Tutorial 7
Analysis of Assignment 2

Kai CHEN

Overview

Tasks

- build an image classifier
- deploy the classifier as a chatbot on Telegram

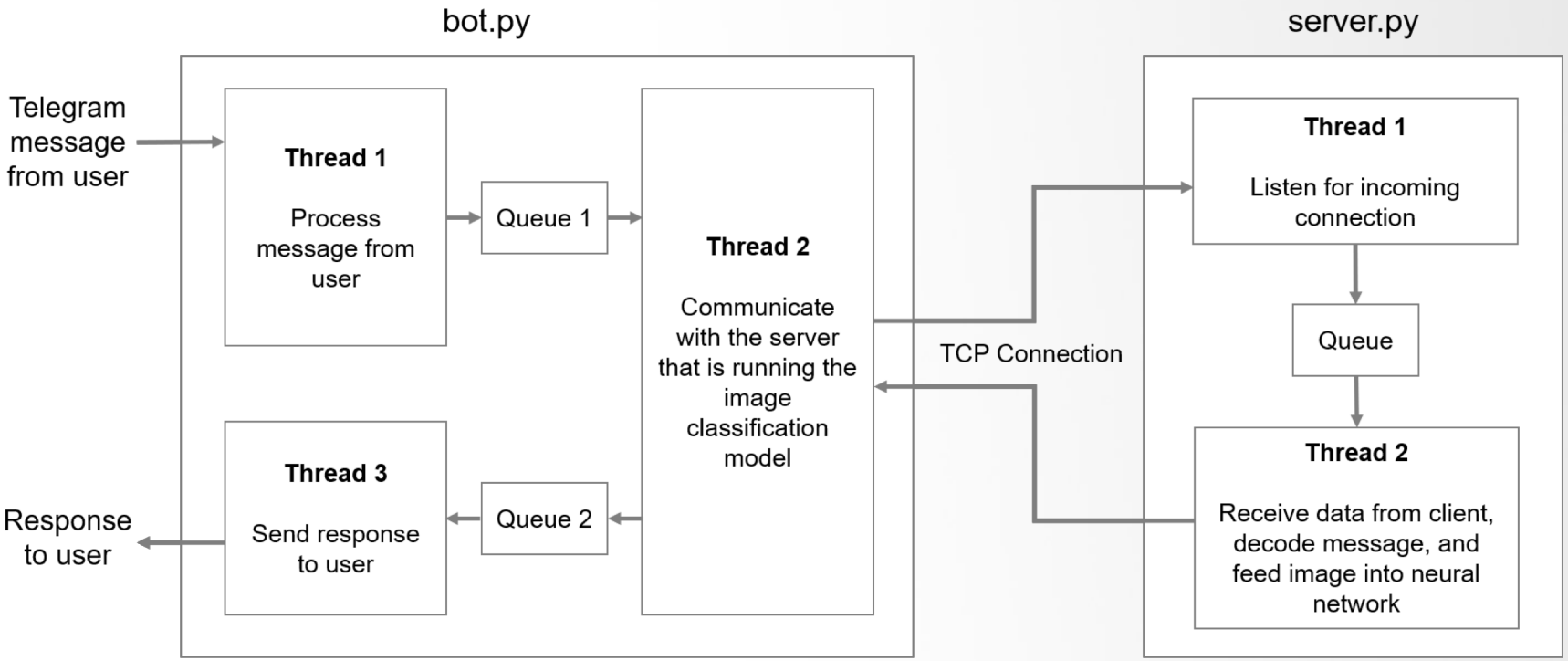
Overview

Difficulties

- socket communication
- multi-threading

Overview

Framework



Modules

Multi-threading (bot.py)

Thread 1

Thread 2

Thread 3

```
queue1 = Queue()
queue2 = Queue()

def handle(msg):
    chat_id = handle_message()  # get chat information
    image_file = download_image(msg, chat_id)  # download image
    image = Image.open(image_file)  # open the image
    queue1.put((image, chat_id))  # put image data and chat_id to queue

def client_thread():
    while True:
        image, chat_id = queue1.get()
        soc = create_socket()  # create a socket and connect to the server
        send_image(soc, image)  # send image data

        data = receive_results()  # wait for response from the server
        queue2.put((data, chat_id))  # put the response to queue

def response_thread():
    while True:
        data, chat_id = queue2.get()
        send_message(data, chat_id)
```

Modules

Multi-threading (server.py)

```
queue = Queue()

def listen_thread():
    server_socket = create_socket()
    while True:
        client_socket, address = server_socket.accept()  # accept connections
        queue.put((client_socket, address))  # put the client socket and address into the queue

def model_thread():
    model = init_model()
    while True:
        client_socket, client_address = queue.get()  # get client socket and address
        image = receive_image(client_socket)  # receive image data from client
        output = predict(model, image)  # model prediction
        reply = format_output(output)  # format output
        send_response(client_socket, reply)  # send response
```

Thread 1

Thread 2

Modules

Send/Recv image data

```python
encoded_image = base64.b64encode(img_data)
soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
soc.settimeout(5)
soc.connect(("localhost", 5000))
data = {
    'image': encoded_image.decode("utf-8"),
    'chat_id': chat_id
}
data = json.dumps(data) + "##END##"
soc.sendall(data.encode("utf-8"))
```

send

```python
data = ""
while True:
    part = client_socket.recv(1024)
    data = data + part.decode("utf-8")
    if "##END##" in data:
        break
data = data.replace("##END##", "")
data = json.loads(data)
image_data = base64.b64decode(data['image'])
```

recv

Modules

Download images

```
# Use the chat_id to name the image
image_file = "{}.png".format(chat_id)

if content_type == "text":
    # If content type is text, it should contain a URL to an image
    content = msg["text"]
    if not content.startswith("http"):
        return
    # Download the image and save to a local file
    image_data = requests.get(content).content
    with open(image_file, 'wb') as outfile:
        outfile.write(image_data)
elif content_type == "photo":
    # If content type is photo, download photo from Telegram
    bot.download_file(msg['photo'][-1]['file_id'], image_file)
else:
    return
```

Modules

Classifier

```
model = ResNet50(weights='imagenet')
# Get a reference of the tensorflow graph
# This is necessary when using the model to do inference later
graph = tf.get_default_graph()
```

```
with open('image.png', 'wb') as outfile:
    outfile.write(image_data)

# Load the image, preprocess it
img = image.load_img('image.png', target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

# Get predictions from the model
with graph.as_default():
    preds = model.predict(x)
decoded = decode_predictions(preds, top=5)[0]
```

Thank you!