# Submitted by

**Harish Tata**

**Class ID: 25**

**Team 4**

# Objectives:

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

# Technologies used:

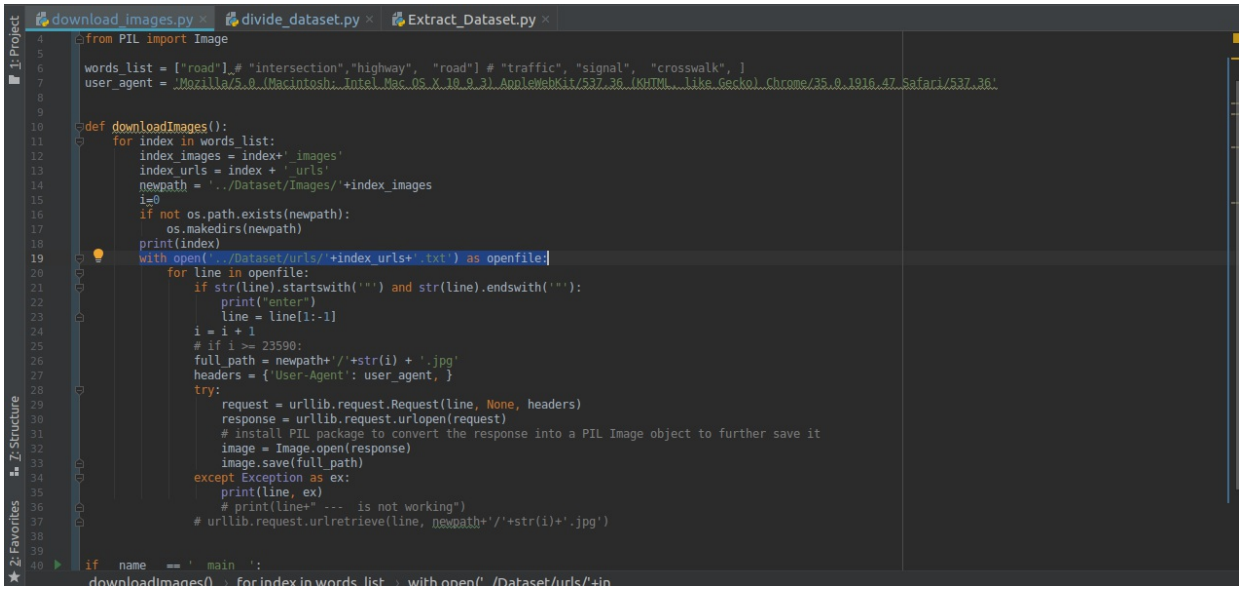
1. TensorFlow
2. Pandas

# Tools Used:

Pycharm

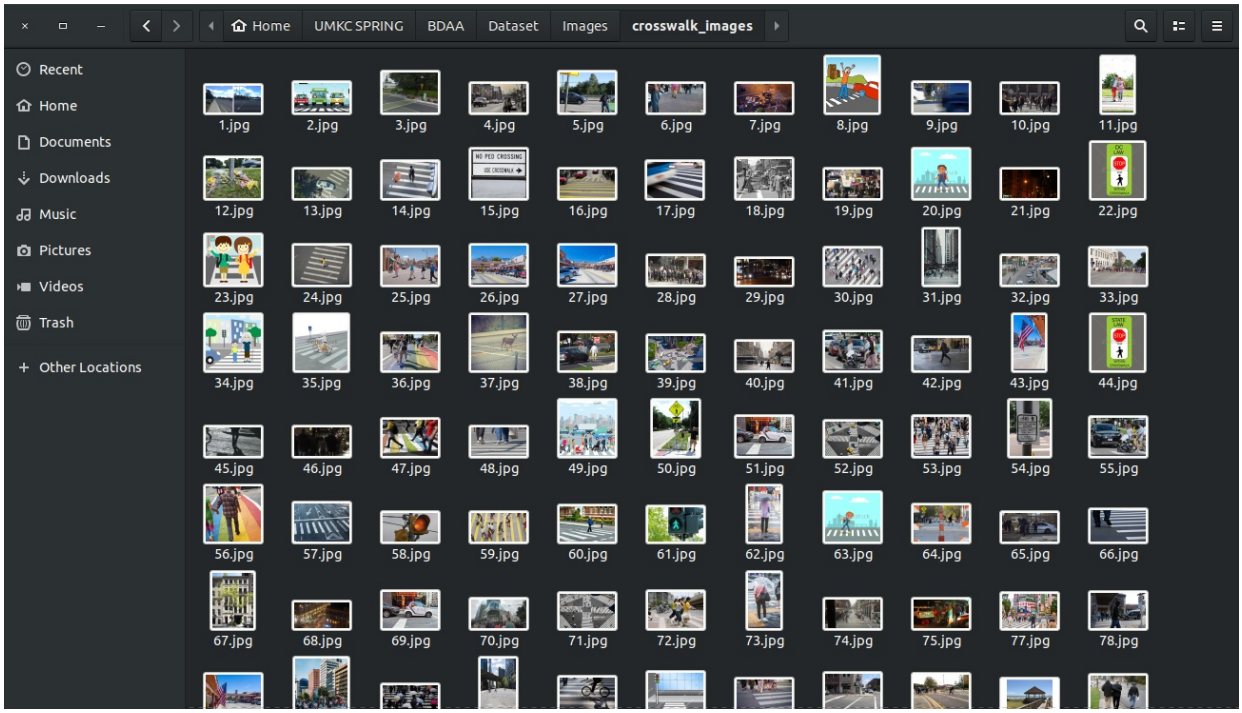
# Dataset

- Downloaded the “Google Conceptual Captions” data set. Filtered the data set with the keywords used for our project theme

