IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Airbnb's Amenity Detection with Detectron2

Project guided by:
Shri krishna balwante
Dept of datascience(mca)

Y SRILEKHA
1993-M-19031993
Dept of MCA data science
Ajeenkya dy patil university
Pune,India.

VAISHALI BOBADE
2021-M-12072000
Dept of MCA data science
Ajeenkya dy patil university
Pune ,India

ABSTRACT

Hosts are often desperate to find ways to rent their house, However, most of them do not have possess the knowledge of knowing what type of image cover would grasp the attention of their customer. You have thousand properties ,it's in your best interest to make sure that the information about those properties are correct using computer vision to aid with amenity detection, could automatically add information to a property listing whether or not it has a kitchen, refrigerator, dining table etc.

KEYWORDS

Detectron2 , weights and biases , streamlit or very much of PyTorch
(machine learning tools)

1. INTRODUCTION

Rental service, or homestay service, is a common sight these days. Its history are so old that the origin is probably lost in prehistory. In medieval times, most land was owned either by the King or by lords, and almost all farmers were tenant farmers who paid a rent - usually a percentage or portion of crops grown - in return for living on and farming the land with. Nowadays , rental service is a popular form of hospitality and lodging, whereby visitors share a residence with a local of the area to which they are traveling.

If we have seen AirBnb's website there are lots of photos of homes and places to stay.Alongside these places to stay are text-based details describing the finer-details. Like whether or not the listing(unless you're a postal service in the business of detecting handwritten digits).

As you upload an image to Airbnb a computer vision machine learning model looks at the images,tries to find the key amenities in each one and adds them to your listing automatically.
It could verify with you that its predictions were correct before they actually go live , but making it happen automatically would help to make sure the information on each listing is as filled out as possible

Having a detailed information about each listing means the people searching for places with specific criteria.

2. LITRETURE REVIEW

Lacking prior experience, we decided to start from something people had worked on before, hopefully to find some hint. We found that Open Image Dataset V4 offered a vast amount of image data. It included about 9M images that had been annotated with image-level labels, object bounding boxes (BB), and visual relationships. In particular, BB annotations span a rich set of 600 object classes. These classes formed a hierarchical structure, and covered a wide spectrum of objects. On the highest level, they included *Animal*, *Clothing*, *Vehicle*, *Food*, *Building*, *Equipment, Tool,* and a collection of household items, such as *Furniture*, *Appliances,* and *Home Supplies*. Our goal was to find out the object classes that were relevant to amenities and to filter out the rest.

We manually reviewed the 600 classes and selected around 40 classes that were relevant to our use case. They were generally important amenities in *kitchen*, *bathroom*, and *bedroom*, such as *gas stove*, *oven*, *refrigerator*, *bathtub*, *shower*, *towel*, *bed*, *pillow*, etc. Open Image Dataset V4 saved us a lot of time. If we were to start from scratch, building a reasonable taxonomy alone would have taken us a long time.

In order to build the project, a few challenges have been identified as follows.

3. METHODOLOGY (EXPERIMENT)

3.1.Experiment 1

We can apply detector2 for any custom dataset to generate the results using six steps. These steps are accessible on google Colab and can be run efficiently. We utilize GPU in this article for faster results generation.

3.2.*Step 1: Installation of detector2*

COCO API and torch vision are some of the dependencies that should be installed first and later on check the availability of CUDA. Tracking of the presently chosen GPU with the help of CUDA. Afterward, download and execute detectron2.

