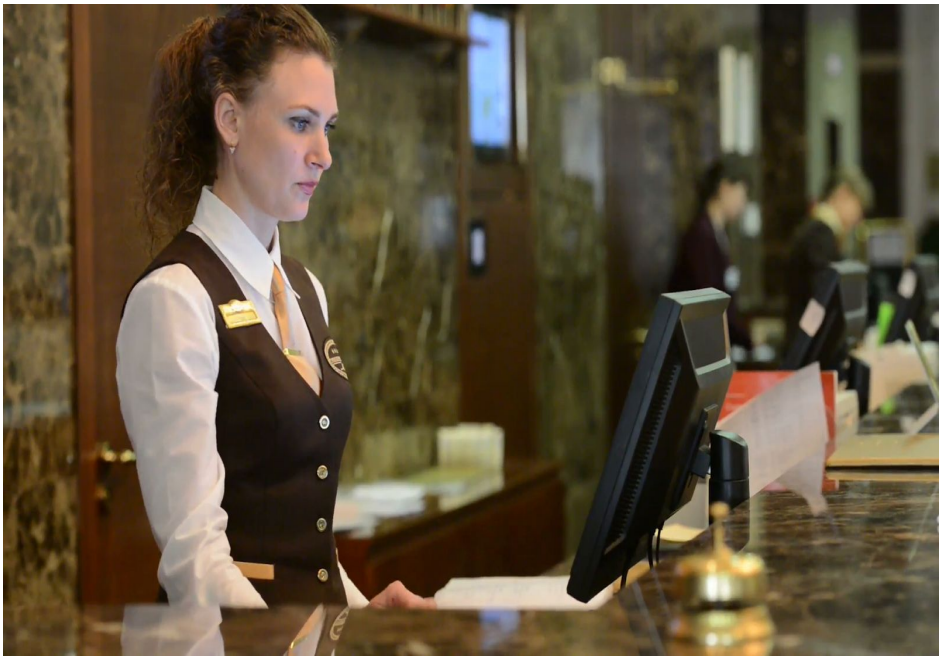
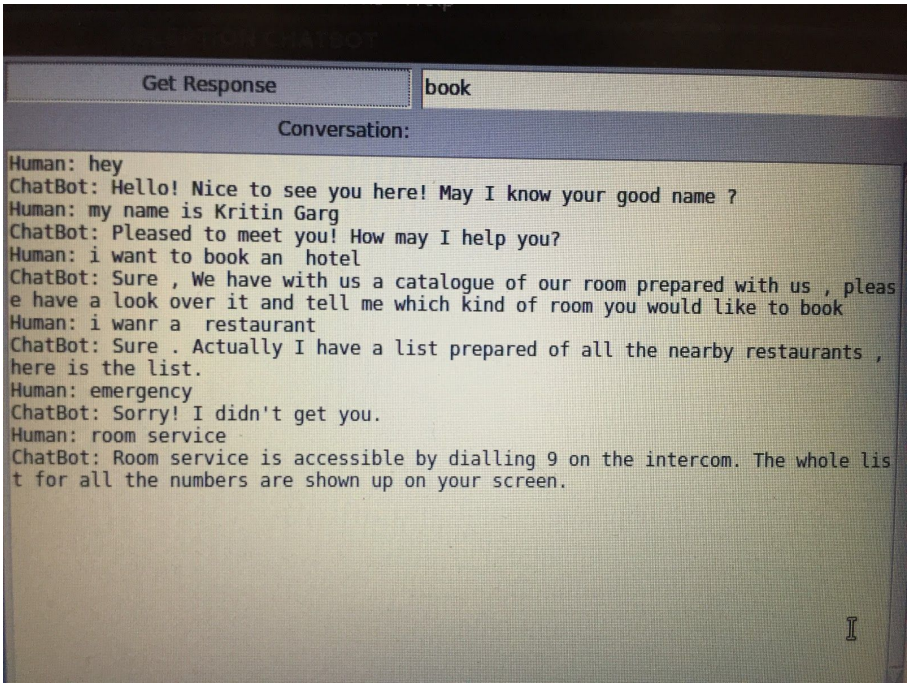
REBECCA

