Sistemas Distribuídos

(Trabalho Final)

Johnny Marcos Silva Soares, 385161
Letícia Saraiva Chaves, 402120
5 de junho de 2019

Análise facial em imagens

Análise facial é o processo de detecção de uma face dentro de uma imagem e de extração dos atributos relevantes dela. O Amazon Rekognition retorna os atributos faciais apresentados na imagem, como por exemplo, sexo, sorriso, emoções, olhos abertos, boca aberta, bigode, barba para cada face detectada, junto com uma caixa delimitadora e uma pontuação de confiança para cada atributo.

A aplicação foi desenvolvida nas linguagens de programação *Python* e Java e consiste em um cliente (Python) que é executado na máquina do usuário, um proxy (Java) e um servidor (Python) que são executados numa máquina virtual da Amazon Elastic Compute Cloud (Amazon EC2). O cliente se conecta ao proxy, que deve estar conectado ao servidor, captura uma imagem e solicita a análise da foto. A imagem é mandada para o Amazon S3 e o servidor vai informar qual imagem o Amazon Rekognition vai buscar no S3 para fazer a análise.

O usuário pode capturar quantas fotos desejar e decidir se quer analisá-las ou não. Após a análise de uma foto, entre os resultados retornados, o usuário pode escolher ver um relatório mais completo do procedimento realizado na foto e, também, ver alguns gráficos gerados a partir de todas as análise que ele já realizou na conta. Quando o usuário quiser trocar de conta ou sair da aplicação, é só ir no botão de '*Logout'*.

Arquitetura da aplicação

A imagem abaixo (Figura 1) ilustra a arquitetura da aplicação. Um dispositivo que contém a aplicação está conectado a Internet e tenta conexão com o proxy, que está conectado ao servidor, e ambos estão localizados em uma máquina virtual da Amazon EC2. Em seguida, a aplicação é executada no dispositivo e uma foto é capturada. Quando o usuário solicita uma análise da foto, é feito um *upload* da imagem para o *bucket* do Amazon Simple Storage Service (Amazon S3), então o cliente envia o nome do arquivo salvo no *bucket* para o proxy, que em seguida, encaminha a mensagem para o servidor que informa ao serviço Rekognition da Amazon para carregar a imagem do Amazon S3.

O serviço Rekognition faz a análise da imagem e retorna os resultados para o servidor que em seguida, salva alguns dados específicos no banco de dados utilizando o serviço Amazon Relational Database Service (Amazon RDS) e envia o resultado para o proxy, que irá direcionar a mensagem para a devida aplicação usuário.

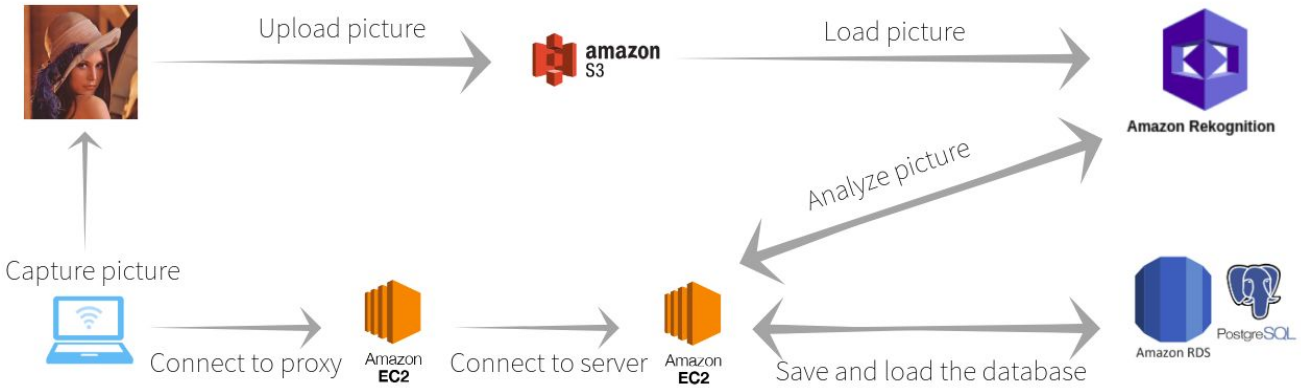

Figura 1: Arquitetura da aplicação

Serviços utilizados na Amazon

- Amazon RDS

  Utilizamos o serviço Amazon RDS e criamos um banco de dados PostgreSQL com duas tabelas, a primeira para armazenar login e senha dos usuários e a outra tabela armazena dados do sorriso e o sentimento predominante de cada rosto analisado.