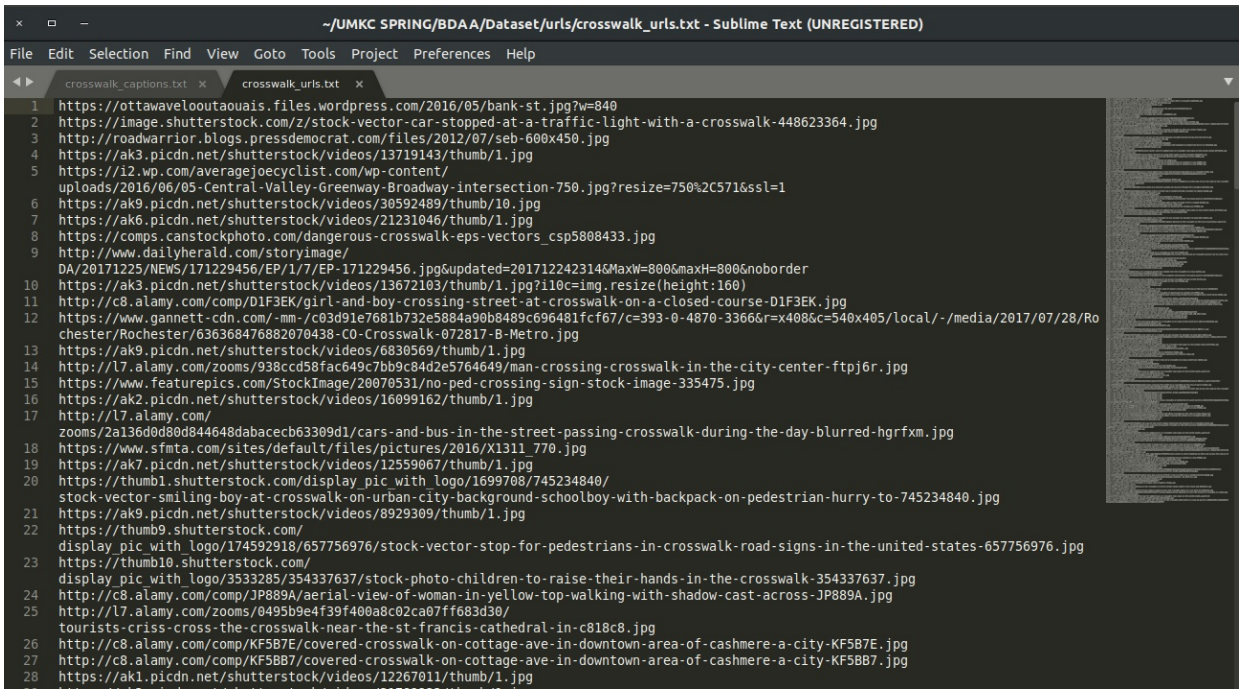



# Output

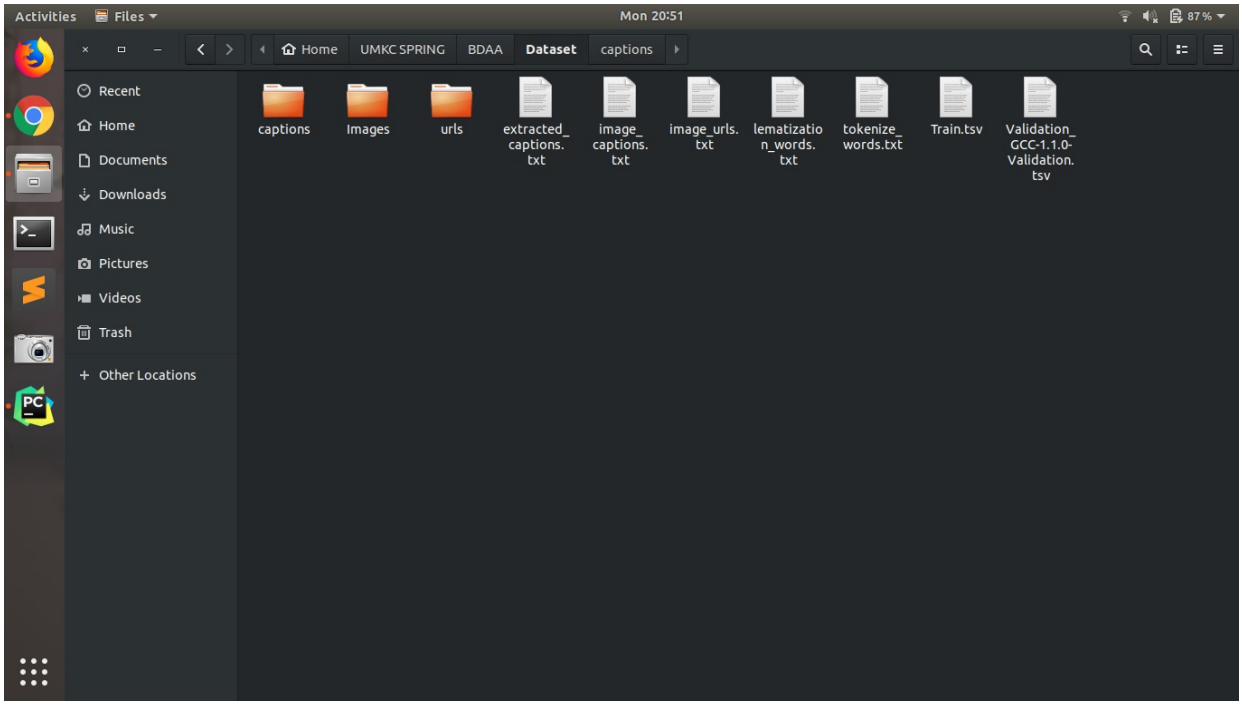
- This is output of dowloaded images data set