Brings you

FROM THIS



TO THIS



Team members:

Name	Roll Number	Email ID
Kushagra Juneja	170050041	*kuku12320@gmail.com*
Pushpendra Jakhar	170050049	*pjalpha17@gmail.com*
Kritin Garg	170050028	*kritingarg0209@gmail.com*
Ojas Thakur	170070017	*ojassanjivthakur@gmail.com*

What is Rebecca:

It is a Python-based chatbot designed to serve the purpose of a receptionist in a hotel . As machines are taking over human labour in almost every fields , this is one of the left out fields .So our team decided to come up with Rebecca, specially designed for this purpose .
We have used OCR for aadhar verification, and Eigenfaces for the face-verification of the same person so that the hotel security is not at stake. Chatbot replies are being generated using regex as well as using a Seq2Seq trained model. We have also implemented gender recognition using a CNN so as to make it more attractive to customers by using their gender specific salutation. We have a Tkinter based GUI so that the users don't have to chat on the Terminal.

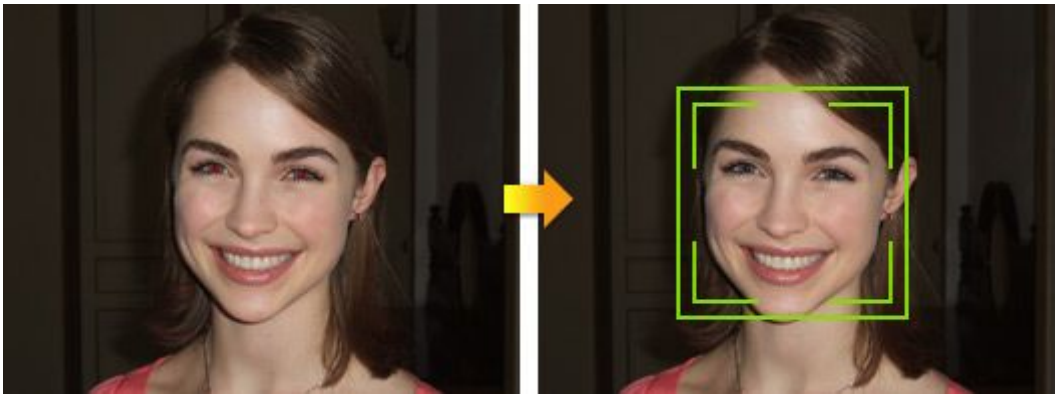
Face detection:

Haar Classifiers: There are a large number of haar features which are different shaped set of pixels. These are supposed to be patterns of alternate white and black bands. These can be used to detect edges in the faces like nose, eyebrows, lips, eye sockets because they have alternate patterns of light and dark areas. These haar features are collected and adaboost algorithm extracts all the useful ones from the large multitude of them. This algorithm simply works as noting the accuracy of each feature applied to each part of the face and keeps the most accurate ones. After this cascading of the features is done do that

parts which do not cross the first ones are classified as negative for a face. This saves a lot of time rather than applying all the features to all parts of the image. We have used the built-in Haar Cascade classifier from OpenCV for our purpose.

Cropping: For better accuracy in the face recognition process, it is important that only the main face area be present and not the surroundings as it wastes picture space. Hence, the detected face area is cropped.

Resizing: All the images need to be of the same size to perform the next face recognition step. Hence we use openCV to resize images to a predetermined size.



Face recognition:

Eigenfaces: This method is based on decomposing all faces into a linear combination of a set of faces. In this, each image is broken down row by row to make a single vector whose entries contain all the information from the image.Then a method called **PCA** is applied on these vectors to find a smaller set of vectors of the same dimensionality. These are called eigenfaces. Unfolding these vectors will give back images, called eigenfaces. Now, all the faces in the dataset can be written as mean face plus a linear combination of all the eigenfaces. These weights are stored and when a new test image is encountered, it is broken down into mean face and linear combination of eigenfaces. Then the distance from each training face is calculated meaning the vectors comprising of the weights are compared with that of the test case. Whichever matches the most, that is the correct face.