- Amazon EC2

  No Amazon EC2 criamos uma máquina virtual com IP público que é utilizada como proxy com a porta 8819 aberta, no qual a aplicação cliente se conecta e envia informações necessárias para login e análise da imagem. Também foi definido um servidor rodando na porta 6000, no qual o proxy se conecta e encaminha mensagens dos clientes, dependendo do método definido pelo cliente.

- Amazon S3

  O serviço Amazon S3 foi utilizado como um repositório de imagens, portanto, a aplicação cliente faz *upload* das fotos capturadas para que o Amazon Rekognition recupere a imagem identificada pelo servidor para fazer a análise necessária.

- Amazon Rekognition

  O Amazon Rekognition é feito com tecnologias de *Deep Learning* desenvolvida pelos cientistas de visão computacional da Amazon para analisar diariamente bilhões de imagens e vídeos. Podendo ser aplicado em vários cenários distintos, no nosso caso, foi utilizado o serviço de análise facial, com o intuito de inferir na faixa de idade, sexo, detecção de sorriso entre outras informações.

Aplicação desenvolvida

➢ Padrão das mensagens

Nas Figuras 1 e 2 é possível ver os detalhes da classe Mensagem nas linguagens Python e Java. Os argumentos das mensagens estão no formato de String, devido aos métodos possuírem argumentos diferentes. No *doOperation* as mensagens são feitas, enviadas e o retorno vindo do Proxy é esperado para ser retornado para a aplicação cliente.

```python
def doOperation(objectReference, methodId,arguments):
    if (objectReference == 'Image Analisys'):
        global requestId
        messageType = 0
        msg_send = {"messageType": messageType,"requestId": requestId,"objectReference": objectReference,
                    "methodId": methodId, "arguments": arguments}
        send = json.dumps(msg_send).encode('utf-8')
        print(send)
        tcp.send(send)

        data_receive= tcp.recv(9000)
        msg_read = json.loads(data_receive.decode('utf-8'))

        if(msg_read['requestId'] == requestId):
            requestId = requestId + 1
            return msg_read['arguments']
    return ''
```

Figura 2: Função doOperation com o padrão da mensagem no Python

```java
class Msg{
    int messageType;
    int requestId;
    String objectReference;
    int methodId;
    String arguments;


public Msg(int messageType,int requestId,String objectReference,int methodId,String arguments){
        this.messageType = messageType;
        this.requestId = requestId;
        this.objectReference = objectReference;
        this.methodId = methodId;
        this.arguments = arguments;

    }
```

Figura 3: Classe mensagem na linguagem Java

A aplicação possui 4 métodos principais que serão utilizados para definir a função utilizada, como mostrado na tabela abaixo.

Identificador do Método	Descrição
MethodId 1	Analisa uma imagem enviada e retornar algumas informações.
MethodId 2	Verifica se o usuário existe para realizar Login.
MethodId 3	Verifica se o usuário existe, caso contrário é criado um novo.
MethodId 4	Recupera do banco de dados todas as análises de um usuário.

➢ Tela de Login

A tela de login (Figura 4) é utilizada para obter um controle de acesso da aplicação. Só tem acesso a aplicação quem está logado no sistema. Cada pessoa tem um *username* único e se o usuário ainda não tiver cadastro no banco, ele pode se registrar. No caso de *Login* e *Register* o cliente Python gera a mensagem no formato json e converte para bytes com os identificadores dos métodos 2 e 3 respectivamente. Após isso, o proxy em Java analisa os métodos enviados e faz o controle de acesso, verificando ou inserindo usuários na tabela de login.

