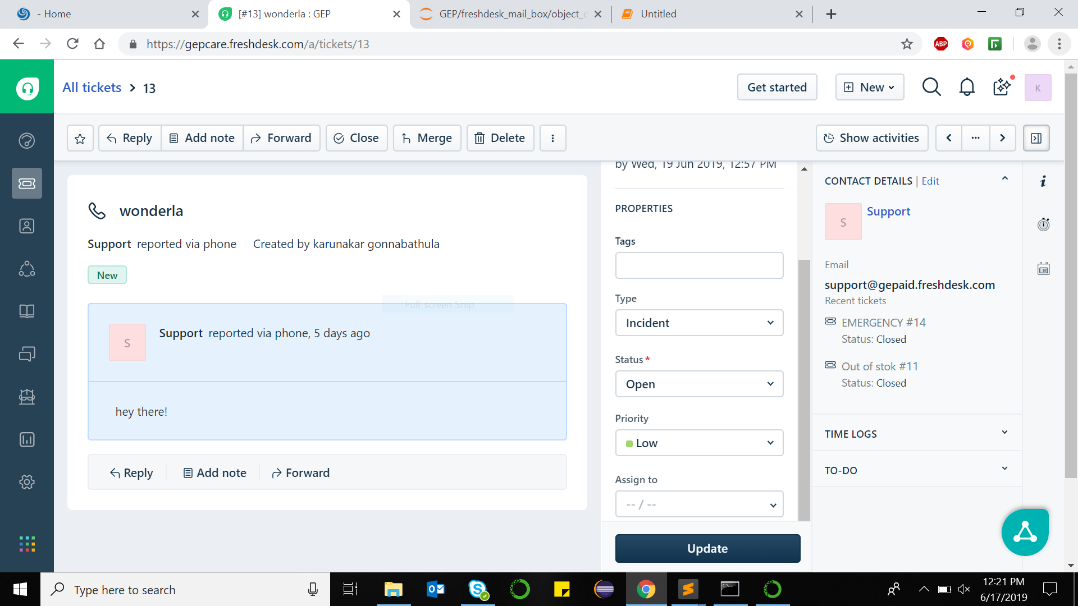
**NLP**

**Tokenized Text**

**Classified**

**Group**

1. Now if there’s a keyword and the assigning action need to be taken, we proceed finding properties and assign to person specified.



1. Assign the inputs in properties and go back to tickets for listening the new ticket again.

# Tesseract

Tesseract has **Unicode (UTF-8) support** and can **recognize more than 100 languages** "out of the box".

Tesseract supports **various output formats**: plain text, OCR (HTML), PDF, invisible-text-only PDF, TSV. The master branch also has experimental support for ALTO (XML) output.

You should note that in many cases, in order to get better OCR results, you'll need to **improve the quality of the image** you are giving Tesseract.

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images.

Python-tesseract is a wrapper for Google’s Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Python Imaging Library, including jpeg, png, gif, bmp, tiff, and others, whereas tesseract-ocr by default only supports tiff and bmp. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

Know further:- <https://github.com/tesseract-ocr/tesseract>

# NLP

Natural language toolkit (NLTK) is the most popular library for natural language processing (NLP). It was written in Python and has a big community behind it.

We proceed using NLTK for our NLP consisting of

* Text Analytics and NLP
* Compare Text Analytics, NLP and Text Mining
  + Text Analysis Operations using NLTK
  + Tokenization
  + Stop words
  + Lexicon Normalization such as Stemming and Lemmatization
  + POS Tagging
* Sentiment Analysis
* Text Classification
* Performing Sentiment Analysis using Text Classification

## **Text Analytics and NLP**

Text communication is one of the most popular forms of day to day conversion. We chat, message, tweet, share status, email, write blogs, share opinion and feedback in our daily routine. All of these activities are generating text in a significant amount, which is unstructured in nature. I this area of the online marketplace and social media, It is essential to analyze vast quantities of data, to understand peoples opinion.

NLP enables the computer to interact with humans in a natural manner. It helps the computer to understand the human language and derive meaning from it. NLP is applicable in several problematic from speech recognition, language translation, classifying documents to information extraction. Analyzing movie review is one of the classic examples to demonstrate a simple NLP Bag-of-words model, on movie reviews.

### **Tokenization**

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

### **Stop words**

Stop words considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.

In NLTK for removing stop words, you need to create a list of stop words and filter out your list of tokens from these words.

#### **Stemming**

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "connect".

## **Text Classification**

Text classification is one of the important tasks of text mining. It is a supervised approach. Identifying category or class of given text such as a blog, book, web page, news articles, and tweets. It has various application in today's computer world such as spam detection, task categorization in CRM services, categorizing products on E-retailer websites, classifying the content of websites for a search engine, sentiments of customer feedback, etc. In the next section, you will learn how you can do text classification in python.

