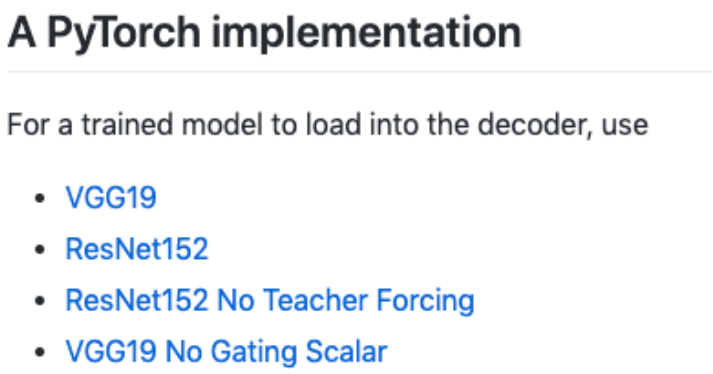
PROJECT 3 INSTRUCTIONS
Show and Tell

1. Environment Setup
   a. Install essential libraries
      Python3 and upper, Pytorch, torchvision, nltk, numpy, tensorboardX, scikit-image, CUDA

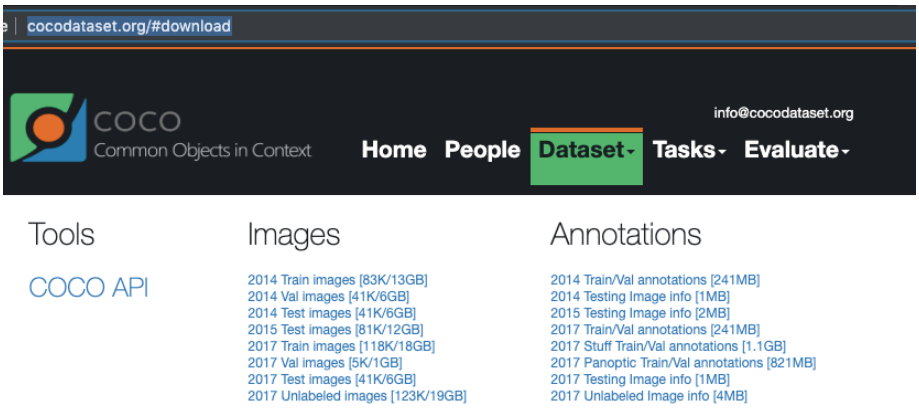
      Go to Pytorch official website and install the library proper for your environment:
      https://pytorch.org/
      The same thing with CUDA, be sure to pair with your NAVIDA driver version.

   b. Use virtual environment if necessary, simply one should be mini conda, Docker and Singularity may be used as well. For first time user, go on the websites to download pre-built img or container with essentials libraries you need. For professionals, you may build your own.

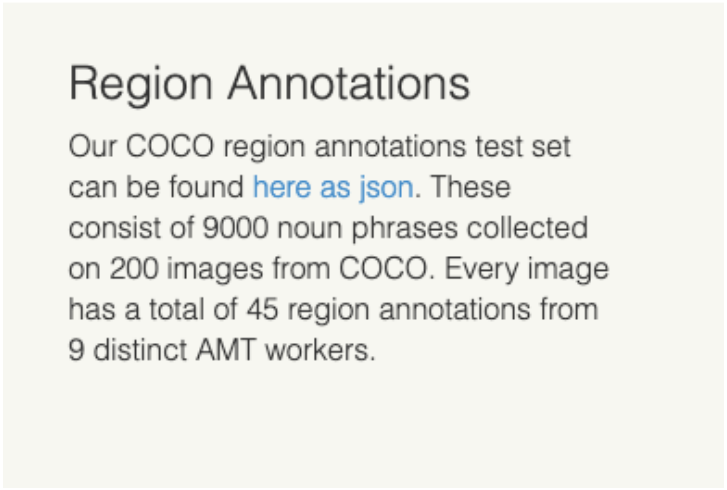
2. Dataset and Json files Preparation (Command Line or interactive)
   a. git clone https://github.com/AaronCCWong/Show-Attend-and-Tell.git
   b. Download the pre-trained models in the website above, and there are four available there. Put them in the model folder.



   c. Download data from the http://cocodataset.org/#download, and put them in the corresponding folders as mentioned on github.

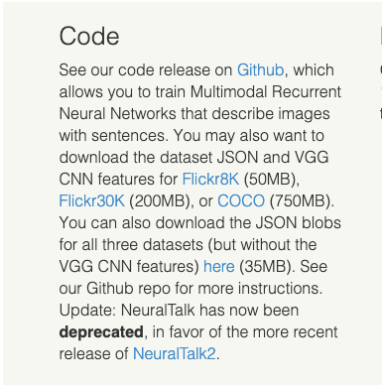


d. Download COCO dataset split JSON file from the link https://cs.stanford.edu/people/karpathy/deepimagesent/, and rename it as dataset.json in the corresponding folders as mentioned on github.



e. Run preprocessing to create JSON files by command line: Python generate_json_data.py

f. Download other JSON files if you would like to try training the dataset by yourself. (Don't have to do this if you just want to use the pre-trained to have a prediction.)