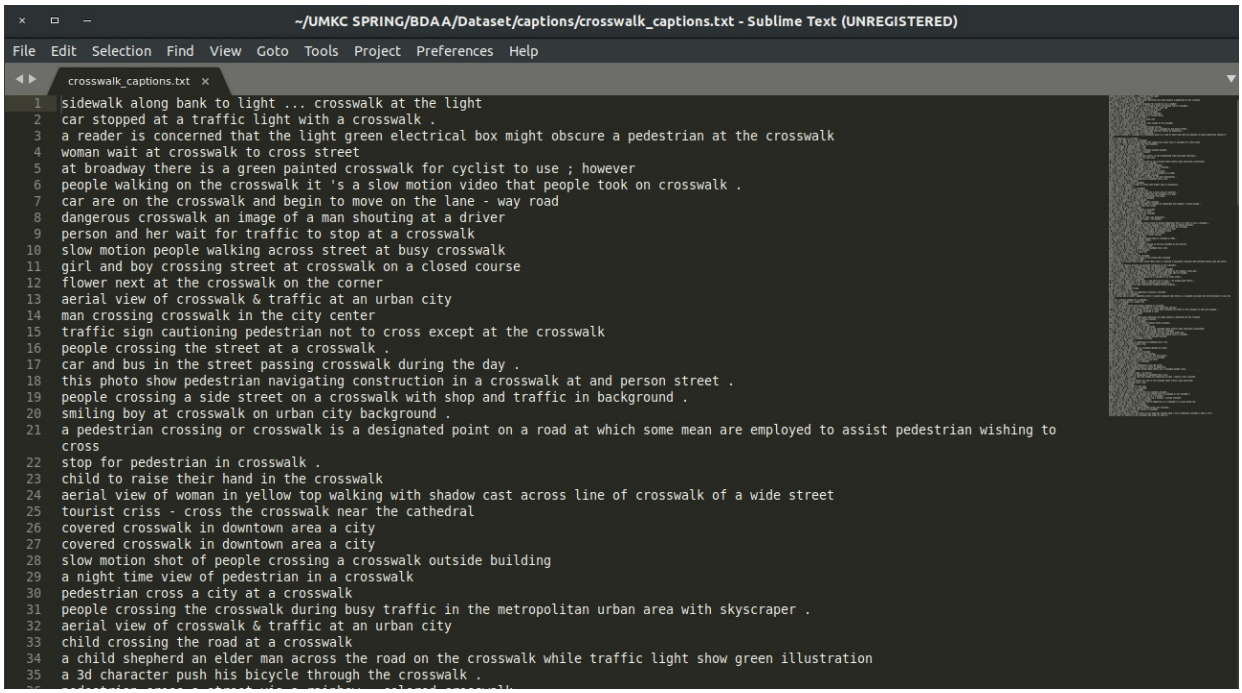

- This is output i.e urls text file of Crosswalk keyword




- This is the data set folder consists of images, url text file and caption text file of filtered keywords