We have used the Eigenfaces algorithm (involving PCA) for face-recognition. We obtained 10 images each of the four of us taken from different angles and then applied our face-detection algorithm to each of the images and obtained 10 cropped-face images of each of us. Then we apply the Eigenfaces algorithm and make eigenfaces for each of us. We use 7 images for making eigenfaces and 3 images for testing and obtained an accuracy of around 85%. Hence, now we use all 10 images for making the Eigenfaces and given a new image of the person it tells whether it is one of these persons or not.
To click the image of the person we use the computer's webcam. Then we apply face-detection on it and scale the detected face to the proper dimension. Now, we check if the image matches the pre-defined faces or not. We expect the person booking online to give us their 10 images(which we can automate from their computer's webcam once they give us the permission to use the camera) so that we can verify when they reach at the reception that they are the same person who booked the room.

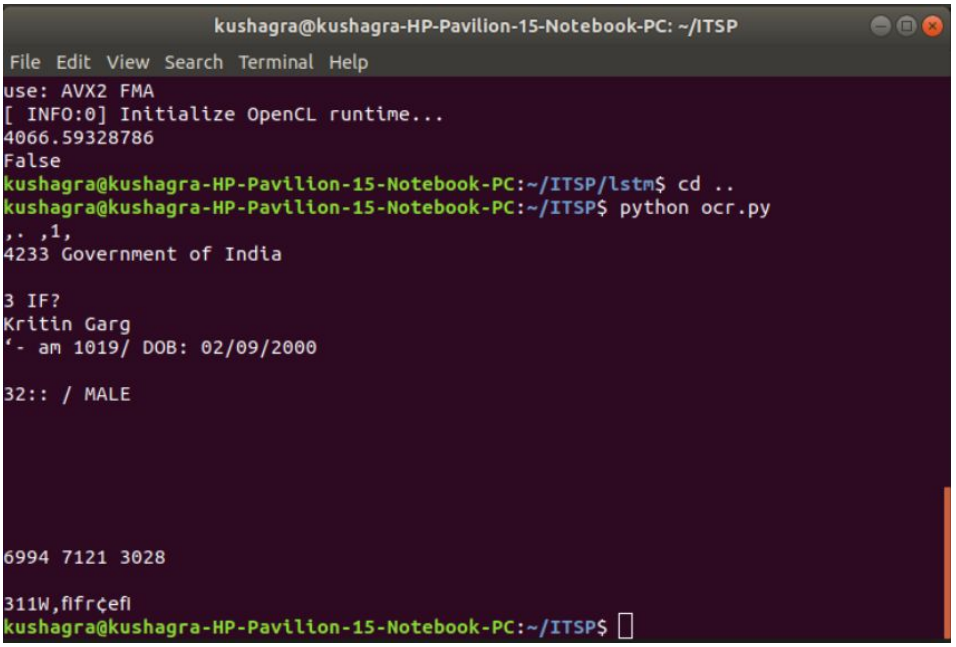
OCR(Aadhar name verification):

OCR: This uses pytesseract library of python to extract text from the image. The idea is to read from the Aadhar card and then verify the persons name from it.

Name matching algorithm: This algorithm works on the idea that the name of the customer will be predetermined from the online booking. Then, since the output is in the form of alphanumeric characters, we can match the string literals and using regular expressions, we can detect the pre known name in the output string.

Mobile camera application: We stream the video from the RTSP app in our mobile into our laptop and take its screenshot when the person has been asked to show the ID Card so that we can detect the name of the person.

This is because the mobile camera has a better quality than the webcam which is very important since the webcam quality was insufficient for the OCR to work.



