[Amazon Go]

[LongTimeNoC]

**Data Science Capstone Project   
Data Acquisition and Pre-Processing Report**

Date:

[10/27/2020]

Team Members:

Name: Himanshu Jat

Name: Smith Amornsaensuk

Name: Chingiz Mardanov

Name: Harshit Geddam

# Identifying Data

### Introduction

This report describes the data used in the Amazon Go project. Major components of this report describe the data sources and how the data have been acquired. Furthermore, issues of the dataset are discussed along with how they are handled. The source code of data-processing is included in Appendix.

### Data Sources:

The Amazon Go project mainly focuses on object detection and classification from images. Therefore, the dataset used are all images of objects in the grocery stores environment. Furthermore, the object detection API that we will be using requires information of objects from images in xml format. Fortunately, the xml information file can be generated from Labelimg application which is available as a open source.

Since our project is quite complex we have decided to use multiple sources of data. For the first source we decided to use a labeled dataset called Holoselecta. This datasource consists of 295 pictures of vending machines in the Zurich area. That’s said, we had to identify the objects in these images that can be easily found in the US stores. After exploring the data we decided to keep the following objects:

1. Redbull
2. Snickers
3. Twix
4. Evian
5. 7Days
6. Skittles
7. Maltesers
8. Coke

After we have selected this set of objects, we have filtered the original dataset and ended up with 291 images in total. Object count was reduced from 9120 objects in total to 1945.

As our second source of data we have decided to manually parse the internet in order to find images of the 8 objects listed above. We found 75 images with 258 objects on them. In order to label the data we used the LabelIMG tool. This tool allows for convenient labeling with GUI and outputs the XML files that are identical to those in our first pre-labeled dataset. We will discuss the labeling process in greater detail in the next section. Besides that we also had to extract the individual objects from the images of the vending machines so we can train without being tight to the layout of the items. We will discuss that in the upcoming sections as well.

Heloselecta dataset:<https://drive.google.com/drive/folders/1OofnsREqF1QNfUqZPZgcXu2VkWr2JUMA>  
Labelimg tool: <https://github.com/kartua/capstone1/tree/main/labelImg>

### Acquisition Process:

To reach our problem goal, we have decided to obtain the training data in two ways.

1. Self-labeled- We collected the image data from different sources manually, and then we have decided to use the LabelImg inorder to draw the bounding boxes and label the images such that we could send the labeled data for the training. So we have totally collected 75 images and then we have drawn the bounding boxes and then found out to have a total of 258 objects. Which includes items like redbull, coke can, snickers, twix, evian, 7days croissant, skittles sour, maltesers.
2. Pre-labeled- Then we have decided to train our model in another way round, so we have collected the data from a data source. This source has a total of 291 image data from where we have obtained a total of 1945 objects. This is enough to train our model to get good results.

We need not write any kind of code in order to obtain the data, the data acquisition was totally based on the above two ways. We have tried to solve our problem statement by taking most of the public data as all of our image data was available in open source, so we have collected all of the data from different open sources. So now as we have collected the data in two ways, we have a total of 2203 objects which we have collected from 366 images, so we would be training our model based on these 2203 objects.

|  |  |  |
| --- | --- | --- |
|  | Images | Objects |
| Pre-Labeled | 291 | 1945 |
| Self-Labeled | 75 | 258 |
| Total | 366 | 2203 |

Table 1: Instances in dataset.

### Issues:

We are using the data from two different sources thus we were facing different kinds of issues:

* The vending machine dataset consists of different kinds of images compared with the manually downloaded data. Since we are dealing with a computer vision problem so making sure the linearity of the images we are using is a must. Thus when we manually downloaded the images we made sure that we are picking images that contain several items in a single image.
* The data we gathered from the vending machine dataset has a vast distribution in the number of images of different items. Thus we picked those items from the vending machine dataset that have a maximum number of images present along with the availability of the item in the nearby stores.
* Another issue we faced was related to the XML files present in the vending machine dataset as we had to convert them in a format that can match with the manually downloaded and labeled images data(described further in the data processing section)

### Data-Processing:

Since our data are images and xml files generated from Labelimg, there is no issue of missing data, sparsity, noise, veracity or ambiguity. However, we have created the label from images by ourselves and integrate with the other sources. Therefore, there are some data-processing procedures that need to be performed.