- These are the captions of Crosswalk keyword



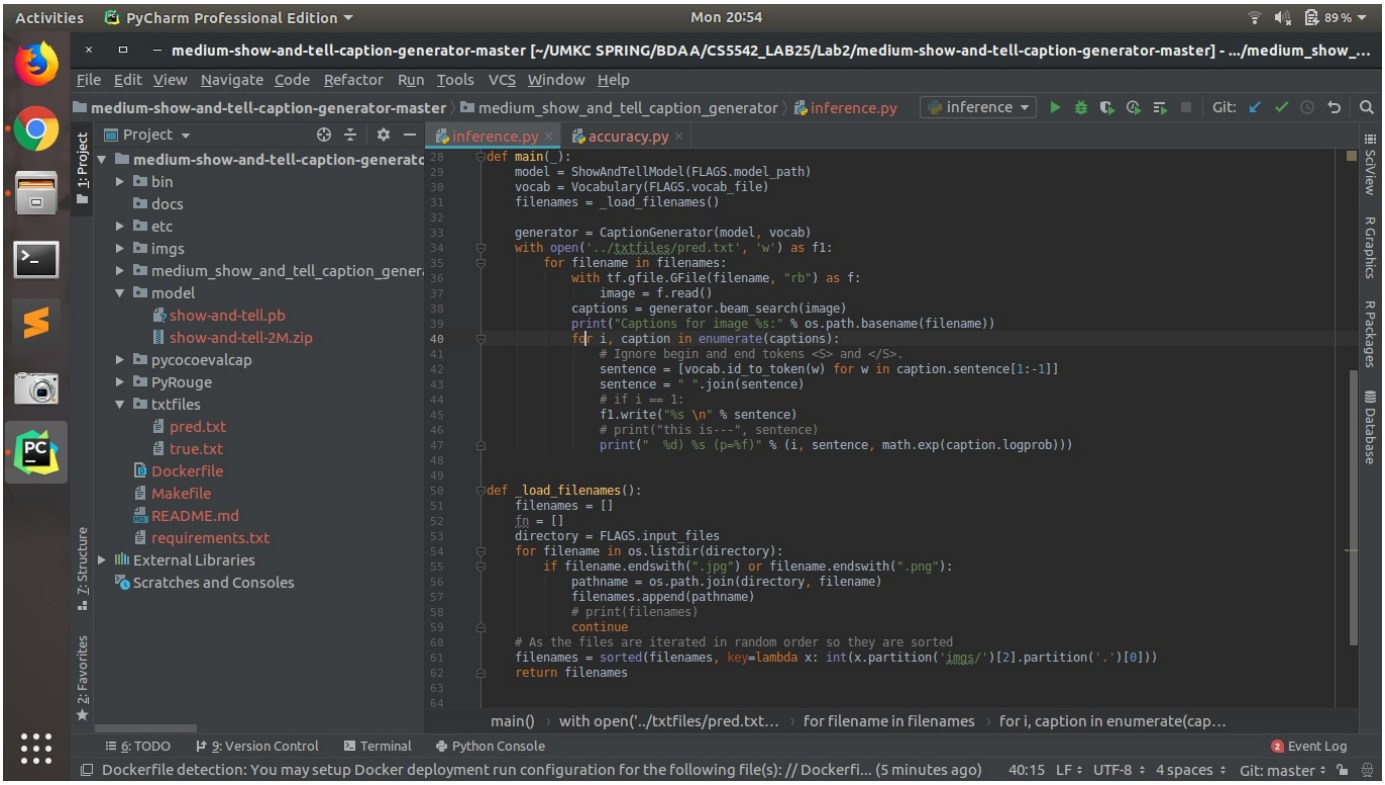

# 1.Generate captions for your own dataset using the show and Tellmodel.

# Code Snippet

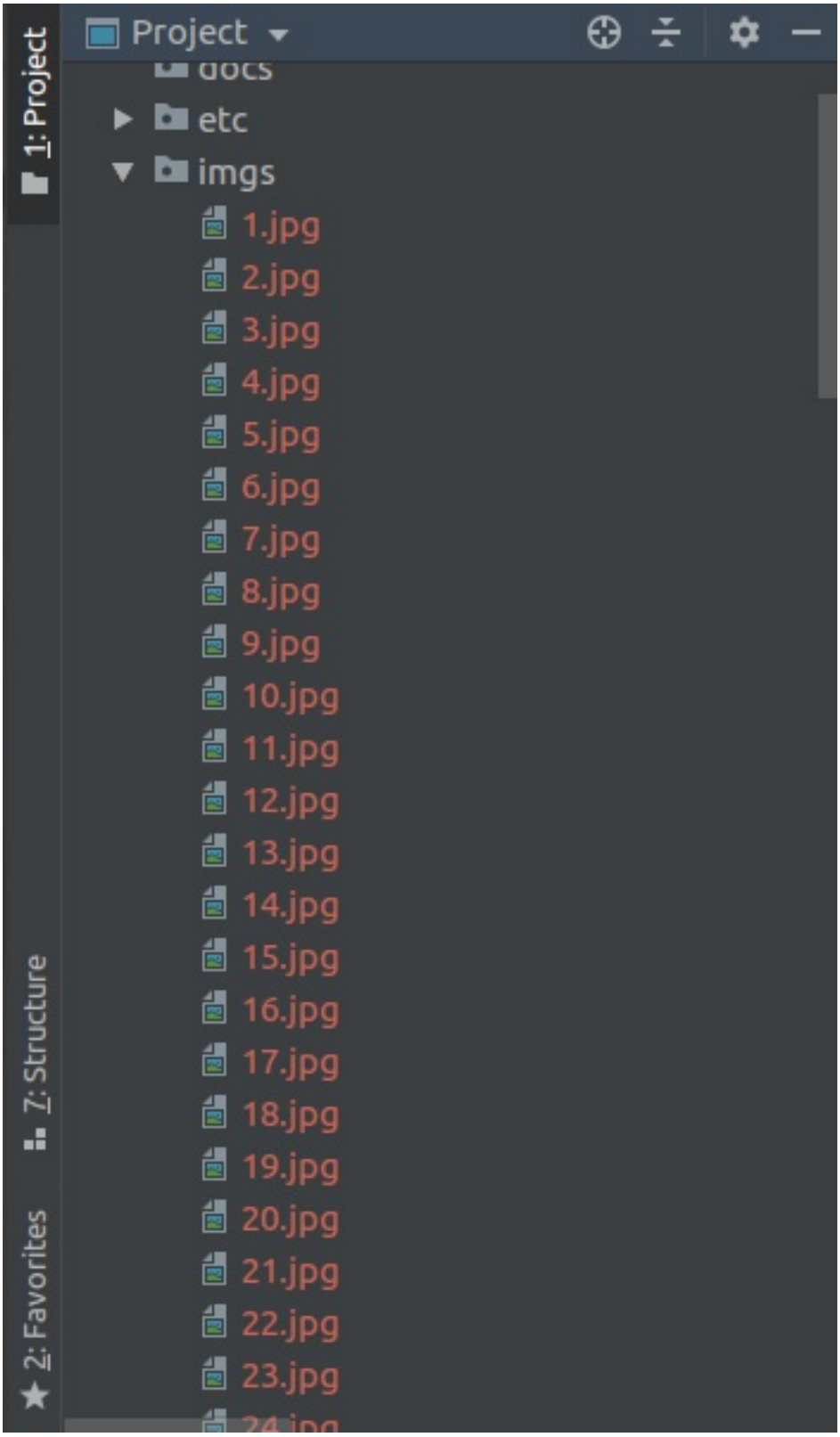
### Steps

1. First, Download the images from image URL’s and make ready the dataset
2. Change the path names which are initialized for the flags
3. Now iterate through the images directory and pass all images to the main method(code is in _load_filenames)
4. In the caption_generator.py change the beam size to 4 i.e. for generating 4 captions.
5. Now give the image to the model it will generate the 4 captions

- Code which iterates the image dataset




- Image dataset is as follows




## 2.Generate 4 captions for each image. (Beam Search k=4)

# Code Snippet

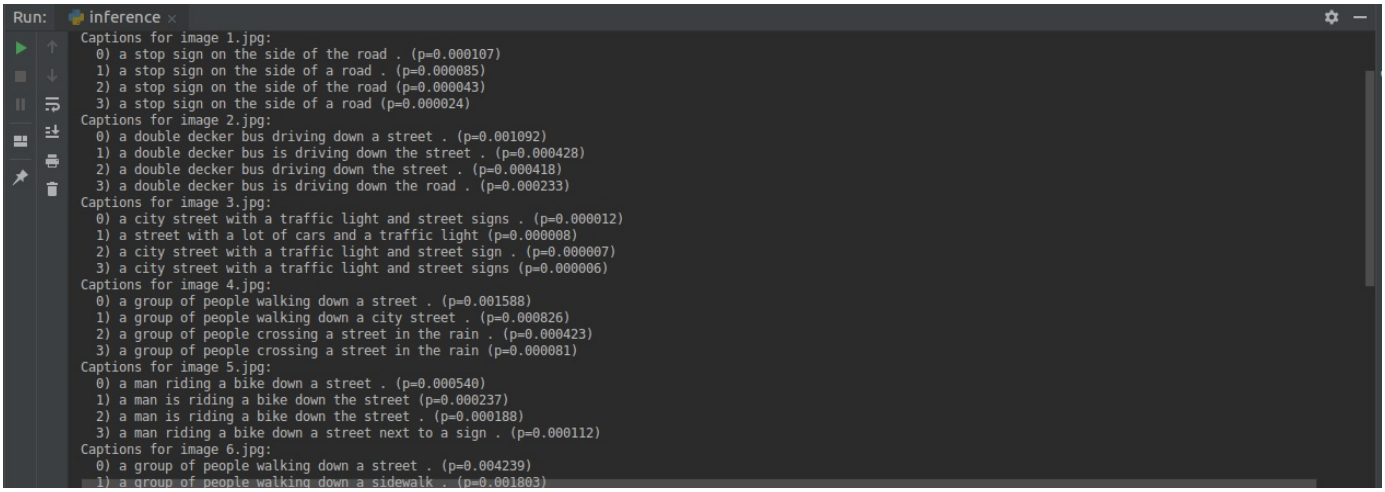
