Projeto Final
COCO Dataset Image Search

Introdução

O projeto consiste em construir um buscador de imagens usando o a parte não classificada do *dataset* COCO (Common Objects in Context). Nessa ferramenta, o usuário pode inserir uma palavra (dentre as "labels" disponíveis) e recebe como retorno um conjunto de imagens que contém o objeto descrito por essa palavra, em ordem de probabilidade de ele estar na imagem.

Movendo o dataset para o Amazon S3

Primeiramente foi necessário passar o dataset para um *bucket* do Amazon S3. Para fazer isso, foi criada uma instância na Amazon EC2 com espaço o suficiente na memória para armazenar o *dataset*. Nela foi feito o *download*, seguido de um *upload* para um *bucket*.

Detecção de objetos

Para a detecção dos objetos nas imagens foi usado um modelo do tipo ***YOLO-v3***, mais especificamente um modelo pré treinado com o próprio dataset **COCO** (usando as imagens do **train2017**). O modelo retorna o tipo de objeto achado na imagem, sua posição e a acurácia da predição.

Foi usado a biblioteca GluonCV para carregar o modelo diretamente na memória física do computador, aumentando a velocidade de execução do código, uma vez que trata-se de mais de 200MB de dados do modelo. Tal biblioteca também auxilia na transformação e manipulação de matrizes posteriormente.

Amazon Cluster e PySpark

Com o *dataset* no *bucket* e o classificador pronto, usou-se a API de Spark para Python (PySpark), para detectar os objetos em todas as imagens de maneira paralelizável em um Cluster do serviço Elastic Map Reduce (da AWS) e o Apache Hadoop.

No código:

- Foi usada a biblioteca Boto3 para conectar-se no S3 e guardar uma lista com os nomes das imagens no banco
- Baixa-se o modelo para a descrição
- Cria-se um RDD com a lista de arquivos das imagens, onde aplica-se uma transformação do tipo *map* que roda, para cada imagem, uma função responsável por ler a imagem para a RAM, classificá-la e retornar as *labels* detectadas e suas respectivas acurácias. (Para a leitura usa-se a função *imdecode* da biblioteca OpenCV.
- Usa-se uma ação no RDD para recolher os resultados

Criando o Banco no MySQL

Foi feito um script de criação de um banco de dados MySQL, que armazenaria o *dataset* trabalhado no item anterior com as tabelas: Imagem (contendo o caminho para a imagem e um id), Label (contendo o nome da *label* e um id), e a Detected (contendo os ids da label e da imagem e a acurácia da detecção), vide Figura 1.

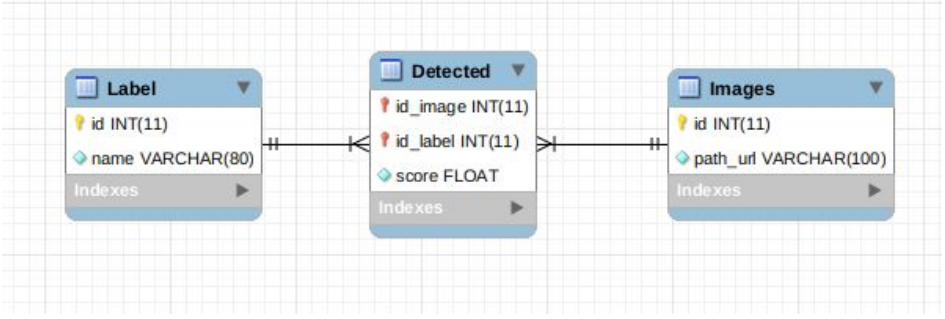

Figura 1 - Modelagem entidade-relacionamento do banco

Dificuldades

- Foi visto que já existia o dataset em um bucket público da AWS porém as imagens continuavam comprimidas e não era possível acessá-las dessa forma. Uma solução foi fazer o download de todo o dataset em uma

instância EC2 da Amazon AWS, descomprimir as imagens e logo após fazer um upload para um Bucket privado na AWS S3

- Através da ferramenta Zeppelin do EMR da Amazon, foi possível listar as imagens que estavam armazenadas no facilmente, porém não foi possível, da primeira vez, acessá-las como uma array numpy (o tipo certo para processar a imagem posteriormente). Para isso, foi aberto uma conexão com o Bucket onde era armazenado as imagens do dataset, cada imagem era salva em um objeto tipo BytesIO do python (basicamente um tipo de arquivo que é armazenado na memória RAM) lida como uma numpy array usando um conjunto de funções oriundas das bibliotecas *Numpy* e *openCV* e depois processada.

- As instâncias do cluster, que não fossem a principal, não conseguiam acessar o modelo pré treinado de detecção, baixado na pasta MXnet. Após um tempo tentando contornar esse problema, chegamos a conclusão que só seria possível executar nosso script em um cluster com uma máquina.

- A principal dificuldade foi trabalhar nessas ferramentas com essa quantidade de imagens. Não importa o tamanho da máquina que fosse instanciada (chegamos a usar uma com 160Gb de memória RAM), sempre ocorria problemas de memória. As mensagens de erro eram prolixas e pouco elucidativas. Diversas alternativas de conserto foram testadas, até chegarmos à conclusão que não era possível arrumar isso nesse projeto. O script estava funcionando para 1000 imagens somente, nota-se no script enviado a tentativa de fazer a detecção de 1000 em 1000 imagens, salvando cada lote em um arquivo *pickle* diferente. Infelizmente isso também nos retornou problemas de memória.