Chatbot:

Regular Expressions: These are patterns of strings which can be searched for in strings. We can search for keywords inside the input statements and corresponding to these inputs, we can then take actions, by maintaining dictionaries inside. Everytime an intent matches, we can generate replies accordingly and take action.

NER using SpaCy : This is named entity recognition. We can directly extract entities(important data) from the text to know the names etc. of the user. SpaCy also gives the type of the entity which it recognises so that we can pick up names an important dates from it.

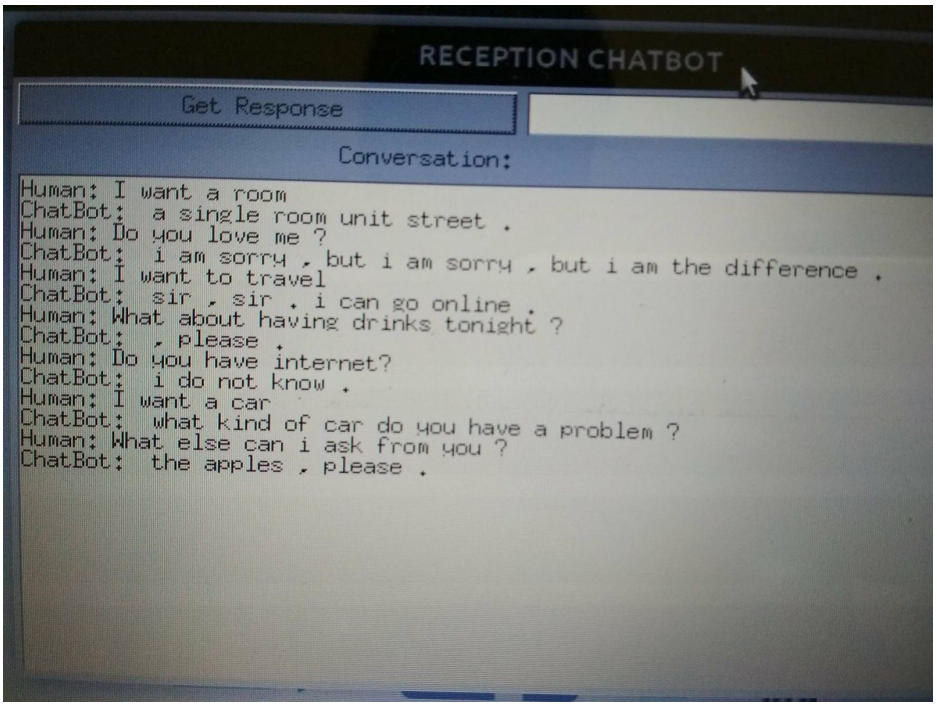
SQL database:

We are using a SQLite database and SQL methods to access the files inside it. This database will store the availability status of all the rooms and also the names of people occupying them. These can be changed and also rooms can be assigned. We have created some catalogues such as room type , restaurants , emergency contacts etc . which will be displayed as per the requirements of the customer .We have considered both the options of online booking and offline booking for any customer and we update the database accordingly. We have also taken care of SQL Injection to prevent the users from entering malicious inputs and harming our system.