- Code for genrating captions

```python
from medium_show_and_tell_caption_generator.model import ShowAndTellModel
from medium_show_and_tell_caption_generator.vocabulary import Vocabulary

FLAGS = tf.flags.FLAGS

tf.flags.DEFINE_string("model_path", "../model/show-and-tell.pb", "Model graph def path")
tf.flags.DEFINE_string("vocab_file", "../etc/word_counts.txt", "Text file containing the vocabulary.")
tf.flags.DEFINE_string("input_files", "../imgs/",
                       "File pattern or comma-separated list of file patterns "
                       "of image files.")

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger(__name__)


def main(_):
    model = ShowAndTellModel(FLAGS.model_path)
    vocab = Vocabulary(FLAGS.vocab_file)
    filenames = _load_filenames()

    generator = CaptionGenerator(model, vocab)
    with open('../txtfiles/pred.txt', 'w') as f1:
        for filename in filenames:
            with tf.gfile.GFile(filename, "rb") as f:
                image = f.read()
            captions = generator.beam_search(image)
            print("Captions for image %s:" % os.path.basename(filename))
            for i, caption in enumerate(captions):
                # Ignore begin and end tokens <S> and </S>.
                sentence = [vocab.id_to_token(w) for w in caption.sentence[1:-1]]
                sentence = " ".join(sentence)
                # if i == 1:
                f1.write("%s \n" % sentence)
                # print("this is---", sentence)
                print("  %d) %s (p=%f)" % (i, sentence, math.exp(caption.logprob)))

_load_filenames()
```