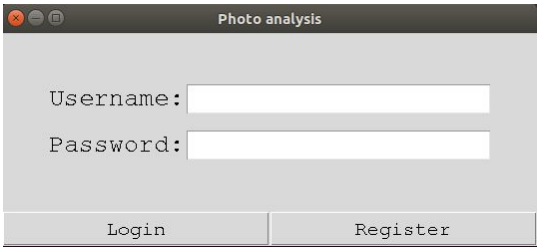

Figura 4: Tela de login

➢ Tela principal

A tela principal (Figura 5) mostra a aplicação assim que o usuário realiza o login. A câmera é ativada e o usuário pode capturar uma foto que tenha somente uma face na imagem, logo em seguida ele pode clicar no botão '*Analyze*' para que a foto capturada seja analisada. Nesse caso, o *methodId* selecionado é 1 e o proxy simplesmente irá deixar a mensagem passar e chegar no servidor, já levando em conta que a mensagem possui um remetente logado. Em seguida, o servidor irá fazer a análise da imagem e retornar para o proxy a mensagem de resposta que será redirecionada para o cliente.

A figura 4 nos mostra a foto já analisada. Algumas informações básicas são retornadas para o usuário, juntamente com opções para gerar relatório e gráficos das análises do usuário e uma caixa delimitando a face detectada na imagem.

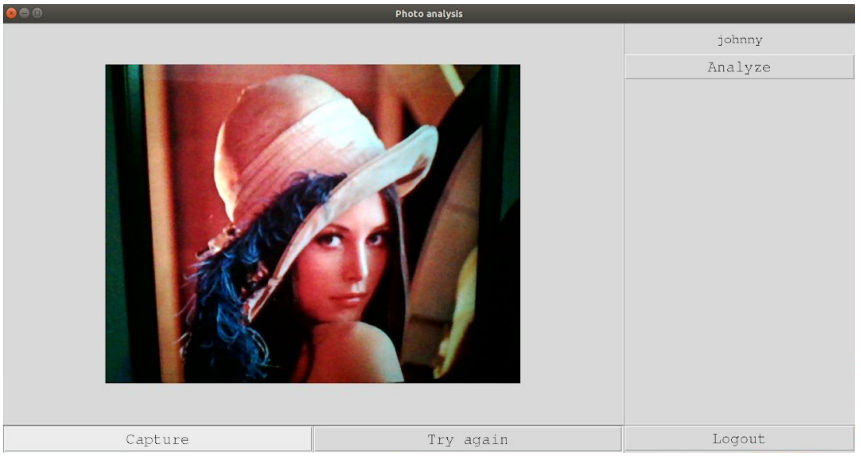

Figura 5: Tela principal

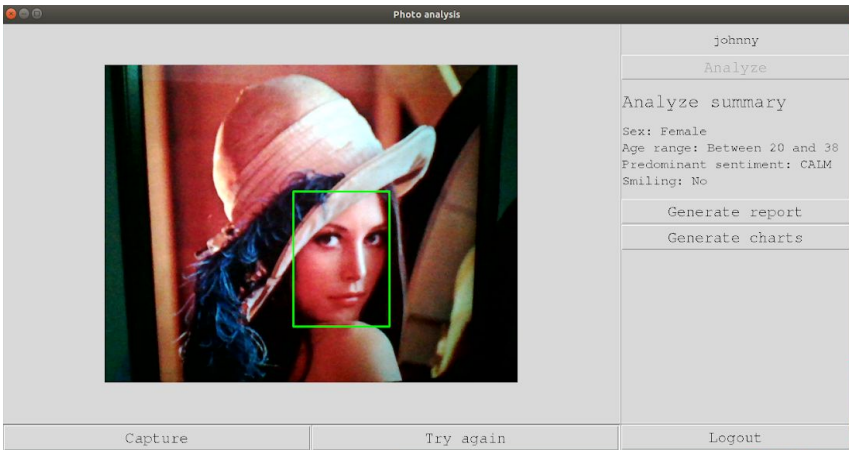

Figura 6: Tela de resultados

➢ Tela de relatório

A tela de relatório (Figura 7) traz informações mais detalhadas a partir da análise da imagem. Além da características identificadas na foto, a confidência, que é como um número de confiança, de cada atributo também é informada.