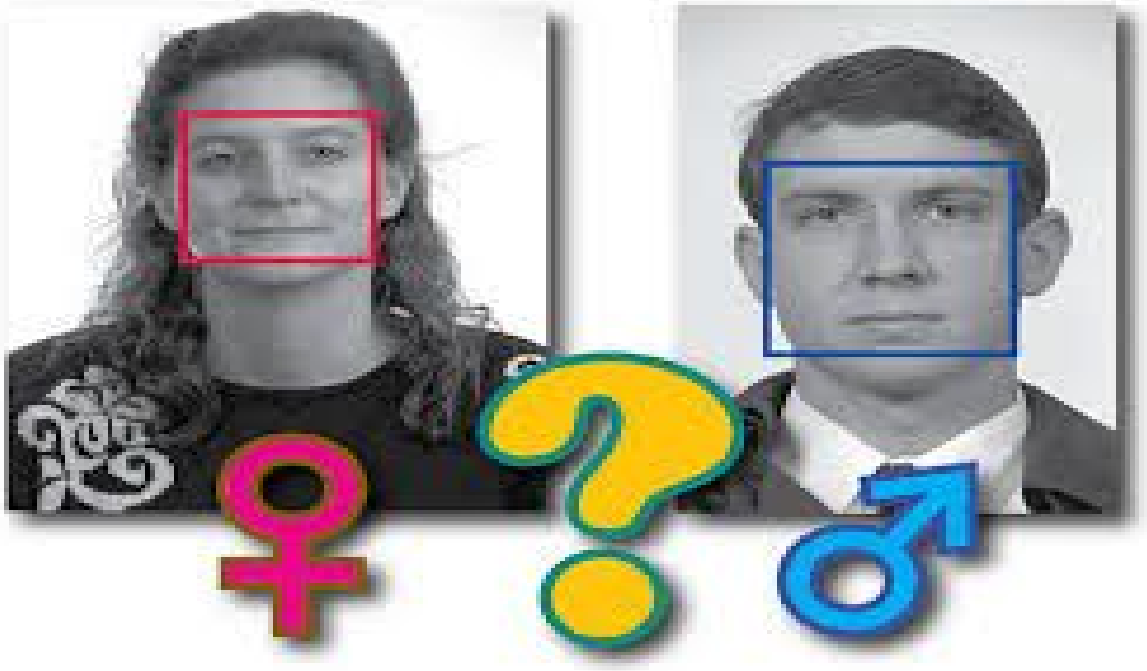
Seq2Seq Modeling:

In the case when the question asked by the user doesn't match any of the pre-defined regular expressions, the input is transferred to the Seq2Seq LSTM-based model to reply. We have used a pre-trained model for this purpose. We could not find a direct compiled dataset of hotel reception chats to fine-tune our model hence we created one. We compiled the hotel-reception chats from over 20 online resources which mainly were example conversations given in English learning classes majorly from https://www.eslfast.com/robot/topics/hotel/hotel.htm and were able to obtain approximately 600 dialogue pairs. Some of the

example replies of the chatbot are given below. All these replies are generated by the Seq2Seq model only and none of them are hardcoded in any sense.



Gender recognition:



We have made a gender classifier so that the chatbot knows the salutation for the particular customer is “Sir” or “Mam”. The chabot clicks an image of the person from the computer’s webcam and then detects face using our face detection code. It then scales the face to the required size of 224*224*3. We have trained a VGG-16 classifier on approx. 10,000 labelled face-images obtained from the UTKFace dataset obtained from https://susanqq.github.io/UTKFace/ We have used pre-trained standard weights of VGG16 and then fine-tuned them on our dataset. We were able to achieve an accuracy of 95 % on training set and 89% on test set after training it for 20 epochs. Now, we just give the clicked image to the model and hence predict the gender of the customer currently talking to the chatbot.

Tkinter interface:

We have created the interface of the chatbot by using Tkinter library . We have used the interface created for Chatterbot by IBM and modified it as per our requirements . The main screen will always be shown for the person to come and chat with it .

Improvements:

1) The interface is quite basic and has a lot of scope of improvement .
2) The face recognition does not work quite well because it uses only 7 images during the training keeping in mind the trouble of the customers if they have to click more images.
3) The seq2seq model of the chatbot doesn’t work that well partly because we could not find enough data for Hotel Reception chats and partly due to our unavailability of good GPUs due to which we were not able to use pre-trained models trained on a very large datasets of English dialogs since they would often have large vocabulary and hence we wouldn’t even be able to fine-tune them with our laptops under reasonable amount of time.

Instructions for using chatbot:

1. First greet the bot with hey, hello, etc.
2. Whenever a window for clicking image appears, press the Enter/Return key to proceed. This will open the chatbot console. Do not click on the (X) button in the image.
3. Write your name as it is mentioned on the Aadhar card.