analyse_coco

May 5, 2021

1 Analysing COCO dataset

The COCO dataset is widely used for training visual ML models. We will produce some statistics that will be used later in the defintion of the AttentionNET model that splits between identifying the focus in a picture and identifying the object in the focus.

First we load the modules we will use:

[2]:
```
%matplotlib inline
from pycocotools.coco import COCO
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
from matplotlib.ticker import PercentFormatter
import pandas as pd
```

Based on the examples from COCO gothub we will setup a directory to load the annotation data. Since this is the largest dataset we will use the training 2017 dataset and we will focus on the "instances" classifications. The dataset annotations are pre-downloaded in the `annotations` directory:

[3]:
```
dataDir='.'
dataType='train2017'
annFile='{}/annotations/instances_{}.json'.format(dataDir,dataType)
```

We now setup a COCO object to deal with the annotations data and to parse it:

[4]:
```
# initialize COCO api for instance annotations
coco=COCO(annFile)
```

```
loading annotations into memory…
Done (t=12.85s)
creating index…
index created!
```

We can analyse the categories from the annotations file. This is provided in the member `cats` of the `COCO` object. This is a dictionary with the key the category ID and as attributes: - `name`: the name of the category - `supercategory`: the name of the supercategory that groups more categories together - `id`: the ID of the category (again although is already the key of the dictionary):

[5]:
```
coco.cats
```

[5]:
```
{1: {'supercategory': 'person', 'id': 1, 'name': 'person'},
 2: {'supercategory': 'vehicle', 'id': 2, 'name': 'bicycle'},
 3: {'supercategory': 'vehicle', 'id': 3, 'name': 'car'},
 4: {'supercategory': 'vehicle', 'id': 4, 'name': 'motorcycle'},
 5: {'supercategory': 'vehicle', 'id': 5, 'name': 'airplane'},
 6: {'supercategory': 'vehicle', 'id': 6, 'name': 'bus'},
 7: {'supercategory': 'vehicle', 'id': 7, 'name': 'train'},
 8: {'supercategory': 'vehicle', 'id': 8, 'name': 'truck'},
 9: {'supercategory': 'vehicle', 'id': 9, 'name': 'boat'},
 10: {'supercategory': 'outdoor', 'id': 10, 'name': 'traffic light'},
 11: {'supercategory': 'outdoor', 'id': 11, 'name': 'fire hydrant'},
 13: {'supercategory': 'outdoor', 'id': 13, 'name': 'stop sign'},
 14: {'supercategory': 'outdoor', 'id': 14, 'name': 'parking meter'},
 15: {'supercategory': 'outdoor', 'id': 15, 'name': 'bench'},
 16: {'supercategory': 'animal', 'id': 16, 'name': 'bird'},
 17: {'supercategory': 'animal', 'id': 17, 'name': 'cat'},
 18: {'supercategory': 'animal', 'id': 18, 'name': 'dog'},
 19: {'supercategory': 'animal', 'id': 19, 'name': 'horse'},
 20: {'supercategory': 'animal', 'id': 20, 'name': 'sheep'},
 21: {'supercategory': 'animal', 'id': 21, 'name': 'cow'},
 22: {'supercategory': 'animal', 'id': 22, 'name': 'elephant'},
 23: {'supercategory': 'animal', 'id': 23, 'name': 'bear'},
 24: {'supercategory': 'animal', 'id': 24, 'name': 'zebra'},
 25: {'supercategory': 'animal', 'id': 25, 'name': 'giraffe'},
 27: {'supercategory': 'accessory', 'id': 27, 'name': 'backpack'},
 28: {'supercategory': 'accessory', 'id': 28, 'name': 'umbrella'},
 31: {'supercategory': 'accessory', 'id': 31, 'name': 'handbag'},
 32: {'supercategory': 'accessory', 'id': 32, 'name': 'tie'},
 33: {'supercategory': 'accessory', 'id': 33, 'name': 'suitcase'},
 34: {'supercategory': 'sports', 'id': 34, 'name': 'frisbee'},
 35: {'supercategory': 'sports', 'id': 35, 'name': 'skis'},
 36: {'supercategory': 'sports', 'id': 36, 'name': 'snowboard'},
 37: {'supercategory': 'sports', 'id': 37, 'name': 'sports ball'},
 38: {'supercategory': 'sports', 'id': 38, 'name': 'kite'},
 39: {'supercategory': 'sports', 'id': 39, 'name': 'baseball bat'},
 40: {'supercategory': 'sports', 'id': 40, 'name': 'baseball glove'},
 41: {'supercategory': 'sports', 'id': 41, 'name': 'skateboard'},
 42: {'supercategory': 'sports', 'id': 42, 'name': 'surfboard'},
 43: {'supercategory': 'sports', 'id': 43, 'name': 'tennis racket'},
 44: {'supercategory': 'kitchen', 'id': 44, 'name': 'bottle'},
 46: {'supercategory': 'kitchen', 'id': 46, 'name': 'wine glass'},
 47: {'supercategory': 'kitchen', 'id': 47, 'name': 'cup'},
 48: {'supercategory': 'kitchen', 'id': 48, 'name': 'fork'},
 49: {'supercategory': 'kitchen', 'id': 49, 'name': 'knife'},
 50: {'supercategory': 'kitchen', 'id': 50, 'name': 'spoon'},
```

```
 51: {'supercategory': 'kitchen', 'id': 51, 'name': 'bowl'},
 52: {'supercategory': 'food', 'id': 52, 'name': 'banana'},
 53: {'supercategory': 'food', 'id': 53, 'name': 'apple'},
 54: {'supercategory': 'food', 'id': 54, 'name': 'sandwich'},
 55: {'supercategory': 'food', 'id': 55, 'name': 'orange'},
 56: {'supercategory': 'food', 'id': 56, 'name': 'broccoli'},
 57: {'supercategory': 'food', 'id': 57, 'name': 'carrot'},
 58: {'supercategory': 'food', 'id': 58, 'name': 'hot dog'},
 59: {'supercategory': 'food', 'id': 59, 'name': 'pizza'},
 60: {'supercategory': 'food', 'id': 60, 'name': 'donut'},
 61: {'supercategory': 'food', 'id': 61, 'name': 'cake'},
 62: {'supercategory': 'furniture', 'id': 62, 'name': 'chair'},
 63: {'supercategory': 'furniture', 'id': 63, 'name': 'couch'},
 64: {'supercategory': 'furniture', 'id': 64, 'name': 'potted plant'},
 65: {'supercategory': 'furniture', 'id': 65, 'name': 'bed'},
 67: {'supercategory': 'furniture', 'id': 67, 'name': 'dining table'},
 70: {'supercategory': 'furniture', 'id': 70, 'name': 'toilet'},
 72: {'supercategory': 'electronic', 'id': 72, 'name': 'tv'},
 73: {'supercategory': 'electronic', 'id': 73, 'name': 'laptop'},
 74: {'supercategory': 'electronic', 'id': 74, 'name': 'mouse'},
 75: {'supercategory': 'electronic', 'id': 75, 'name': 'remote'},
 76: {'supercategory': 'electronic', 'id': 76, 'name': 'keyboard'},
 77: {'supercategory': 'electronic', 'id': 77, 'name': 'cell phone'},
 78: {'supercategory': 'appliance', 'id': 78, 'name': 'microwave'},
 79: {'supercategory': 'appliance', 'id': 79, 'name': 'oven'},
 80: {'supercategory': 'appliance', 'id': 80, 'name': 'toaster'},
 81: {'supercategory': 'appliance', 'id': 81, 'name': 'sink'},
 82: {'supercategory': 'appliance', 'id': 82, 'name': 'refrigerator'},
 84: {'supercategory': 'indoor', 'id': 84, 'name': 'book'},
 85: {'supercategory': 'indoor', 'id': 85, 'name': 'clock'},
 86: {'supercategory': 'indoor', 'id': 86, 'name': 'vase'},
 87: {'supercategory': 'indoor', 'id': 87, 'name': 'scissors'},
 88: {'supercategory': 'indoor', 'id': 88, 'name': 'teddy bear'},
 89: {'supercategory': 'indoor', 'id': 89, 'name': 'hair drier'},
 90: {'supercategory': 'indoor', 'id': 90, 'name': 'toothbrush'}}
```

