

## CS5542 – Bigdata Analytics and Applications

### Lab report – 2

#### Submitted by:

Saitejaswi Koppuravuri – 13

Team – 5

#### Objectives:

*The objectives of the Lab assignment 2:*

- Generate captions for your own dataset using the Show and Tell model.
- Generate 4 captions for each image. (Beam Search k=4)
- Report your accuracy in BLEU, CIDER, METEOR and ROGUE measures.

#### Technologies:

Pycharm – IDE for executing the python files

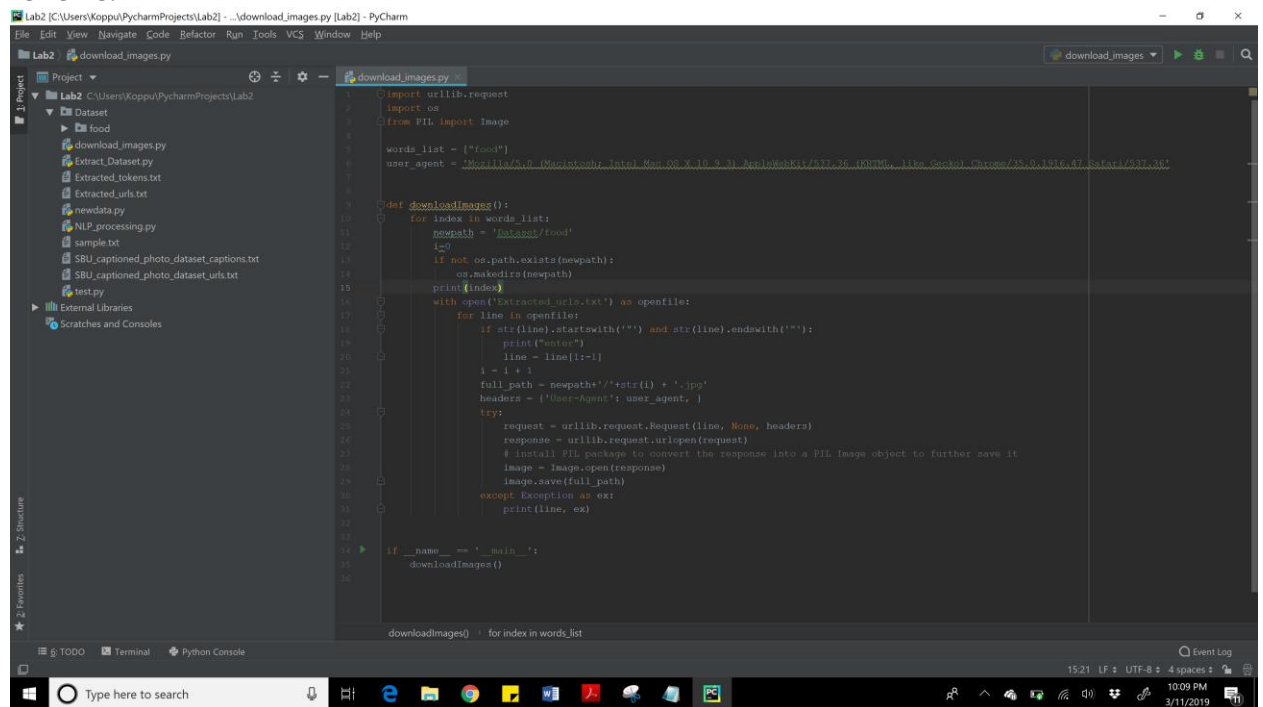
#### Packages used:

- nltk
- opencv-python
- numpy
- matplotlib
- Tensorflow
- PyRouge
- BLEU score
- Show and tell model
- PIL

