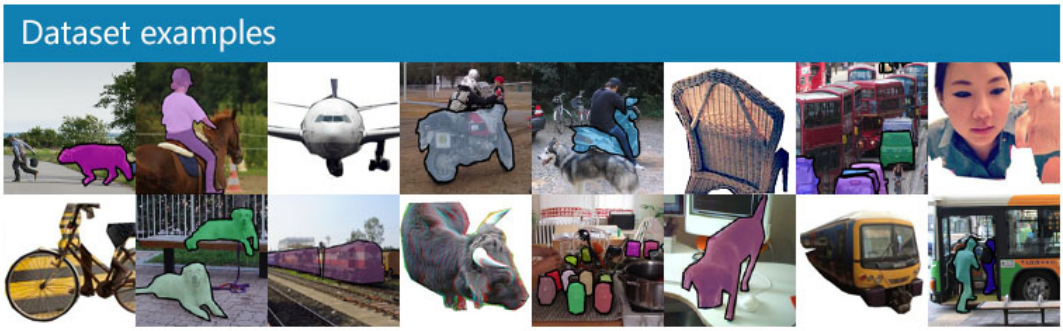
0_Dataset

October 19, 2024

1 Computer Vision Nanodegree

1.1 Project: Image Captioning

The Microsoft **C**ommon **O**bjects in **CO**ntext (MS COCO) dataset is a large-scale dataset for scene understanding. The dataset is commonly used to train and benchmark object detection, segmentation, and captioning algorithms.



You can read more about the dataset on the website or in the research paper.

In this notebook, you will explore this dataset, in preparation for the project.

1.2 Step 1: Initialize the COCO API

We begin by initializing the COCO API that you will use to obtain the data.

[2]:
```
import os
import sys
sys.path.append('/opt/cocoapi/PythonAPI')
from pycocotools.coco import COCO

# initialize COCO API for instance annotations
dataDir = '/opt/cocoapi'
dataType = 'val2014'
instances_annFile = os.path.join(dataDir, 'annotations/instances_{}.json'.
 ↪format(dataType))
coco = COCO(instances_annFile)
```

```
# initialize COCO API for caption annotations
captions_annFile = os.path.join(dataDir, 'annotations/captions_{}.json'.
 ↪format(dataType))
coco_caps = COCO(captions_annFile)

# get image ids
ids = list(coco.anns.keys())
```

```
loading annotations into memory…
Done (t=3.85s)
creating index…
index created!
loading annotations into memory…
Done (t=0.23s)
creating index…
index created!
```

[11]:
```
list(coco.anns.values())[0]
```

[11]:
```
{'segmentation': [[510.66,
   423.01,
   511.72,
   420.03,
   510.45,
   416.0,
   510.34,
   413.02,
   510.77,
   410.26,
   510.77,
   407.5,
   510.34,
   405.16,
   511.51,
   402.83,
   511.41,
   400.49,
   510.24,
   398.16,
   509.39,
   397.31,
   504.61,
   399.22,
   502.17,
   399.64,
   500.89,
```

401.66,
500.47,
402.08,
499.09,
401.87,
495.79,
401.98,
490.59,
401.77,
488.79,
401.77,
485.39,
398.58,
483.9,
397.31,
481.56,
396.35,
478.48,
395.93,
476.68,
396.03,
475.4,
396.77,
473.92,
398.79,
473.28,
399.96,
473.49,
401.87,
474.56,
403.47,
473.07,
405.59,
473.39,
407.71,
476.68,
409.41,
479.23,
409.73,
481.56,
410.69,
480.4,
411.85,
481.35,
414.93,
479.86,
418.65,

```
477.32,
420.03,
476.04,
422.58,
479.02,
422.58,
480.29,
423.01,
483.79,
419.93,
486.66,
416.21,
490.06,
415.57,
492.18,
416.85,
491.65,
420.24,
492.82,
422.9,
493.56,
424.39,
496.43,
424.6,
498.02,
423.01,
498.13,
421.31,
497.07,
420.03,
497.07,
415.15,
496.33,
414.51,
501.1,
411.96,
502.06,
411.32,
503.02,
415.04,
503.33,
418.12,
501.1,
420.24,
498.98,
421.63,
500.47,
```

```
    424.39,
    505.03,
    423.32,
    506.2,
    421.31,
    507.69,
    419.5,
    506.31,
    423.32,
    510.03,
    423.01,
    510.45,
    423.01]],
  'area': 702.1057499999998,
  'iscrowd': 0,
  'image_id': 289343,
  'bbox': [473.07, 395.93, 38.65, 28.67],
  'category_id': 18,
  'id': 1768}
```