The supercategories can be shown with:

[6]:
```
nms = set([cat['supercategory'] for cat in coco.cats.values()])
print('COCO supercategories: \n{}'.format(' '.join(nms)))
```

```
COCO supercategories:
indoor vehicle electronic accessory food kitchen furniture animal person outdoor
appliance sports
```

1.1 Number of images

Let see now how many images we have in this training set. We can use the member `imgs` of the `COCO` object:

[7]:
```
len(coco.imgs)
```

[7]:
```
118287
```

So we have 118,287 images in this dataset. We are interested to see how many annotations (individual objects) are for each image and how they are distributed.

The `imgs` is a dictionary that uses the picture ID as the key and contains the following information about each image:

[8]:
```
coco.imgs[391895]
```

[8]:
```
{'license': 3,
 'file_name': '000000391895.jpg',
 'coco_url': 'http://images.cocodataset.org/train2017/000000391895.jpg',
 'height': 360,
 'width': 640,
 'date_captured': '2013-11-14 11:18:45',
 'flickr_url': 'http://farm9.staticflickr.com/8186/8119368305_4e622c8349_z.jpg',
 'id': 391895}
```

From this list we are interested at this time by the size of the picture (**`height`** and **`width`**). Later, when training the models we will put in place we will also have to download the images using the `coco_url` provided so that we can do multiple passes on the dataset during training.

1.2 Annotations

Additionally we have in the **`anns`** attibute that contains information about the picture. It is a dictionary with key the annotation **`id`** and then a number of informations related to that annotation:

[9]:
```
coco.anns[156]
```

[9]:
```
{'segmentation': [[239.97,
   260.24,
   222.04,
   270.49,
   199.84,
   253.41,
   213.5,
   227.79,
   259.62,
   200.46,
   274.13,
```

```
    202.17,
    277.55,
    210.71,
    249.37,
    253.41,
    237.41,
    264.51,
    242.54,
    261.95,
    228.87,
    271.34]],
  'area': 2765.1486500000005,
  'iscrowd': 0,
  'image_id': 558840,
  'bbox': [199.84, 200.46, 77.71, 70.88],
  'category_id': 58,
  'id': 156}
```

The interesting attributes of an annotation are the `image_id` for which that annotation was created, the `area` that represents the surrounding areas of the `segmentation` in pixels and the `bbox` that is the bounding box for the item.

To get all the annotations for a given image we can use:

[10]:
```
an_image_id = 475546

annIds = coco.getAnnIds(imgIds=an_image_id, iscrowd=None)
print(annIds)
```

```
[439530, 493130, 499271, 509711, 523544, 662991, 666474, 667097, 667891,
1503979, 1643421, 1720965, 1983599]
```

And if we want to see the details:

[11]:
```
for anno in annIds:
    print(coco.anns[anno])
```

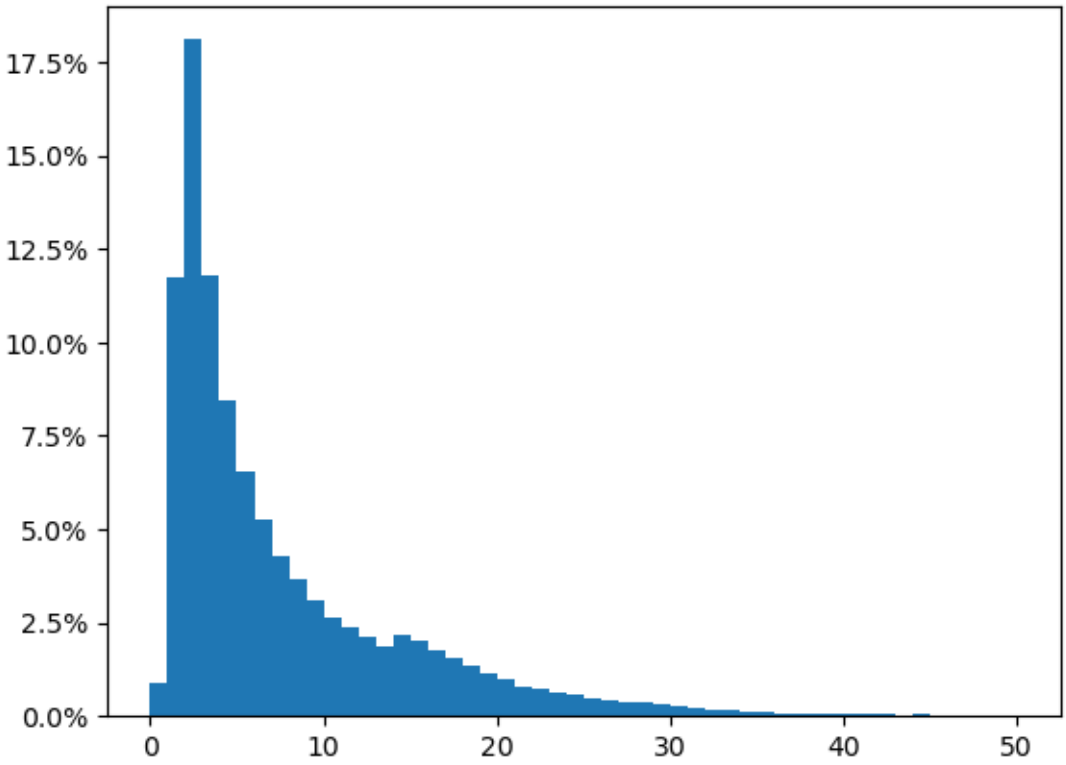
```
{'segmentation': [[303.0, 227.0, 303.0, 221.0, 296.0, 215.0, 292.0, 208.0,
297.0, 199.0, 301.0, 189.0, 308.0, 179.0, 316.0, 169.0, 325.0, 166.0, 326.0,
166.0, 323.0, 159.0, 330.0, 141.0, 335.0, 132.0, 339.0, 130.0, 351.0, 131.0,
358.0, 139.0, 359.0, 146.0, 359.0, 158.0, 363.2, 173.05, 371.11, 178.99, 374.41,
182.29, 376.39, 196.14, 377.05, 204.72, 380.35, 213.95, 382.99, 220.55, 384.31,
231.77, 382.33, 237.05, 375.07, 236.39, 370.45, 231.77, 369.79, 225.83, 361.88,
231.11, 348.68, 231.11, 307.78, 235.07, 306.46, 229.79, 303.16, 223.19]],
'area': 5945.7057, 'iscrowd': 0, 'image_id': 475546, 'bbox': [292.0, 130.0,
92.31, 107.05], 'category_id': 1, 'id': 439530}
{'segmentation': [[413.5, 228.18, 415.38, 220.2, 416.32, 212.21, 415.85, 200.0,
413.5, 196.24, 411.62, 192.48, 413.5, 188.73, 421.01, 184.97, 425.71, 181.21,
429.0, 177.45, 431.82, 173.69, 429.47, 165.71, 429.47, 156.32, 429.47, 153.03,
```