3.3.*Step 2: Generate the dataset and register it*

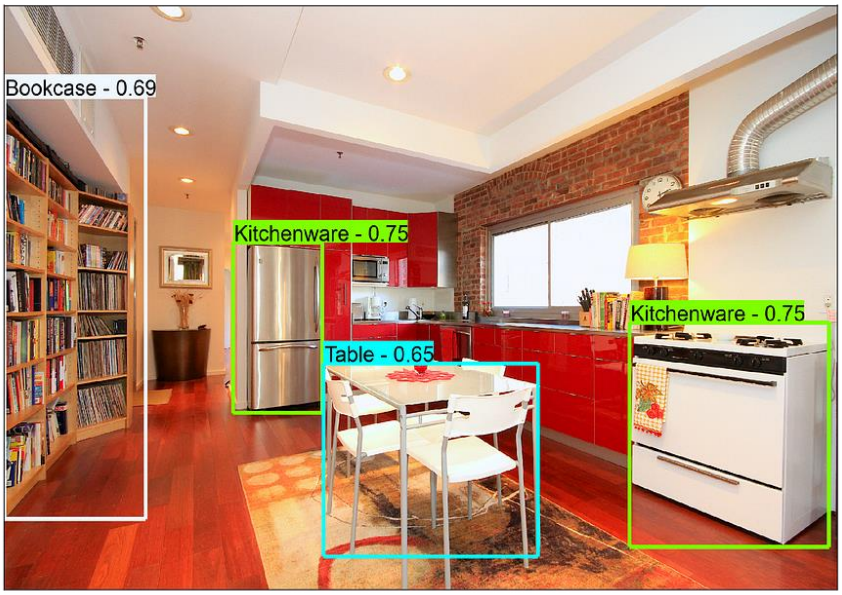
In this step, firstly, we should install all the necessary packages. Some datasets are listed by default in dtector2 and if you want to train the model with the custom dataset, you should register it. We will train our model with the help of the detectron2 model zoo, which is already pre-trained on the COCO dataset. Dtectron2 accepts only those datasets that are in COCO format. The format of COCO accepts JSON file that includes information about image size etc. Currently, only two formats like BoxMode.XYWH_ABS and BoxMode.XYXY_ABS are supported by detectron2 but we utilize the only first format.

3.4.*Step 3: Training set is Visualized*

That is the major step in which the model's configuration takes place and makes it ready for training. While the model is already pre-trained on the COCO dataset and we have to fit it only according to our dataset. There is different dataset exist in detectron2 but we utilize only faster_rcnn_R_50_FPN_3X. The resent network that is considered the backbone network is utilized to extract the features from the image.

3.5.*Step 5: Utilizing the Trained Model for Reasoning*

In this step, the findings of the model are determined by utilizing a validation set. The below figure shows the results of the training model.



A sample amenity detection result of a third-party API service, from an industry-leading vendor

Even though the API service is able to detect certain amenities, the labels predicted are too vague. In a home sharing business like Airbnb, knowing that *kitchenware* exists in a picture does not tell us much other than the room type. Likewise, knowing there is a table in the picture doesn't help us either. We don't know what kind of a table it is, or what it could be used for. Our actual goal is to understand whether the detected amenities provide convenience for guests

A sample result from Airbnb's amenity detection model, with more specific labels

As one can see, the predicted labels are much more specific. We can use these results to verify the accuracy of listing descriptions and serve Homes searches to guests with specific amenity requests.

Our first hypothesis is that the percentage error of the predicted value from our model and actual value from the Airbnb page will within 5 percent. To prove our hypothesis, we use images from the existing airbnb and predict its rating. Next, We perform the same procedure 4 more times using different datasets (image). We take the prediction values, and it's corresponding actual value in airbnb for analysis. We calculated the percent error to contrast them to see the accuracy of our model. This process is completed by using 5 PLACES (experiment image set) from airbnb using its ID and URL. The average percent error turns out to be 1.456 percent, which indicate that the project has worked well.
Like all machine learning projects, it begins and ends with the data.

The Airbnb article mentioned to build their proof of concept, they used 32k public images and 43k internal images. The good news was the 32k public images they used were from Open Images (a massive free and open-source resource of millions of images from 600+ categories of different things).

```
!python3 downloadOI.py --dataset "validation" --classes "Kitchen & dining room table"
```

This line says, "get the images with kitchen & dining room tables in them from the validation set of Open Images."