The data processing part consists of 2 steps:

1. Labeling the images
2. Matching the different XML files generated by images from different sources

#### Labeling the images

For labeling the images we used an open-source tool called Labelimg(see Appendix A). We setup Labelimg in our local systems and drew the bounding boxes on the items in the images from the second data source(i.e. manually downloaded images). The tool then converted the coordinates of bounding boxes into XML files (see appendix A). The information provided from Labelimg such as label tag and coordinate of the object in an image.

#### Matching the different XML files generated by images from different sources

Since one of our data sources provided xml files from Labelimg, we need to ensure that information of xml files from images that we labeled are consistent and match with xml files from other sources. Firstly, because xml files from other sources contain objects that we will not use in the model, they need to be deleted (see Appendix B). As a result, there are only 8 labels remaining in our xml files i.e. redbull, snickers, evian, coke, twix, skittles, 7days croissant and maltesers. Finally, the label name from data in different sources needs to be consistent according to each object, the prelabeled dataset needs to be modified to match our labels (see Appendix B). After eliminating irreverent objects and matching all labels we end up with a number of instances as shown in Table 1.

#### Cropping Images for Analysis purpose

In order to perform further data analysis, images of each object are extracted from original images which results in 2,203 small images of objects. The process has been done by leveraging the coordinate information of each object from xml files which are used to locate the position of objects in an image (see Appendix A). The cropped image files are named according to objects which end up with 2,203 files in total.

### Conclusion

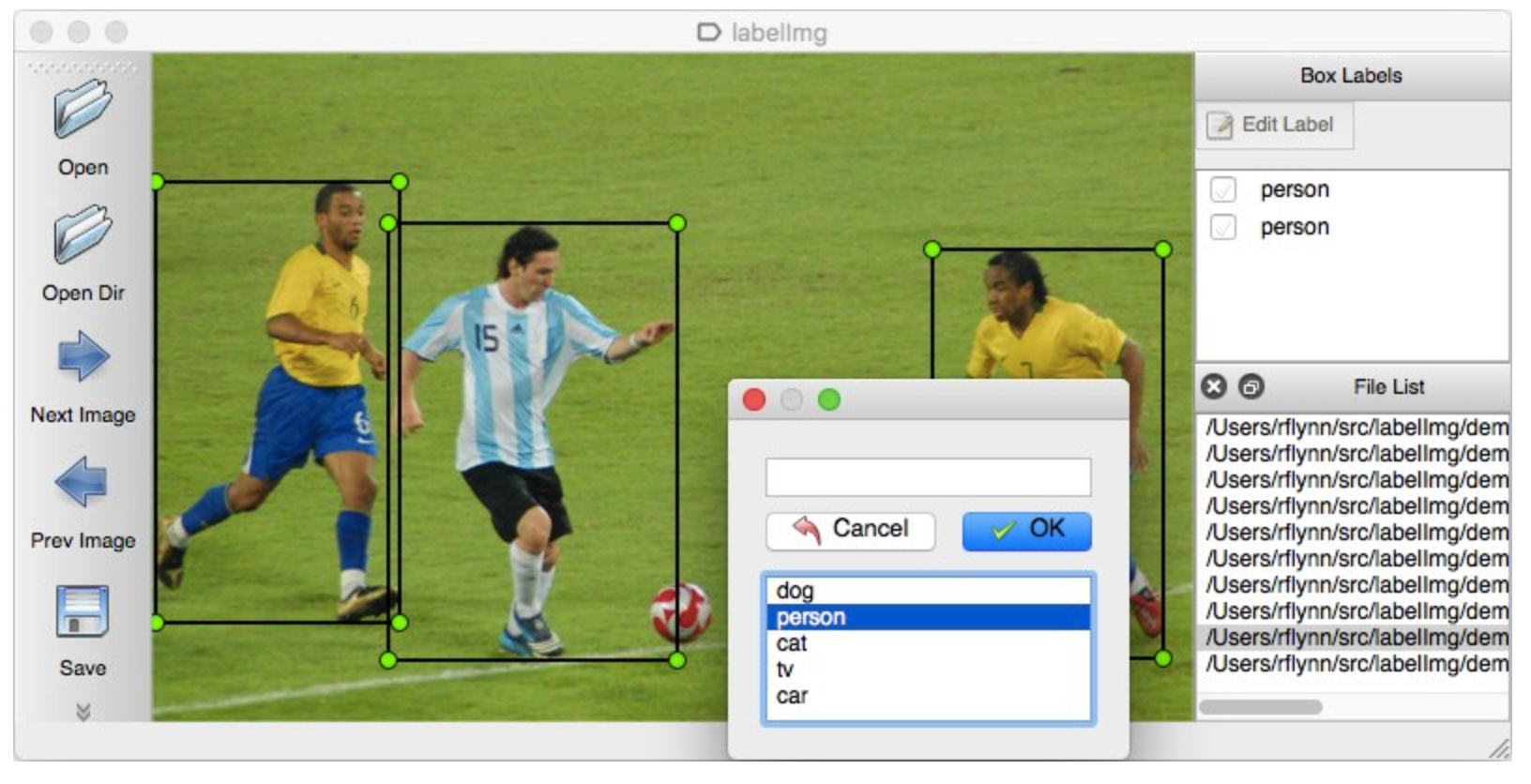
The main task of the Amazon Go project is to perform object detection and classification on images data. Data are obtained from many different sources i.e. google searching and pre-labeled data from github. Each image from google searching is labeled to generate xml files of objects’ information which will be used in object detection API. The xml files are processed to ensure dataset’s consistency before using in the API. The information in xml files are also used to extract each object from original images for the purpose of data analysis. The next procedure for the project is to furthurly perform data analysis and config API to train a model.