431.35, 148.8, 436.98, 145.04, 442.62, 140.81, 448.73, 138.47, 460.47, 138.94,
467.51, 143.16, 469.86, 153.03, 469.86, 164.77, 465.63, 173.69, 465.17, 175.1,
469.39, 183.56, 475.5, 193.89, 481.14, 206.57, 483.48, 212.21, 484.89, 214.56,
479.26, 218.79, 473.15, 224.42, 472.68, 225.83, 470.33, 222.54, 468.45, 216.91,
468.92, 210.8, 469.86, 204.7, 469.39, 198.12, 456.24, 197.18, 451.54, 201.88,
449.19, 214.56, 449.19, 222.54, 447.32, 227.71, 437.92, 229.59, 425.24, 230.06,
415.85, 230.53]], 'area': 3851.7903999999994, 'iscrowd': 0, 'image_id': 475546,
'bbox': [411.62, 138.47, 73.27, 92.06], 'category_id': 1, 'id': 493130}
{'segmentation': [[252.69, 234.0, 225.02, 234.28, 217.96, 235.13, 214.01,
232.87, 207.8, 233.72, 209.77, 219.32, 210.06, 194.19, 211.47, 176.4, 230.67,
167.93, 225.3, 149.58, 228.41, 136.87, 238.86, 131.79, 252.97, 136.31, 260.32,
145.34, 260.88, 161.16, 260.6, 167.08, 276.97, 173.3, 283.75, 192.78, 290.25,
217.63, 287.42, 216.5, 284.32, 207.74, 284.03, 205.2, 273.31, 204.36, 258.34,
206.61, 260.32, 219.89, 263.14, 234.28]], 'area': 5099.0448, 'iscrowd': 0,
'image_id': 475546, 'bbox': [207.8, 131.79, 82.45, 103.34], 'category_id': 1,
'id': 499271}
{'segmentation': [[111.49, 200.17, 121.62, 190.03, 130.07, 177.36, 132.6,
176.52, 141.89, 176.52, 144.43, 176.52, 145.27, 169.76, 145.27, 161.32, 146.96,
151.18, 151.18, 143.58, 154.56, 141.89, 163.85, 139.36, 173.14, 138.51, 179.05,
143.58, 180.74, 151.18, 183.28, 162.16, 184.97, 168.07, 178.21, 176.52, 184.97,
179.9, 190.03, 182.43, 191.72, 194.26, 194.26, 202.7, 197.64, 213.68, 200.17,
218.75, 200.17, 228.04, 180.74, 236.49, 173.14, 241.55, 160.47, 242.4, 138.51,
238.18, 131.76, 236.49, 127.53, 216.22, 122.47, 211.99, 117.4, 208.61, 110.64,
201.86]], 'area': 5632.724699999999, 'iscrowd': 0, 'image_id': 475546, 'bbox':
[110.64, 138.51, 89.53, 103.89], 'category_id': 1, 'id': 509711}
{'segmentation': [[0.0, 101.97, 14.33, 96.91, 30.34, 96.91, 42.98, 105.34,
48.88, 122.19, 51.4, 136.52, 50.56, 155.9, 59.83, 166.85, 91.01, 179.49, 106.18,
188.76, 116.29, 204.78, 125.56, 216.57, 129.78, 237.64, 125.56, 247.75, 112.08,
247.75, 98.6, 251.12, 103.65, 317.7, 103.65, 338.76, 110.39, 370.79, 26.12,
370.79, 5.06, 359.83, 0.0, 358.99]], 'area': 24938.945050000002, 'iscrowd': 0,
'image_id': 475546, 'bbox': [0.0, 96.91, 129.78, 273.88], 'category_id': 1,
'id': 523544}
{'segmentation': [[472.93, 209.0, 481.89, 208.85, 484.68, 214.0, 483.36, 222.37,
479.54, 225.16, 478.81, 233.24, 479.4, 235.01, 482.19, 236.03, 483.66, 236.92,
482.63, 237.94, 471.61, 238.09, 470.73, 236.77, 474.99, 234.42, 476.75, 225.9,
473.67, 224.28, 471.17, 220.31, 471.46, 209.0]], 'area': 252.64100000000016,
'iscrowd': 0, 'image_id': 475546, 'bbox': [470.73, 208.85, 13.95, 29.24],
'category_id': 46, 'id': 662991}
{'segmentation': [[162.11, 214.2, 170.22, 213.4, 170.86, 218.49, 170.86, 223.58,
167.04, 231.21, 164.98, 236.77, 165.77, 242.97, 168.95, 242.97, 170.54, 243.44,
171.65, 245.03, 169.59, 246.94, 166.88, 247.42, 161.0, 246.94, 157.35, 245.83,
157.82, 244.4, 160.53, 243.13, 162.91, 241.22, 162.11, 231.68, 158.62, 228.19,
156.87, 222.15, 157.35, 214.2]], 'area': 297.0983000000001, 'iscrowd': 0,
'image_id': 475546, 'bbox': [156.87, 213.4, 14.78, 34.02], 'category_id': 46,
'id': 666474}
{'segmentation': [[244.64, 171.72, 256.22, 170.7, 256.66, 178.03, 249.47,
189.46, 243.17, 181.98], [246.69, 196.2, 244.2, 200.02, 252.41, 199.87, 249.18,
195.47]], 'area': 194.01195000000072, 'iscrowd': 0, 'image_id': 475546, 'bbox':

```
[243.17, 170.7, 13.49, 29.32], 'category_id': 46, 'id': 667097}
{'segmentation': [[335.98, 161.72, 343.64, 160.89, 344.8, 163.05, 344.97,
168.21, 343.47, 172.87, 340.81, 175.7, 335.15, 173.87, 333.15, 172.54, 331.65,
169.88, 331.16, 167.88, 331.32, 164.22, 331.49, 163.22, 333.15, 161.55, 338.15,
161.39, 338.15, 161.39], [337.65, 186.52, 339.14, 190.18, 329.99, 190.68,
330.32, 187.69]], 'area': 190.17595000000034, 'iscrowd': 0, 'image_id': 475546,
'bbox': [329.99, 160.89, 14.98, 29.79], 'category_id': 46, 'id': 667891}
{'segmentation': [[207.16, 292.34, 200.01, 292.91, 196.87, 292.91, 196.58,
282.91, 195.73, 279.48, 195.73, 267.76, 208.3, 267.76, 208.87, 281.48]], 'area':
302.8553000000004, 'iscrowd': 0, 'image_id': 475546, 'bbox': [195.73, 267.76,
13.14, 25.15], 'category_id': 47, 'id': 1503979}
{'segmentation': [[270.32, 294.7, 298.58, 291.9, 302.31, 283.83, 312.56, 284.14,
313.49, 284.14, 313.8, 260.53, 317.53, 260.22, 317.84, 263.02, 318.15, 267.68,
317.22, 274.51, 318.15, 284.45, 328.71, 283.52, 330.88, 276.99, 334.61, 275.44,
335.23, 265.81, 343.0, 266.12, 338.96, 269.54, 341.44, 277.93, 344.24, 285.07,
346.1, 291.9, 352.62, 290.97, 385.24, 315.2, 291.75, 327.62]], 'area':
3372.4544000000005, 'iscrowd': 0, 'image_id': 475546, 'bbox': [270.32, 260.22,
114.92, 67.4], 'category_id': 81, 'id': 1643421}
{'segmentation': [[500.0, 227.99, 484.29, 217.35, 479.53, 208.18, 479.16,
197.91, 479.53, 194.98, 474.76, 188.38, 483.56, 170.04, 493.1, 160.5, 500.0,
158.66]], 'area': 1172.8369000000002, 'iscrowd': 0, 'image_id': 475546, 'bbox':
[474.76, 158.66, 25.24, 69.33], 'category_id': 1, 'id': 1720965}
{'segmentation': [[105.04, 334.09, 277.72, 327.6, 284.76, 325.97, 288.55,
327.05, 292.88, 337.34, 287.46, 340.05, 288.55, 372.53, 108.29, 375.0, 105.58,
338.42]], 'area': 7863.753249999996, 'iscrowd': 0, 'image_id': 475546, 'bbox':
[105.04, 325.97, 187.84, 49.03], 'category_id': 82, 'id': 1983599}
```