- Set Beam size as 4

```python
    def __init__(self,
                 model,
                 vocab,
                 beam_size=4,
                 max_caption_length=20,
                 length_normalization_factor=0.0):

        self.vocab = vocab
        self.model = model
```

# Output




# 3.Report your accuracy in BLEU, CIDER ROGUE measures.

**Steps**

1. When you run inference.py it will generate the pred.txt file which has all the predicted for captions for the dataset
2. Create a txt file which has true dataset captions
3. Iterate both files and make a list of 4 captions and one true caption give this data to senctence_bleu to get BLEU SCORE
4. Send the single caption of pred and true to the rouge you will get presion, recall and f_score(code is taken from git hub url - https://github.com/pcyin/PyRouge)
5. Make the dict of pred caption and true captions and set to comput_score to get the CIDEr score
6. (code is taken from github url - https://github.com/salaniz/pycocoevalcap)

# Code Snippet

```python
    ]
    final_scores = {}
    with open("../txtfiles/pred.txt") as f2,  open("../txtfiles/true.txt") as f1:
        k = 0
        j = 0
        for line in f2:
            list.append(line)
        for line1 in f1:
            y_true = []
            y_true_dict = {}
            y_pred_dict = {}
            for i in range(3):
                y_true.append(word_tokenize(list[k]))
                k = k + 1
            y_true_line = list[k-4]
            y_true_dict[j] = str(list[k])
            # print(y_true)
            y_pred = word_tokenize(line1)
            y_pred_line = line1
            y_pred_dict[j] = str(line1)
            j = j+1
            for scorer, method in scorers:
                # print(type(y_pred_dict))
                score, scores = scorer.compute_score(y_true_dict, y_true_dict)
                if type(score) == list:
                    for m, s in zip(method, score):
                        final_scores[m] = s
                else:
                    print("%s: %0.3f" % (method, score))
                    final_scores[method] = score
            BLEUscore = sentence_bleu(y_true, y_pred, weights=(1, 0, 0, 0))
            [precision, recall, f_score] = r.rouge_l([y_true_line], [y_pred_line])
            print("Rouge Score is :", f_score)
            print("BLEUscore is: ", BLEUscore)
            # print(final_scores)
```

# BLEU

- BLEU (bilingual evaluation understudy)
- It is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another
- BLEU’s output is always a number between 0 and 1.
- This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts.