3. Run the prediction
   Follow the instruction on the github or you may refer to my command line for reference as well.

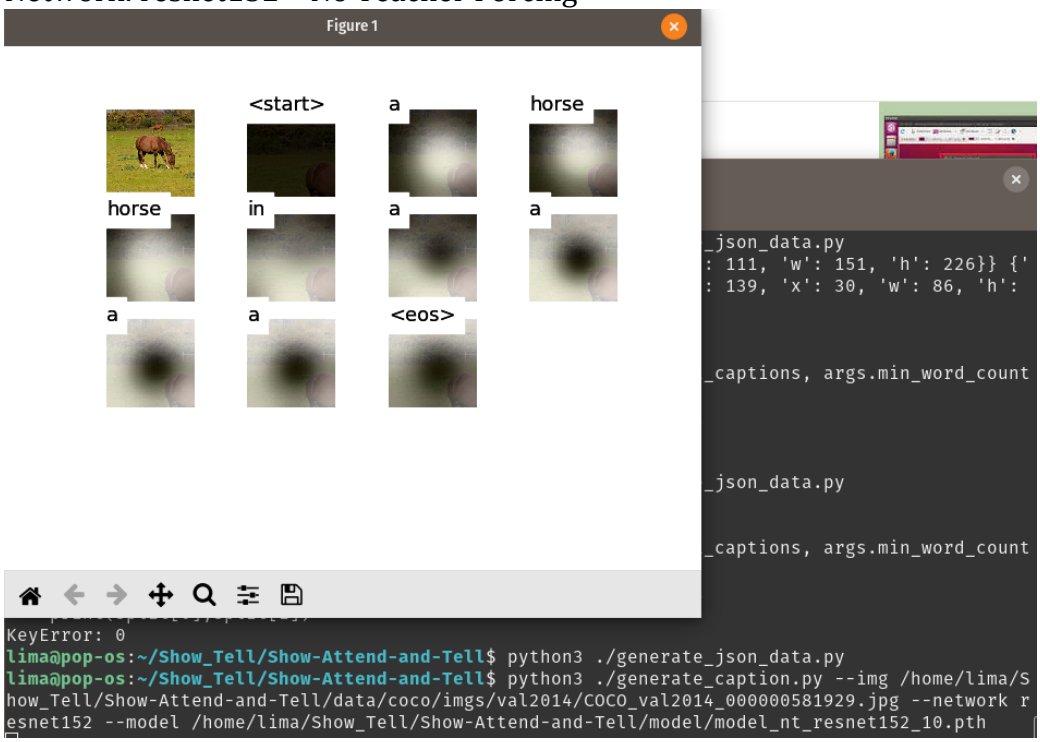
Reference Results:

The implement was performed on a Linux Ubuntu 18.04 LTS system.

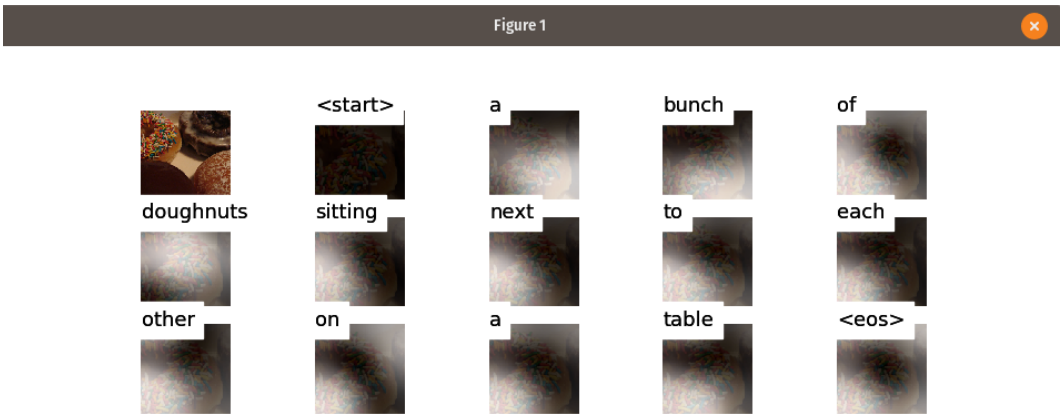
```
lima@pop-os:~/Show_Tell$ python3 --version
Python 3.6.8
lima@pop-os:~/Show_Tell$ python3 ./lib_verification.py
torch: 1.3.0
torchvision 0.4.1
numpy 1.17.2
tensorboardX 1.9
nltk 3.4.5
```

scikit-image 3.4.5

Network: resnet152 – No Teacher Forcing



Network: resnet152




```
lima@pop-os:~/Show_Tell/Show-Attend-and-Tell$ python3 ./generate_caption.py --img /home/lima/S
how_Tell/Show-Attend-and-Tell/data/coco/imgs/val2014/COCO_val2014_000000581913.jpg --network r
esnet152 --model /home/lima/Show_Tell/Show-Attend-and-Tell/model/model_resnet152_10.pth
```