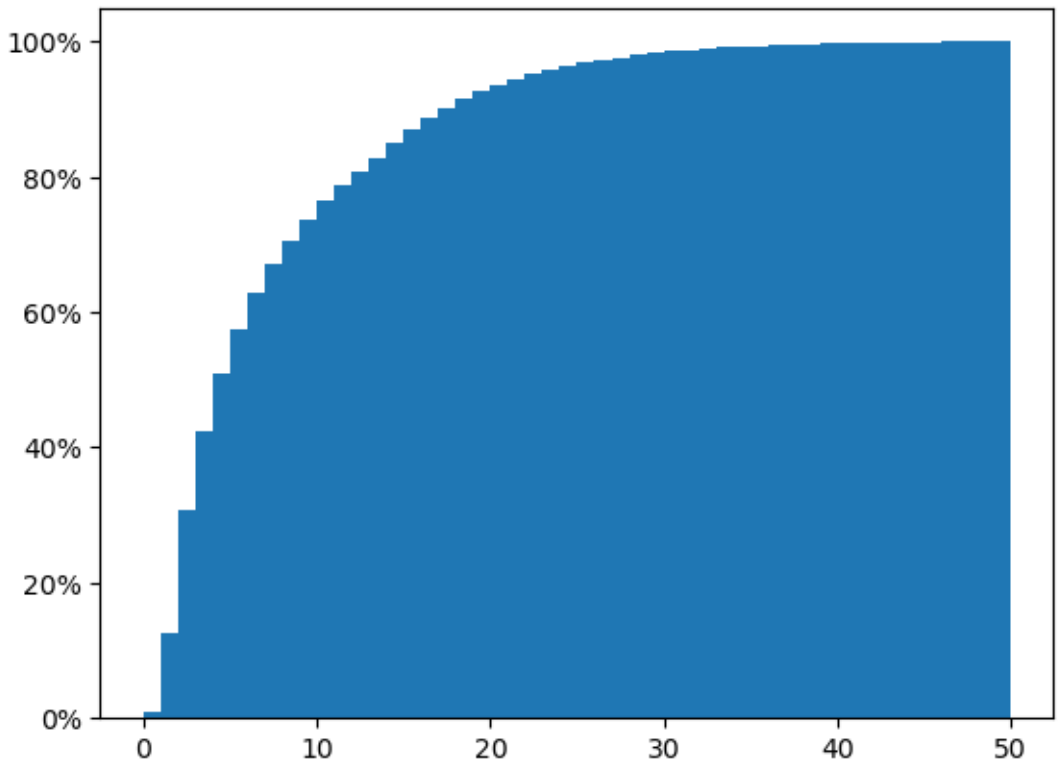
Let's try to have a better understanding of the annotations across the whole dataset. We will count the number of annotations for all images and then we will display a histogram:

[12]:
```
no_anno = []
for img in coco.imgs.values():
    ann = coco.getAnnIds(imgIds=img['id'], iscrowd=None)
    no_anno.append(len(ann))

plt.hist(no_anno, bins=50, range=(0,50), density=True)
plt.gca().yaxis.set_major_formatter(PercentFormatter(xmax=1))
plt.show()
```



Or in a cummulative fashion:

[13]:
```
plt.hist(no_anno, bins=50, range=(0,50), density=True, cumulative=True)
plt.gca().yaxis.set_major_formatter(PercentFormatter(xmax=1))
plt.show()
```



1.3 Items in Images (annotations)

One of the characteristics of the annotated items in a picture is that they tend to cover a significantly smaller portion of the image than the whole area of the picture. For example, for the `an_image_id` we selected earlier, if we extract all the annotations and the associated `area` and compare that with the overall area of the image we will get the following results:

[14]:
```
an_img = coco.imgs[an_image_id]
image_area = an_img['height'] * an_img['width']

for anno in annIds:
    item_categ_id = coco.anns[anno]['category_id']
    item_area = coco.anns[anno]['area']
    item_area_perc = item_area / image_area * 100
    print(f"category: {coco.cats[item_categ_id]['name']}, area: {item_area:.1f}␣
 ↪({item_area_perc:.2f}%)")
```

```
category: person, area: 5945.7 (3.17%)
category: person, area: 3851.8 (2.05%)
category: person, area: 5099.0 (2.72%)
category: person, area: 5632.7 (3.00%)
category: person, area: 24938.9 (13.30%)
```

```
category: wine glass, area: 252.6 (0.13%)
category: wine glass, area: 297.1 (0.16%)
category: wine glass, area: 194.0 (0.10%)
category: wine glass, area: 190.2 (0.10%)
category: cup, area: 302.9 (0.16%)
category: sink, area: 3372.5 (1.80%)
category: person, area: 1172.8 (0.63%)
category: refrigerator, area: 7863.8 (4.19%)
```

As you can see the majority of the items identified in the picture are less than 3% of the whole image, with only one element covering 13.3%. This is clearly visible if we display the corresponding image and the annotations:

[15]:
```
I = io.imread(coco.imgs[an_image_id]['coco_url'])
plt.imshow(I)
plt.axis('off')
coco.showAnns(coco.loadAnns(annIds))
```

We will perform the above calculations for each picture in the set and put the resulting data into a Panda frame for easier manipulation.