#### Extraction of Images from the URL's:

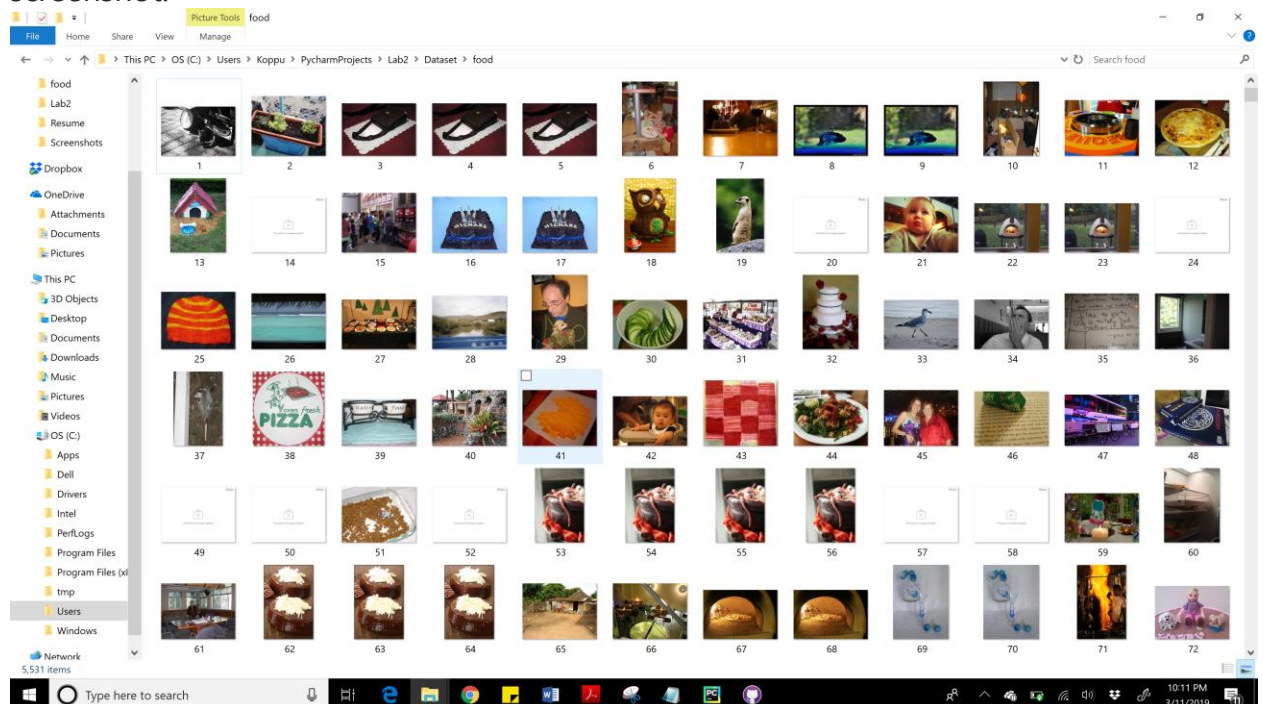
- As our dataset choosen is SBU, we have extracted the images from the URL's. The screenshot is as

follows:



```
1 import urllib.request
2 import os
3 from PIL import Image
4
5 words_list = ["food"]
6 user_agent = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1526.47 Safari/537.36"
7
8 def downloadImages():
9     for index in words_list:
10         newpath = 'Dataset/' + index
11         if not os.path.exists(newpath):
12             os.makedirs(newpath)
13         print(index)
14         with open('Extracted_urls.txt') as openfile:
15             for line in openfile:
16                 if str(line).startswith('http') and str(line).endswith('.jpg'):
17                     print("enter")
18                     line = line[:-1]
19                     i = i + 1
20                     full_path = newpath + '/' + str(i) + '.jpg'
21                     headers = {'User-Agent': user_agent, }
22                     try:
23                         request = urllib.request.Request(line, None, headers)
24                         response = urllib.request.urlopen(request)
25                         # install PIL package to convert the response into a PIL Image object to further save it
26                         image = Image.open(response)
27                         image.save(full_path)
28                     except Exception as ex:
29                         print(line, ex)
30
31 if __name__ == '__main__':
32     downloadImages()
```

- The dataset downloaded screenshot:



Goals achieved:

- Captions for the extracted images have been generated using the show and tell model.

```

12 from medium_show_and_tell_caption_generator.caption_generator import CaptionGenerator
13 from medium_show_and_tell_caption_generator.model import ShowAndTellModel
14 from medium_show_and_tell_caption_generator.vocabulary import Vocabulary
15
16 FLAGS = tf.flags.FLAGS
17
18 f.flags.DEFINE_string("model_path", "C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/model/show-and-tell.py",
19 f.flags.DEFINE_string("vocab_file", "C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/word_counts.txt",
20 f.flags.DEFINE_string("input_files", "C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/inputs",
21 "file pattern or comma-separated list of file patterns"
22 "of image files.")
23
24 logging.basicConfig(level=logging.INFO)
25 logger = logging.getLogger(__name__)
26
27 def main():
28     model = ShowAndTellModel(FLAGS.model_path)
29     vocab = Vocabulary(FLAGS.vocab_file)
30     filenames = _load_filenames()
31
32     generator = CaptionGenerator(model, vocab)
33     with open("C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/inputs/predict.txt", "a") as f:
34         for filename in filenames:
35             with tf.gfile.GFile(filename, "rb") as f:
36                 image = f.read()
37                 captions = generator.beam_search(image)
38                 print("Captions for image %s" % os.path.basename(filename))
39                 for i, caption in enumerate(captions):
40                     # ignores begin and end tokens <S> and </S>
41                     sentence = [vocab.id_to_token(w) for w in caption.sentence[1:-1]]
42                     sentence = " ".join(sentence)
43                     if i == 0:
44                         f.write("%s\n" % sentence)
45                         # print("this is---", sentence)
46                     print("%d %s (p=%f)" % (i, sentence, math.exp(caption.logprob)))
47
48     main()

```

- For every image four captions are generated by changing the Beam size value to 4.

```

12 from medium_show_and_tell_caption_generator.caption_generator import CaptionGenerator
13 from medium_show_and_tell_caption_generator.model import ShowAndTellModel
14 from medium_show_and_tell_caption_generator.vocabulary import Vocabulary
15
16 FLAGS = tf.flags.FLAGS
17
18 f.flags.DEFINE_string("model_path", "C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/model/show-and-tell.py",
19 f.flags.DEFINE_string("vocab_file", "C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/word_counts.txt",
20 f.flags.DEFINE_string("input_files", "C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/inputs",
21 "file pattern or comma-separated list of file patterns"
22 "of image files.")
23
24 logging.basicConfig(level=logging.INFO)
25 logger = logging.getLogger(__name__)
26
27 def main():
28     model = ShowAndTellModel(FLAGS.model_path)
29     vocab = Vocabulary(FLAGS.vocab_file)
30     filenames = _load_filenames()
31
32     generator = CaptionGenerator(model, vocab)
33     with open("C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/inputs/predict.txt", "a") as f:
34         for filename in filenames:
35             with tf.gfile.GFile(filename, "rb") as f:
36                 image = f.read()
37                 captions = generator.beam_search(image)
38                 print("Captions for image %s" % os.path.basename(filename))
39                 for i, caption in enumerate(captions):
40                     # ignores begin and end tokens <S> and </S>
41                     sentence = [vocab.id_to_token(w) for w in caption.sentence[1:-1]]
42                     sentence = " ".join(sentence)
43                     if i == 0:
44                         f.write("%s\n" % sentence)
45                         # print("this is---", sentence)
46                     print("%d %s (p=%f)" % (i, sentence, math.exp(caption.logprob)))
47
48     main()

```

Run: inference\_change1

```

0) a white plate topped with meat and vegetables . (p=0.002101)
1) a white plate topped with meat and broccoli . (p=0.001419)
2) a white plate topped with meat and vegies . (p=0.001326)
3) a plate of food with broccoli and meat . (p=0.000738)
Captions for image trading_floor.jpg:
0) a group of people sitting at tables in a room . (p=0.000307)
1) a group of people sitting at tables with laptops . (p=0.000216)
2) a group of people sitting around a table with laptops . (p=0.000140)
3) a group of people sitting at a table with laptops . (p=0.000069)

```

- Calculation of BLEU score and ROGUE measures:

```

def accuracy():
    r = Rouge()
    list = []
    with open("C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/medium_show_and_tell_caption_generator/predict.txt") as f2:
        for line in f2:
            list.append(line)
    with open("C:/Users/Koppu/Desktop/UMKC/Big data applications/Tutorial source codes/Tutorial 6 Source Code/medium-show-and-tell-caption-generator-master/medium_show_and_tell_caption_generator/true.txt") as f1:
        for line1 in f1:
            i = 0
            y_true = list[i]
            i += 1
            y_pred = f1.readline()
            BLEUScore = sentence_bleu(word_tokenize(y_true), word_tokenize(y_pred), weights=(1, 0, 0, 0))
            [precision, recall, f_score] = r.rouge_1([y_true], [y_pred])
            print("Precision is : " + str(precision) + "\nRecall is : " + str(recall) + "\nF Score is : " + str(f_score))
            print(BLEUScore)

if __name__ == "__main__":
    accuracy()

```

```

Run: accuracy
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
warnings.warn(msg)
Precision is :0.8297872340425532
Recall is :1.0
F Score is :0.9069776946187155
0.25
Precision is :0.425531914893617
Recall is :0.4444444444444444
F Score is :0.4347835587192874
0.2222222222222222
Process finished with exit code 0

```