**Appendix A**

Example of xml files generated from Labelimg

|  |
| --- |
| -<annotation>  <folder>cleaning5</folder>  <filename>20190111\_111600\_HoloLens.jpg</filename>  <path>C:\Users\tg\Desktop\cleaning5\20190111\_111600\_HoloLens.jpg</path>  -<source>  <database>Unknown</database>  </source>  -<size>  <width>1408</width>  <height>792</height>  <depth>3</depth>  </size>  <segmented>0</segmented>  -<object>  <name>snickers</name>  <pose>Unspecified</pose>  <truncated>0</truncated>  <difficult>0</difficult>  -<bndbox>  <xmin>902</xmin>  <ymin>388</ymin>  <xmax>966</xmax>  <ymax>518</ymax>  </bndbox>  </object>  -<object>  <name>evian</name>  <pose>Unspecified</pose>  <truncated>0</truncated>  <difficult>0</difficult>  -<bndbox>  <xmin>590</xmin>  <ymin>532</ymin>  <xmax>668</xmax>  <ymax>742</ymax>  </bndbox>  </object> </annotation> |
|  |

Labelimg interface



**Appendix B**

|  |
| --- |
| # Delete irrelevant objects from xml files import xmltodict import os from collections import OrderedDict import copy import glob from PIL import Image import re  def reduce\_xml(file\_name):  target\_directory = "./Data/CleanXML"  with open(file\_name, "r") as f:  dic = xmltodict.parse(f.read())  new\_xml = copy.deepcopy(dic)  new\_xml['annotation']['object'] = []  for item in dic['annotation']['object']:  if isinstance(item, str):  print('passing')  continue  if item['name'] in ITEM\_LIST:  print(f"Match: {item['name']}")  new\_xml['annotation']['object'].append(item)  with open(os.path.join(target\_directory, filename), 'w') as f:  f.write(xmltodict.unparse(new\_xml)) ITEM\_LIST = ['redbull\_\_\_33\_\_90162909', 'snickers\_\_\_50\_\_5000159461122', 'twix\_\_\_50\_\_5000159459228', 'evian\_\_\_50\_\_3068320353500', '7days\_croissantschoko\_packung\_80\_1\_0000000000003', 'skittles\_sour\_riegel\_51\_1\_0000000000004', 'maltesers\_\_\_100\_\_5000159023061', 'coke\_\_dose\_33\_\_54491472'] directory = "./Data/" xml\_files = [] for filename in os.listdir(directory):  if filename.endswith(".xml"):  reduce\_xml(os.path.join(directory, filename))  else:  continue |
| # Rename labels to match all dataset  def read\_xmls(folder\_path):  '''   return a tuple  (pic filename, list of objects) '''  folder\_path += '/\*.xml'  file\_name = glob.glob(folder\_path)  file\_use = []  data = []  for file in file\_name:  data\_read = xmltodict.parse(open(file,'r').read())  if 'object' in data\_read['annotation']:  data.append((file, data\_read['annotation']['object']))  return data  def get\_obj\_name(folder\_path):  '''  arg:folder path  return:all object\_name form xml files in the folder  '''  folder\_path += '/\*.xml'  file\_name = glob.glob(folder\_path)  s\_name = set()  for file in file\_name:  data = xmltodict.parse(open(file,'r').read())  if 'object' in data['annotation']:  if isinstance(data['annotation']['object'],list):  for item in data['annotation']['object']:  s\_name.add(item['name'])  else:  s\_name.add(data['annotation']['object']['name'])  return s\_name