1.3 Step 2: Plot a Sample Image

Next, we plot a random image from the dataset, along with its five corresponding captions. Each time you run the code cell below, a different image is selected.

In the project, you will use this dataset to train your own model to generate captions from images!

[3]:
```
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
%matplotlib inline

# pick a random image and obtain the corresponding URL
ann_id = np.random.choice(ids)
img_id = coco.anns[ann_id]['image_id']
img = coco.loadImgs(img_id)[0]
url = img['coco_url']

# print URL and visualize corresponding image
print(url)
I = io.imread(url)
plt.axis('off')
plt.imshow(I)
plt.show()

# load and display captions
annIds = coco_caps.getAnnIds(imgIds=img['id']);
anns = coco_caps.loadAnns(annIds)
```

```
coco_caps.showAnns(anns)
```

```
http://images.cocodataset.org/val2014/COCO_val2014_000000072539.jpg
```

```
Several sailboats in the water docked at a marina.
several boats docked at a marina with clear water
Numerous boats displayed in a lake outside large buildings
A row of boats in a harbor next to buildings.
A bunch of boats floating in the water
```

1.4 Step 3: What's to Come!

In this project, you will use the dataset of image-caption pairs to train a CNN-RNN model to automatically generate images from captions. You'll learn more about how to design the architecture in the next notebook in the sequence (**1_Preliminaries.ipynb**).

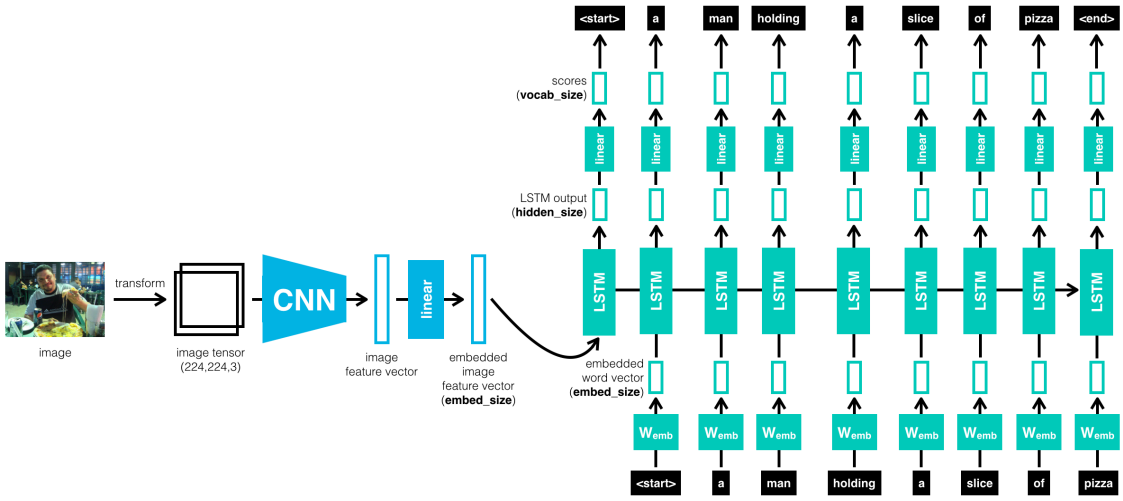


image
transform
image tensor
(224,224,3)
CNN
image
feature vector
linear
embedded
image
feature vector
(embed_size)
embedded
word vector
(embed_size)
LSTM output
(hidden_size)
scores
(vocab_size)
LSTM
linear
Wemb
<start>
a
man
holding
a
slice
of
pizza
<end>

[ ]: