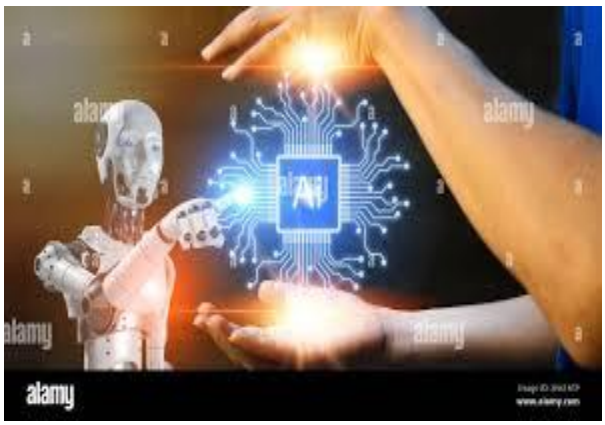
CREATE A CHATBOT USING PYTHON

TEAM MEMBER

510521104023: LAVANYA K

PHASE II: INNOVATION OF CHATBOT



CHATBOT INTRODUCTION:

A chatbot is essentially a computer program written to copy a human and have a conversation with a human.

A chatbot is not only used for purpose of conversation and it has been used many field which means entertainment, finance, medical field, legal and customer services.

While consumer chatbots like Siri and Alexa are gaining mass popularity, chatbots are also increasingly used in organizations like finance and banking institutions to handle repetitive customer queries.

Business chatbots are usually designed to answer a very narrow field of questions, not unlike consumer voice assistants.

For example, Domino's has a chatbot that allows you to order pizza through text. The bot mimics the conversation that you would have had with a human phone operator except, unlike a human operator, the bot would probably not be able to answer a non-pizza related question.

As an AI language model, I do not have the capability to provide opinions on bugs or programming errors. However, some common issues or bugs found in chatbots are:

- Delayed response time
- Inability to understand user queries or respond appropriately
- Lack of personalization or customization
- Over-reliance on scripted responses
- Inconsistent user experience
- Inability to handle complex queries or contexts
- Integration issues with other systems or applications
- Troubleshooting errors and maintaining codebase
- Difficulty in handling multiple languages

Developers need to continuously improve the chatbot's performance and make sure it caters to user needs to provide a seamless experience.

MY QUESTION:

And also I found that in chatbot which means if I want to text some questions mean I can do that but I have images to ask or search some information about that images. But I don't have any chatbot scanner to scan my images and give the information .

SOLUTION TO CREATE SCANNER IN CHATBOT:

So,I was dealing with an API that extract text from images. and I noticed something important API only have good results when i send it scanned image. This simply means if I take a photo from my mobile camera and send it directly to the API i won't get results.

The API was simple and its accuracy was fine giving scanned photo so I decided to look into how to scan image in python. After some digging and watching many tutorials I realized that each tutorial have a part of the puzzle and I should bring them together in order to got what i want.

I 'll try to make this as much simple and clear as I could I hope you enjoy the journey.

First let's install necessary packages in our PyCharm project. I preferred PyCharm as it creates virtual environment let you do whatever you want inside it without affecting the main root.

Installing Packages :

- opencv-contrib-python I can use opencv-python package but I prefer contrib package to get access to extra modules developers have created.
- scikit-image to handle some image processing stuff.
- numpy to handle matrices
- imutils to make image processing functions easier.

all these package I can install it using pip already defined in venv of PyCharms projects that’s it now we have our environment clear and ready to go.

Edge Detection :

```
from skimage.filters import threshold_local
import numpy as np
import cv2
import imutils

image = cv2.imread('your photo name')
ratio = image.shape[0] / 500.0
orig = image.copy()
image = imutils.resize(image, height = 500)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray = cv2.bilateralFilter(gray, 11, 17, 17)  # 11  //TODO 11 FRO OFFLINE MAY NEED TO TUNE TO 5 FOR ONLINE
gray = cv2.medianBlur(gray, 5)
```

I will explain how the code work briefly to get sense of what is happening

first I am reading our image then, I take ratio of how big our image is compared to height 500 pixel. After that I resize our image using imutils library, this step aims to speed up our process and make our edge detection more accurate. and here comes the interesting part.

I convert our image from colored to gray scale

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

then I apply bilateral Filter I should talk about that a little bit.

Bilateral filter is basically used for smoothing images, reduce the noise while preserving edges. Normal blurring usually get rid of noise whether it was noise or actual edges to overcome this problem I use non-linear bilateral filter.

```
gray = cv2.bilateralFilter(gray, 11, 17, 17)
```

Now let's smooth the edges a little bit with medianBlur

```
gray = cv2.medianBlur(gray, 5)
```

Now, I should be able to detect our edges perfectly fine using canny algorithm I can read more about it here. It is quite interesting.

```
edged = cv2.Canny(gray, 30, 400)
```

Finding contours :

Before looking into the code I need to imagine the idea now I have each edge of the image. and I have a very important advantage, as I are scanning a piece of paper which usually will take the shape of rectangle So what I know until now is that I have a rectangle shape with four points and four edges. usually the document to be scanned would be the largest area in the image which means in other words largest edges have higher probability to be the document we are scanning.

```
# find contours in the edged image, keep only the largest ones, and initialize our screen contour
countours, hierarcy = cv2.findContours(edged, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)
imageCopy = image.copy()
```

```
# approximate the contour

cnts = sorted(countours, key=cv2.contourArea, reverse=True)

screenCntList = []
scrWidths = []
for cnt in cnts:
    peri = cv2.arcLength(cnt, True)
    # you want square but you got bad one so you need to
approximate
    approx = cv2.approxPolyDP(cnt, 0.02 * peri, True)

    screenCnt = approx
    if len(screenCnt) == 4:
        (X, Y, W, H) = cv2.boundingRect(cnt)
        screenCntList.append(screenCnt)
        scrWidths.append(W)
```

in open-cv we detect the object black on background white so it is important to apply threshold or canny edge detection before coming this far.

See, there are three arguments in **cv.findContours()** function, first is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the contours and hierarchy. Contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object. “as open-cv documentation said”

in line 6 I sort the contours in reverse way so that I have the largest contours which we are interested in first.

then I loop through contours and approximate the contours then specify that it is a closed shape passing parameter True to cv2.arcLength

if I have four points I simply determine the width and height and the top left two points of our rectangle as in line 17 using cv2.boundingRect

then I save all rectangles and widths I find to apply findLargestCountours and find the largest one as I said the largest rectangle should be our document.

Check:

```
if not len(screenCntList) >= 2:  # there is no rectangle found
    print("No rectangle found")
elif scrWidths[0] != scrWidths[1]:  # mismatch in rect
    print("Mismatch in rectangle")
if not len(screenCntList) >= 2:  # there is no rectangle found
    print("No rectangle found")
elif scrWidths[0] != scrWidths[1]:  # mismatch in rect
    print("Mismatch in rectangle")
```

if length of points is less than 2 this means I failed to find rectangle so print that. Also if the widths are not the same this means that our shape is nor rectangle thus print mismatch in rectangle.

Apply Transform and show the results:

```
pts = screenCntList[0].reshape(4, 2)
```

```
# Define our rectangle
rect = order_points(pts)

warped = four_point_transform(orig, screenCntList[0].reshape(4, 2) * ratio)
warped = cv2.cvtColor(warped, cv2.COLOR_BGR2GRAY)
T = threshold_local(warped, 11, offset = 10, method = "gaussian")
warped = (warped > T).astype("uint8") * 255
cv2.imshow("Original", image)
imutils.resize(warped, height = 650)
cv2.imshow("warp", warped )
cv2.waitKey(0)
cv2.imwrite('scanned.jpg', warped)
```

Now I take our largest screen and reshape it to 4*2 means four points each one has x,y. In line 3 I just order our points to shape the rectangle using order_points function

In line 5 I apply four point transformation this is simply gives you a bird look to your document like you are flying and see from a vertical perspective “birds eye view” you can find more about that here.

then I apply gray scale to our image and define threshold to get the black white image scanned as in line 8&9

I see the value of ratio in line 6 is that I can get back the size of the original image. Notice that I working on an image with 500 pixel height only while our original image is 500*ratio.

CONCLUSION:

Due to this solution or application can insert in chatbot and it made easier to user.It has efficient method to collect the information from the chatbot and also if we have give text means it consumes more time.

As you can see, innovative chatbots can meet many customer needs. What is more, this solution works in a lot of different industries. However, all of these chatbots have one thing in common – they help and engage. All in all, these definitely are some of the more innovative chatbot usecases and the ones we're likely to see more of in the coming years.