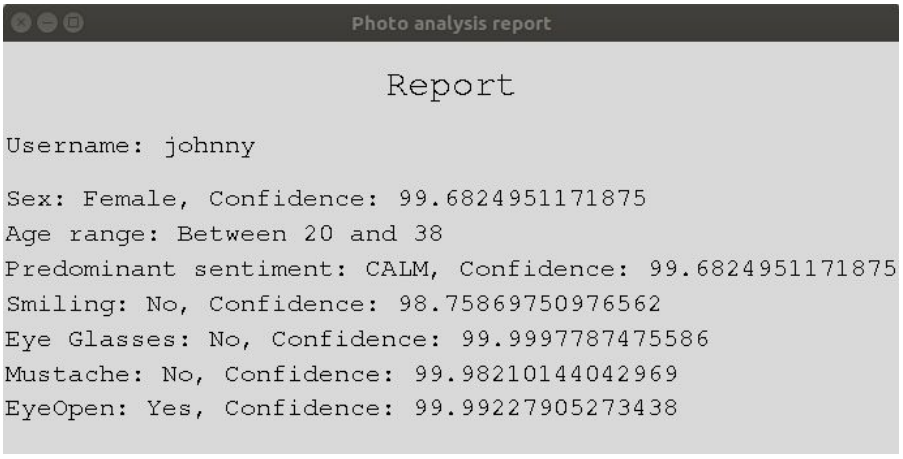

Figura 7: Tela de relatório

➢ Tela de gráficos

A tela de gráficos (Figura 8) traz dois tipos de gráficos, um gráfico de pizza, que informa o percentual de fotos que a pessoa tem sorrindo e não sorrindo, e outro gráfico de colunas que lista quantas fotos a pessoa já tirou de cada sentimento identificado. O último método com identificador 4 é enviado pelo cliente para o proxy, que em seguida, encaminha para o servidor que irá fazer uma busca no banco de dados pelo usuário logado e retornará um lista de tuplas com os detalhes do usuário. Esses dados serão enviados para o proxy e imediatamente enviados para o cliente para as devidas análises.

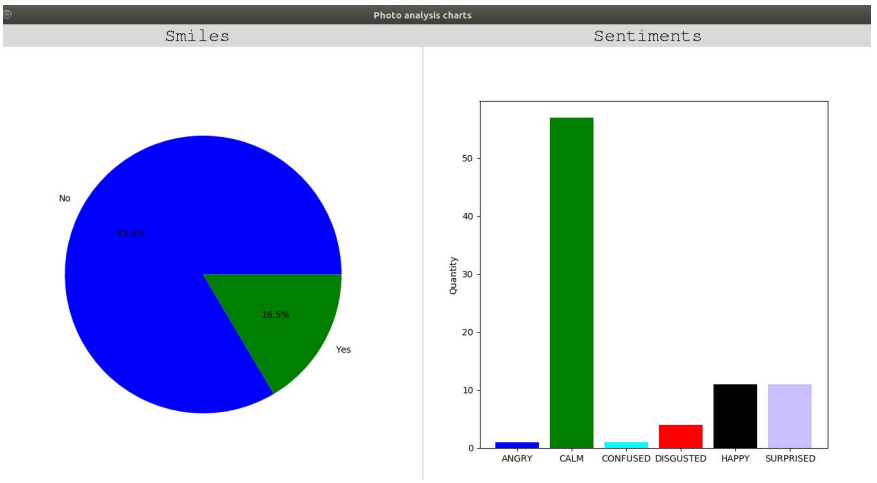

Figura 8: Tela de gráficos

Referências

Amazon Rekognition – Video e Image – AWS ([S.d.]). https://aws.amazon.com/pt/rekognition/

Boto 3 Documentation ([S.d.]). https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

Documentação do Amazon Simple Storage Service ([S.d.]). https://docs.aws.amazon.com/s3/index.html

Documentação do Amazon Elastic Compute Cloud ([S.d.]). https://docs.aws.amazon.com/ec2/index.html

Documentação do Amazon Relational Database Service ([S.d.]). https://docs.aws.amazon.com/rds/index.html

Documentação do Psycopg ([S.d.]). http://initd.org/psycopg/docs/

Documentação OpenCV ([S.d.]). https://docs.opencv.org/

Documentação Tkinter ([S.d.]). https://docs.python.org/3/library/tkinter.html