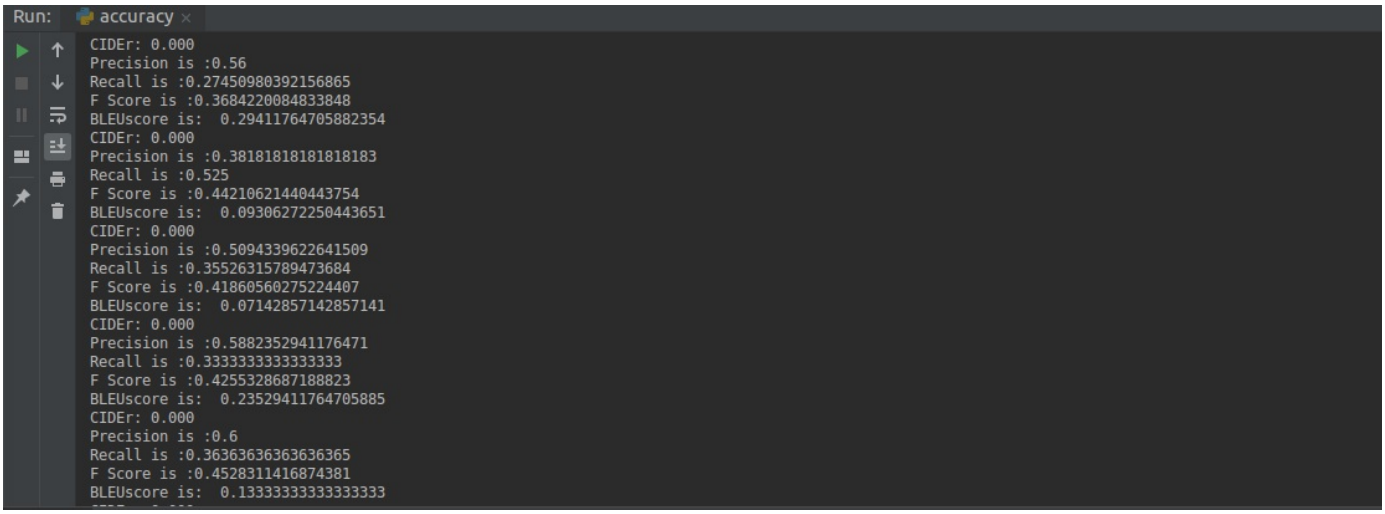
# ROUGE

- ROUGE, or Recall-Oriented Understudy for Gisting Evaluation
- It is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing.
- The following five evaluation metrics are available.

- ROUGE-N :Overlap of N-grams between the system and reference summaries.
- ROUGE-L: Longest Common Subsequence (LCS)
- ROUGE-W: Weighted LCS-based statistics that favors consecutive LCSes
- ROUGE-S: Skip-bigram based co-occurrence statistics. Skip-bigram is any pair of words in their sentence order.
- ROUGE-SU: Skip-bigram plus unigram-based co-occurrence statistics.

# Output

Here in this accuracy score ROUGE-L BLEU and CIDEr are as follows



```
Run:  accuracy
CIDEr: 0.000
Precision is :0.56
Recall is :0.27450980392156865
F Score is :0.3684220084833848
BLEUscore is:  0.29411764705882354
CIDEr: 0.000
Precision is :0.38181818181818183
Recall is :0.525
F Score is :0.44210621440443754
BLEUscore is:  0.09306272250443651
CIDEr: 0.000
Precision is :0.5094339622641509
Recall is :0.35526315789473684
F Score is :0.41860560275224407
BLEUscore is:  0.07142857142857141
CIDEr: 0.000
Precision is :0.5882352941176471
Recall is :0.3333333333333333
F Score is :0.4255328687188823
BLEUscore is:  0.23529411764705885
CIDEr: 0.000
Precision is :0.6
Recall is :0.36363636363636365
F Score is :0.4528311416874381
BLEUscore is:  0.13333333333333333
```

# Conclusion

- According to BLEU the accuracy of the model is low.
- BLEU value ranges from 0 to 1. 1 being most accurate and 0 being not accurate at all.
- Here we are getting an average of 0.3 accuracy which is not

good enough

- According to ROUGE precision of image is around 0.45
- CIDEr score is very less near to 0.01

# References


- https://machinelearningmastery.com/calculate-bleu-score-for-text-python/
- https://github.com/salaniz/pycocoevalcap
- https://gist.github.com/kracwarlock/c979b10433fe4ac9fb97
- https://github.com/pcyin/PyRouge