Running the line of code above will download all of the images from the target class(es) into a folder with the same name as the target dataset.

```
# What your image folder(s) look like after using downloadOI.py
validation <- top file
|   kitchen_&_dining_room_table_image_1.jpg <- image 1
|   kitchen_&_dining_room_table_image_2.jpg <- image 2
|   ... <- more images...
```

The original downloadOI.py from LearnOpenCV script worked great, it even generated labels as the images downloaded. But after a little experimentation, I found these labels incompatible for working with Detectron2.

So I scrapped the label creation on download and modified the script to only download images. I decided I'd write my own label creating code.

The Open Images labels are a bit more accessible and can be downloaded by clicking the specific download links on the Open Images download page or by running the following code (tidbit: scroll to the bottom of the download page to see information about the labels).

```
# Open Images training dataset bounding boxes (1.11G)
!wget https://storage.googleapis.com/openimages/2018_04/train/train-annotations-bbox.csv

# Open Images validation dataset bounding boxes (23.94M)
!wget https://storage.googleapis.com/openimages/v5/validation-annotations-bbox.csv

# Open Images testing bounding boxes (73.89M)
!wget https://storage.googleapis.com/openimages/v5/test-annotations-bbox.csv

# Class names of images (11.73K)
!wget https://storage.googleapis.com/openimages/v5/class-descriptions-boxable.csv
```

The first stage of data collection resulted in acquiring the following files.

- 1 class (coffeemaker) of images from Open Images train, validation and test sets.
- Training (train-annotations-bbox.csv), validation (validation-annotations-bbox.csv) and test (test-annotations-bbox.csv) set bounding box image labels.
- The descriptions of different Open Images classes (class-descriptions-boxable.csv

4. CONCLUSIONS

What we have done is proposing an application to predict the rating of a house by the attribute inside it's image by first collecting the raw data set from a house rental company name airbnb that contains the ID, URL, and the rating of the house. Then we Fetch the URL from the webpage by locating the image resource in the JSON and extracting that section of the JSON value out to return the URL of the image. Afterward, we downloaded the image from the URL that we extracted from the previous step to created a folder so that each place(web page) will have its own folder for image. To ensure that we got all the places, we keep repeating the last 2 steps in a loop for every element in our raw dataset. Next up, we imported the pre-build model Yolov5 to count different items of the image in the imputed data set that we have made into a data frame [15]. Lastly, use machine learning to train data of the input value x(data frame) and the output value y which is the "rating" value from the raw data set. We build a model for future use and apply the application to experiment by predicting rating from a new "place" other than the raw data set. The experiment results indicate its effectiveness and solve challenges because it successfully found all the input value and the rating was extremely close to the actual rating in the airbnb.

There are still quite a few limitations in our application, first of all, the data is considerably small, so the accuracy might not be enough to cover all, secondly all data used for machine learning are from airbnb, although Airbnb is a considerably large website, it might still affect the accuracy, in addition, the current version of our project does not consider location and outdoor view, this might also affect practicality of some user. In the future optimization, we plan on adding more variables like how many customers selected the price rate, how far away is the house, or how many people rated and percent of how many people like it.

In the future, I plan on adding a lot more database to improve the accuracy of the project, another limitation I will solve is the renter's location, I plan in the future to extract the renter's location JSON out of the page source so that it could also be placed in the machine learning.

REFERENCES

[1] https://analyticsvidhya.com/blog/2018/01/facebook-launched-detectron-platform-object-detection-research/
[2] https://towardsdatascience.com/image-labelling-using-facebooks-detectron-4931e30c4d0c
[3] Archana B. Patankar; Purnima A. Kubde; Ankita Karia (Aug. 2016). Image cartoonization methods IEEE
[4] Vung Pham; Chau Pham; Tommy Dang (2020). Road Damage Detection and Classification with Detectron2 and Faster R-CNN 20511275 IEEE