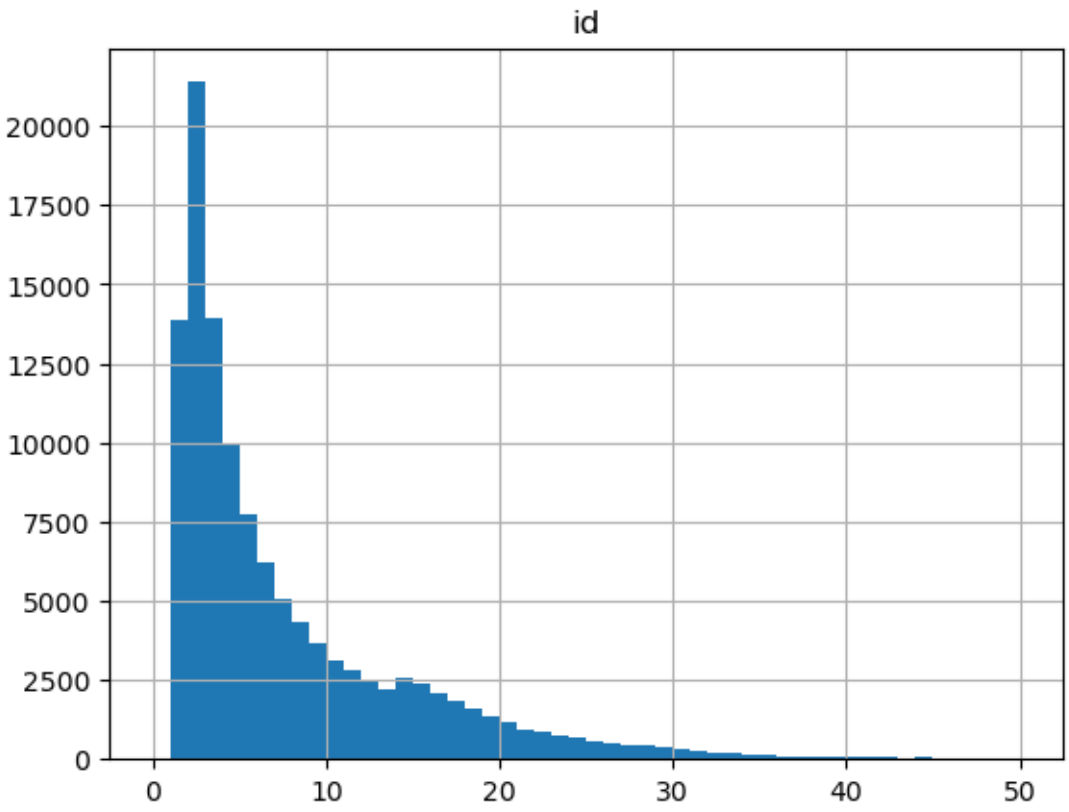
[26]:
```
df = pd.DataFrame(coco.anns.values(), index=coco.anns.keys())
```

[27]:
```
df
```

[27]:
```
                                                segmentation        area  \
156           [[239.97, 260.24, 222.04, 270.49, 199.84, 253…  2765.14865
509           [[247.71, 354.7, 253.49, 346.99, 276.63, 337.3…  1545.42130
603           [[274.58, 405.68, 298.32, 405.68, 302.45, 402…  5607.66135
918           [[296.65, 388.33, 296.65, 388.33, 297.68, 388…     0.00000
1072          [[251.87, 356.13, 260.13, 343.74, 300.39, 335…   800.41325
…                                                         …         …
900100390883  {'counts': [13254, 1, 316, 4, 6, 1, 315, 7, 31…  4227.00000
905300049902  {'counts': [68786, 6, 492, 9, 494, 7, 497, 4, …  6058.00000
904300363764  {'counts': [203528, 6, 420, 8, 418, 10, 416, 1…   737.00000
900100554743  {'counts': [99015, 6, 352, 8, 350, 9, 322, 11,…  6478.00000
900100095999  {'counts': [97214, 1, 425, 4, 422, 6, 420, 9, …  3489.00000

              iscrowd  image_id                              bbox  category_id  \
156                 0    558840   [199.84, 200.46, 77.71, 70.88]           58
509                 0    200365  [234.22, 317.11, 149.39, 38.55]           58
603                 0    200365   [239.48, 347.87, 160.0, 57.81]           58
918                 0    200365      [296.65, 388.33, 1.03, 0.0]           58
1072                0    200365  [251.87, 333.42, 125.94, 22.71]           58
…                 …         …                                 …         …
900100390883        1    390883               [40, 104, 394, 43]            1
905300049902        1     49902             [137, 195, 140, 138]           53
904300363764        1    363764              [476, 200, 153, 85]           43
900100554743        1    554743             [275, 207, 153, 148]            1
900100095999        1     95999              [227, 260, 397, 82]            1

                           id
156                       156
509                       509
603                       603
918                       918
1072                     1072
…                         …
900100390883  900100390883
905300049902  905300049902
904300363764  904300363764
900100554743  900100554743
900100095999  900100095999

[860001 rows x 7 columns]
```

Let's make sure that the numbers match the previously calculated number of annotations per image:

[55]:
```
df[['image_id', 'id']].groupby(by="image_id").count().hist(bins=50,␣
↪range=(0,50))
plt.show()
```



They seem to be the same. Let's add now the image area and the percentage of annotated item in the image:

[42]:
```
df['image_height'] = df.apply(lambda x: coco.imgs[x['image_id']]['height'],␣
↪axis=1)
df['image_width'] = df.apply(lambda x: coco.imgs[x['image_id']]['width'],␣
↪axis=1)
df['image_area'] = df['image_height'] * df['image_width']
df['anno_area_perc'] = df['area'] / df['image_area'] * 100
```

Let's see the results:

[44]:
```
df
```

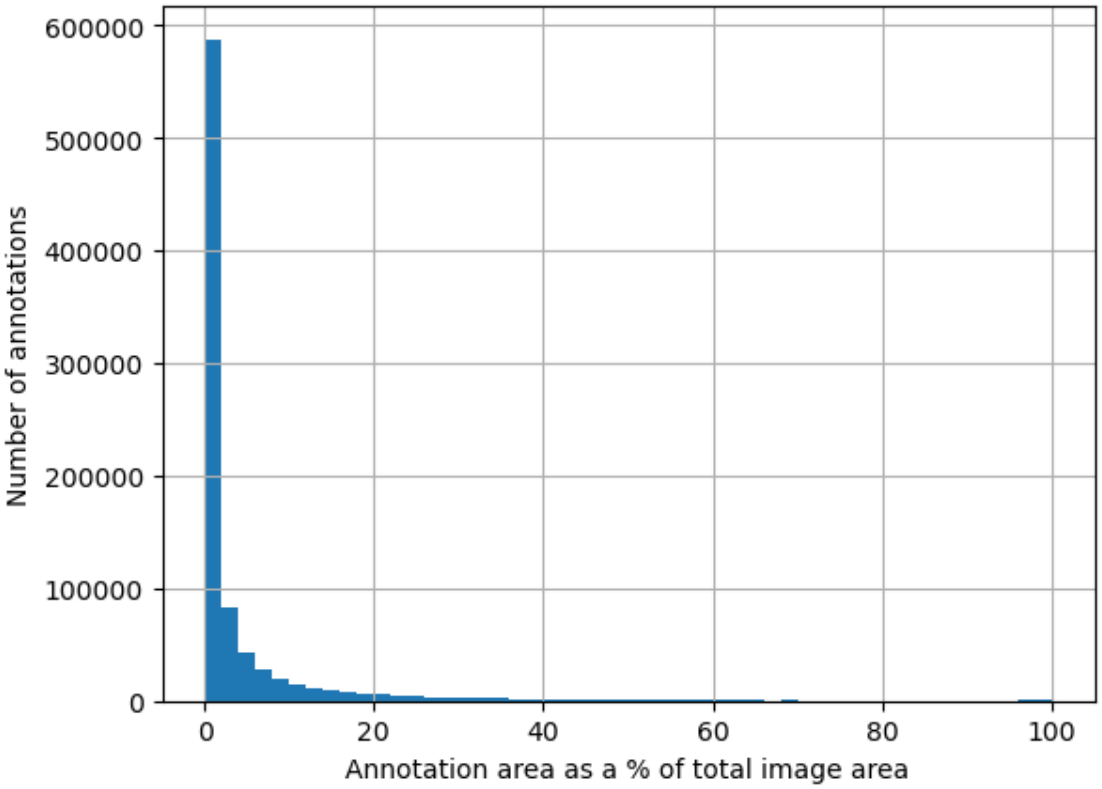
[44]:

	segmentation	area	\
156	[[239.97, 260.24, 222.04, 270.49, 199.84, 253…	2765.14865	
509	[[247.71, 354.7, 253.49, 346.99, 276.63, 337.3…	1545.42130	
603	[[274.58, 405.68, 298.32, 405.68, 302.45, 402…	5607.66135	
918	[[296.65, 388.33, 296.65, 388.33, 297.68, 388…	0.00000	
1072	[[251.87, 356.13, 260.13, 343.74, 300.39, 335…	800.41325	
…	…	…	
900100390883	{'counts': [13254, 1, 316, 4, 6, 1, 315, 7, 31…	4227.00000	

```
905300049902  {'counts': [68786, 6, 492, 9, 494, 7, 497, 4, …  6058.00000
904300363764  {'counts': [203528, 6, 420, 8, 418, 10, 416, 1…   737.00000
900100554743  {'counts': [99015, 6, 352, 8, 350, 9, 322, 11,…  6478.00000
900100095999  {'counts': [97214, 1, 425, 4, 422, 6, 420, 9, …  3489.00000

              iscrowd  image_id                              bbox  category_id  \
156                 0    558840   [199.84, 200.46, 77.71, 70.88]           58
509                 0    200365  [234.22, 317.11, 149.39, 38.55]           58
603                 0    200365   [239.48, 347.87, 160.0, 57.81]           58
918                 0    200365      [296.65, 388.33, 1.03, 0.0]           58
1072                0    200365  [251.87, 333.42, 125.94, 22.71]           58
…                 …        …                                 …          …
900100390883        1    390883               [40, 104, 394, 43]            1
905300049902        1     49902             [137, 195, 140, 138]           53
904300363764        1    363764              [476, 200, 153, 85]           43
900100554743        1    554743             [275, 207, 153, 148]            1
900100095999        1     95999              [227, 260, 397, 82]            1

                          id  image_height  image_width  image_area  \
156                      156           427          640      273280
509                      509           480          640      307200
603                      603           480          640      307200
918                      918           480          640      307200
1072                    1072           480          640      307200
…                        …            …            …          …
900100390883  900100390883           328          500      164000
905300049902  905300049902           500          375      187500
904300363764  904300363764           427          640      273280
900100554743  900100554743           359          640      229760
900100095999  900100095999           427          640      273280

              anno_area_perc
156                 1.011837
509                 0.503067
603                 1.825411
918                 0.000000
1072                0.260551
…                      …
900100390883        2.577439
905300049902        3.230933
904300363764        0.269687
900100554743        2.819464
900100095999        1.276713

[860001 rows x 11 columns]
```

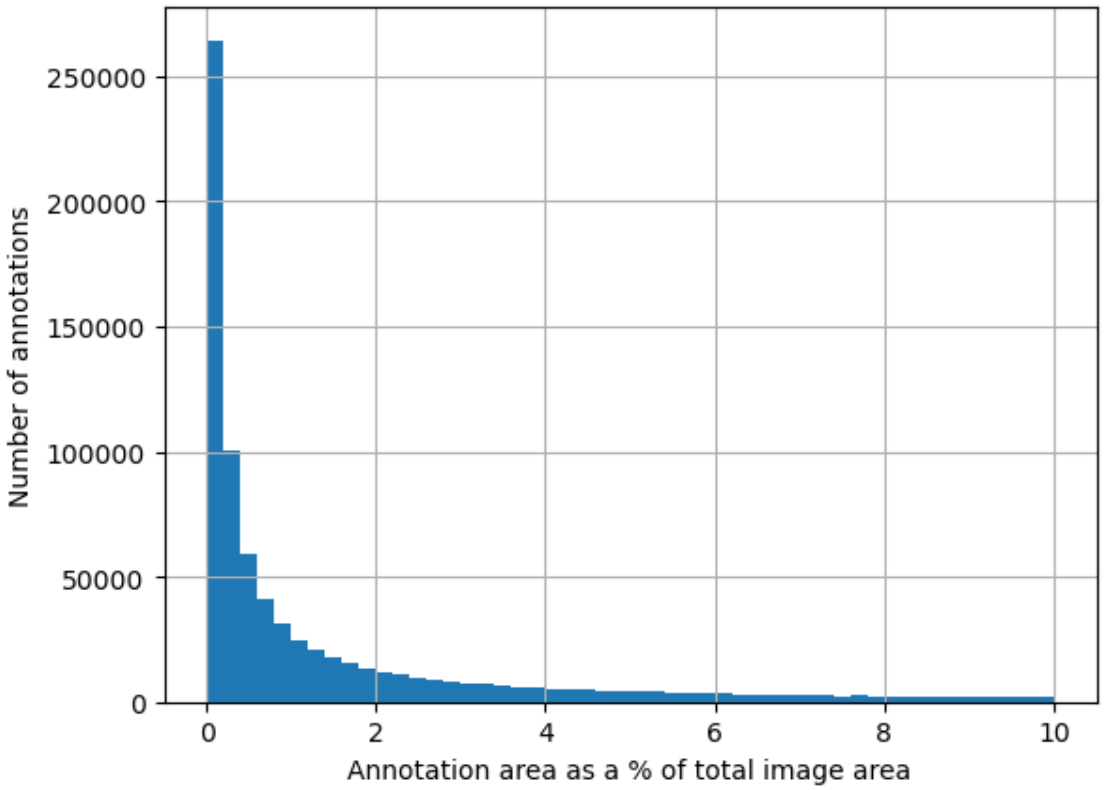
And let's plot a histogram with these percentages:

[53]:
```
df['anno_area_perc'].hist(bins=50, range=(0,100))
plt.gca().set_ylabel('Number of annotations')
plt.gca().set_xlabel('Annotation area as a % of total image area')
plt.show()
```



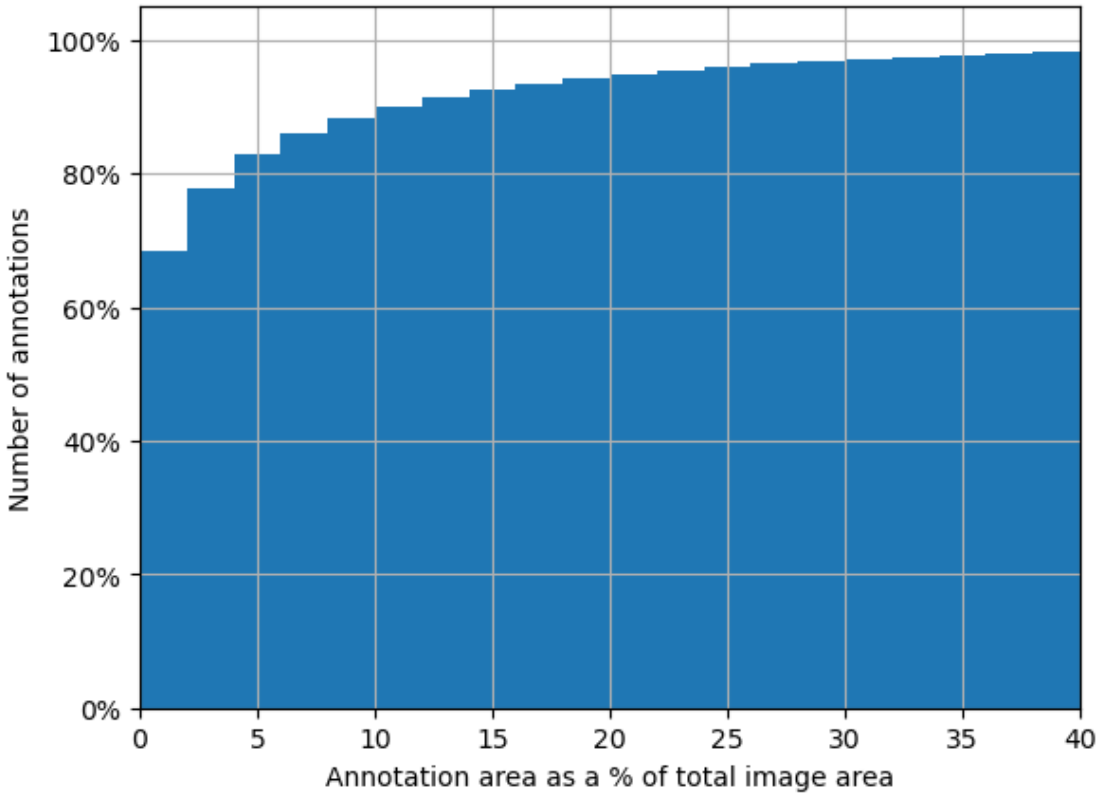
As expected the vast majority is bellow 10%. Let's focus on that part:

[54]:
```
df['anno_area_perc'].hist(bins=50, range=(0,10))
plt.gca().set_ylabel('Number of annotations')
plt.gca().set_xlabel('Annotation area as a % of total image area')
plt.show()
```



A cumulative histogram with % of all annotation will give an even better perpective:

[60]:
```
df['anno_area_perc'].hist(bins=50, range=(0,100), density=True, cumulative=True)
plt.gca().set_ylabel('Number of annotations')
plt.gca().set_xlabel('Annotation area as a % of total image area')
plt.gca().yaxis.set_major_formatter(PercentFormatter(xmax=1))
plt.xlim(left=0, right=40)
plt.show()
```



[16]:
```
dir(coco)
```

[16]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
```

```
'__reduce_ex__',
'__repr__',
'__setattr__',
'__sizeof__',
'__str__',
'__subclasshook__',
'__weakref__',
'annToMask',
'annToRLE',
'anns',
'catToImgs',
'cats',
'createIndex',
'dataset',
'download',
'getAnnIds',
'getCatIds',
'getImgIds',
'imgToAnns',
'imgs',
'info',
'loadAnns',
'loadCats',
'loadImgs',
'loadNumpyAnnotations',
'loadRes',
'showAnns']
```

1.4 Biography

[1]T.-Y. Lin et al., 'Microsoft COCO: Common Objects in Context', arXiv:1405.0312 [cs], Feb. 2015, Accessed: May 05, 2021. [Online]. Available: http://arxiv.org/abs/1405.0312.

[ ]: