case_18.2_downloader

May 30, 2020

1 How to download training images from Google Images

For this case, we scraped photographs directly from Google Image Search, with minimal cleaning afterwards. You can use this method to easily get images on a topic of your choice. In our case, we wanted images from three Colombian celebrities: Isabella Santo Domingo, Sofía Vergara, and Shakira.

We adapted this method from the second lecture of Practical Deep Learning for Coders.

1.1 Step 0: Imports

[0]:
```
from fastai.vision import *
```

1.2 Step 1: Get a list of image URLs

Search Google Images for your keyword, in a window without Ad blockers so that pop-ups aren't blocked. Use `Ctrl+Shift+J` (Windows/Linux) or `Cmd+Opt+J` (Mac) to access a Javascript console and run this code to download the URLs of the images found in the image search:

```
urls = Array.from(document.querySelectorAll('.rg_di .rg_meta')).map(el=>JSON.parse(el.textCont
window.open('data:text/csv;charset=utf-8,' + escape(urls.join('\n')));
```

This will download a CSV file of the URLs in your search results. Save this file with the same name that you'll want as a label for this class, and put it in the folder defined in the variable `path` in the next cell below. If you're running this notebook on AWS, you'll want to create the URL CSVs locally on your laptop, and then upload them to your AWS environment.

1.3 Step 2: Download the pictures

[0]:
```
# Define the classes you'll use.  Use the same naming convention as the CSVs␣
 ↪above.
celebrities = ['sofia',
               'isabella',
               'shakira']

# Define a folder where to put them
```

```
path = Path('photos2')
```

[6]:
```
# Automatically create a new folder for each celebrity,
# and download 200 images from Google Image Search
for name in celebrities:
    print(name)
    folder = name
    file = f'{name}.csv'

    dest = path/folder
    dest.mkdir(parents=True, exist_ok=True)
    download_images(path/file, dest, max_pics=200)
```

```
sofia


      ␣
↪---------------------------------------------------------------------------

       FileNotFoundError                         Traceback (most recent call␣
↪last)

       <ipython-input-6-d448372e0cd7> in <module>()
         6     dest = path/folder
         7     dest.mkdir(parents=True, exist_ok=True)
   ----> 8     download_images(path/file, dest, max_pics=200)


       /usr/local/lib/python3.6/dist-packages/fastai/vision/data.py in␣
↪download_images(urls, dest, max_pics, max_workers, timeout)
       192 def download_images(urls:Collection[str], dest:PathOrStr, max_pics:
↪int=1000, max_workers:int=8, timeout=4):
       193     "Download images listed in text file `urls` to path `dest`, at␣
↪most `max_pics`"
   --> 194     urls = open(urls).read().strip().split("\n")[:max_pics]
       195     dest = Path(dest)
       196     dest.mkdir(exist_ok=True)


       FileNotFoundError: [Errno 2] No such file or directory: 'photos2/sofia.
↪csv'
```

1.4 Step 3: Clean up and organize

Once you get the images, some pictures won't even open, some will be weird formats, and some will just be bad data because they come straight from Google Image Search. For the pictures we

used here, we manually deleted images that:

1. Had the wrong person
2. Also had other people
3. Had too much text
4. Were actually cartoons or drawings
5. Were NSFW (not safe for work)

[0]:
```
# Delete any pictures that can't be opened
for c in celebrities:
    print(c)
    verify_images(path/c, delete=True, max_size=500)
```

```
sofia_vergara


      ␣
↪---------------------------------------------------------------------------

       NameError                                 Traceback (most recent call␣
↪last)

       <ipython-input-6-b12af0950636> in <module>()
         2 for c in celebrities:
         3     print(c)
   ----> 4     verify_images(path/c, delete=True, max_size=500)


       NameError: name 'verify_images' is not defined
```

Most importantly, we stored the images in a folder structure divided first by train/test/valid set, and then by the labels we wish the DataBunch to use. This is the standard structure for "Imagenet-style" datasets (note that the test folder is optional):

```
path\
  train\
    clas1\
    clas2\
    ...
  valid\
    clas1\
    clas2\
    ...
  test\
```

We downloaded ~200 pictures per celebrity, and in the end used 70 for training and 30 for validation. Image names were generated automatically during downloading, and are not important. Our pictures come in several sizes, and the celebrities are by no means photographed in the same

way in each. Sometimes you see their face clearly, other times you cannot.