get\_obj\_name('./CleanXML')  # get file names that not include our object (to delete the xml files that we don’t use automatically) def file\_not\_use(folder\_path):  '''   get the file name that won't be used  '''  folder\_path += '/\*.xml'  file\_name = glob.glob(folder\_path)  file\_use = []  data = []  for file in file\_name:  data\_read = xmltodict.parse(open(file,'r').read())  if 'object' not in data\_read['annotation']:  data.append(file)  return data not\_use = file\_not\_use('./dataClean') all\_not\_use = [] for file in not\_use:  all\_not\_use.extend(glob.glob(file[:-3]+'\*')) # change object name to match our own label file\_names = glob.glob('./dataClean/\*.xml') name\_dict = {'7days\_croissantschoko\_packung\_80\_1\_0000000000003': '7days',  'coke\_\_dose\_33\_\_54491472':'coke',  'evian\_\_\_50\_\_3068320353500':'evian',  'maltesers\_\_\_100\_\_5000159023061':'maltesers',  'redbull\_\_\_33\_\_90162909':'redbull',  'skittles\_sour\_riegel\_51\_1\_0000000000004':'skittles',  'snickers\_\_\_50\_\_5000159461122':'snickers',  'twix\_\_\_50\_\_5000159459228':'twix'} for file in file\_names:  with open(file,'r',encoding='utf-8') as f:  xml = f.read()  for name in name\_dict:  xml = re.sub(name,name\_dict[name],xml)  with open(file,'w') as f:  f.write(xml)  # Cropping Images  crop\_pic = defaultdict(list) data = read\_xmls('./dataFinal') for i,each\_pic in enumerate(data):  if (i+1) % 36 == 0:  print(f'progress: {((i//36)+1)\*10}%')  names = glob.glob(each\_pic[0][:-3] + '\*\*')  for name in names:  if not re.search('xml',name):  pic\_name = name   im = Image.open(pic\_name)  im\_list = []  if isinstance(each\_pic[1],list):  for obj in each\_pic[1]:  c\_im = im.crop((int(obj['bndbox']['xmin']),  int(obj['bndbox']['ymin']),  int(obj['bndbox']['xmax']),  int(obj['bndbox']['ymax'])))  crop\_pic[obj['name']].append(c\_im)  else:  c\_im = im.crop((int(each\_pic[1]['bndbox']['xmin']),  int(each\_pic[1]['bndbox']['ymin']),  int(each\_pic[1]['bndbox']['xmax']),  int(each\_pic[1]['bndbox']['ymax'])))  crop\_pic[each\_pic[1]['name']].append(c\_im) progress = 1 for item in (crop\_pic):  for i, im in enumerate(crop\_pic[item]):  if progress % 220 == 0:  print(f'pregress: {(progress//220)\*10}%')  im.save(f'./cropPics/{item}{i}.png')  progress += 1 |

Table of Contributions

The table below identifies contributors to various sections of this document.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Section** | **Writing** | **Editing** |
| **1** | **Data Sources** | **Chingiz Mardanov** | **Smith Amornsaensuk** |
| **2** | **Acquisition Process** | **Harshit** | **Harshit** |
| **3** | **Data Pre-Processing** | **Himanshu Jat, Smith Amornsaensuk** | **Himanshu Jat** |
| **4** | **Appendix** | **Smith Amornsaensuk** |  |

**Grading**

The grade is given on the basis of quality, clarity, presentation, completeness, and writing of each section in the report. This is the grade of the group. Individual grades will be assigned at the end of the term when peer reviews are collected.