Network: vgg19


```
lima@pop-os:~/Show_Tell/Show-Attend-and-Tell$ python3 ./generate_caption.py --img /home/lima/S
how_Tell/Show-Attend-and-Tell/data/coco/imgs/val2014/COCO_val2014_000000581899.jpg --network v
gg19 --model /home/lima/Show_Tell/Show-Attend-and-Tell/model/model_10.pth
```

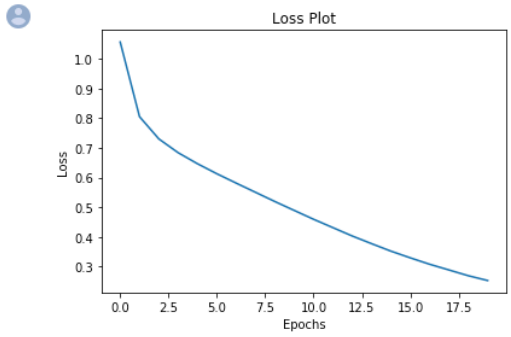
TensorFlow Implement:
https://www.tensorflow.org/tutorials/text/image_captioning

Click Run in Google Colab directly; don't copy it into your own colab, which will limit the capacity of the RAM and Disk.

Training Procedure:

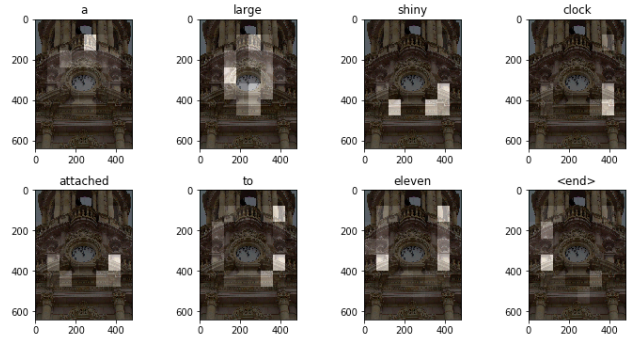
```
[30] Epoch 17 Batch 200 Loss 0.2982
     Epoch 17 Batch 300 Loss 0.3104
     Epoch 17 Loss 0.307570
     Time taken for 1 epoch 123.64052319526672 sec

     Epoch 18 Batch 0 Loss 0.3314
     Epoch 18 Batch 100 Loss 0.3409
     Epoch 18 Batch 200 Loss 0.2828
     Epoch 18 Batch 300 Loss 0.2877
     Epoch 18 Loss 0.288281
     Time taken for 1 epoch 118.91653037071228 sec

     Epoch 19 Batch 0 Loss 0.2945
     Epoch 19 Batch 100 Loss 0.2900
     Epoch 19 Batch 200 Loss 0.2545
     Epoch 19 Batch 300 Loss 0.2624
     Epoch 19 Loss 0.268873
     Time taken for 1 epoch 132.19374108314514 sec

     Epoch 20 Batch 0 Loss 0.2902
     Epoch 20 Batch 100 Loss 0.2276
     Epoch 20 Batch 200 Loss 0.2581
     Epoch 20 Batch 300 Loss 0.2787
     Epoch 20 Loss 0.252668
     Time taken for 1 epoch 120.01273250579834 sec
```

```
[31] plt.plot(loss_plot)
     plt.xlabel('Epochs')
     plt.ylabel('Loss')
     plt.title('Loss Plot')
     plt.show()
```



Caption Results:

```
[34]
Real Caption: <start> a clock that is in a very nice building <end>
Prediction Caption: a large shiny clock attached to eleven <end>
```



```
image_url = 'https://tensorflow.org/images/surf.jpg'
image_extension = image_url[-4:]
image_path = tf.keras.utils.get_file('image'+image_extension,
                                     origin=image_url)

result, attention_plot = evaluate(image_path)
print ('Prediction Caption:', ' '.join(result))
plot_attention(image_path, result, attention_plot)
# opening the image
Image.open(image_path)
```

```
Downloading data from https://tensorflow.org/images/surf.jpg
65536/64400 [==============================] - 0s 0us/step
Prediction Caption: a man riding a surfboard riding a surfboard <end>
```