# CS5542 Big Data Analytics and App

# Lab Assignment #3

**Submitted by:**

**Koushik Katakam – 10**

**Team – 5**

**Objectives:**

There are two objectives of the Lab assignment 3:

- Image Caption Generator
- Data Analytics based on Unsupervised Learning

**Technologies:**

Pycharm – IDE for executing the python files

IntelIJ - IDE for executing the Scala files

**Packages used:**

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

Subject: Big Data Analytics and Applications
Scope of the Document: Lab Assignment 3

**Explanation of Objective-1:**

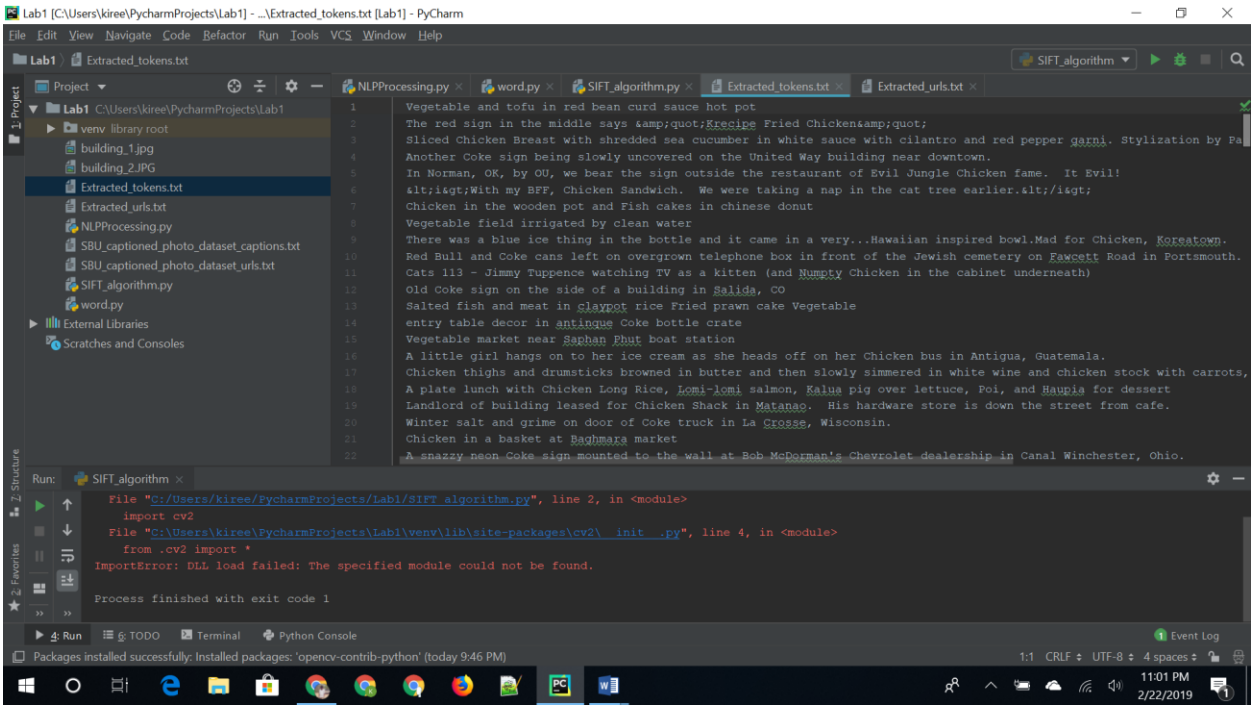
Create your own Show and Tell Model using your dataset.

Generate captions for your own dataset using theShow and Tellmodel.

Report your accuracy in BLEU, CIDER, METEOR and ROGUE measures.

A Stony Brook University (SBU) dataset has been choosen as it contains two files namely image data and text data. In the SBU dataset the image data is in the form of text data ehich contains urls and the captions data is in the form of text file. This dataset is choosen because accessing a text file is easier compared to any other files.

The caption data for the required keywords:



A Show and Tell model is created for a Caption Generator technique. The output of the show and tell model is as follows.

```
Therefore the BLEU score evaluates to 0, independently of
  1) a white plate topped with meat and vegetables . (p=0.000661)
how many N-gram overlaps of lower order it contains.
Blue cumulative 1-gram: 0.214708
Blue cumulative 2-gram: 0.000000
Consider using lower n-gram order or use SmoothingFunction()
Glue score for this sentence: 0.11764705882352941
  warnings.warn(_msg)
  2) a white plate topped with meat , potatoes and vegetables . (p=0.000306)
C:\Users\kiree\PycharmProjects\Lab1\venv\lib\site-packages\nltk\translate\bleu_score.py:523: UserWarning:
Blue cumulative 1-gram: 0.161348
The hypothesis contains 0 counts of 3-gram overlaps.
Blue cumulative 2-gram: 0.000000
Therefore the BLEU score evaluates to 0, independently of
Glue score for this sentence: 0.09523809523809523
how many N-gram overlaps of lower order it contains.
  3) a plate of food on a table (p=0.000286)
Consider using lower n-gram order or use SmoothingFunction()
Blue cumulative 1-gram: 0.571429
  warnings.warn(_msg)
Blue cumulative 2-gram: 0.436436
Glue score for this sentence: 0.46153846153846156
C:\Users\kiree\PycharmProjects\Lab1\venv\lib\site-packages\nltk\translate\bleu_score.py:523: UserWarning:
  4) a white plate topped with meat , potatoes and veggies . (p=0.000279)
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
Blue cumulative 1-gram: 0.161348
how many N-gram overlaps of lower order it contains.
Blue cumulative 2-gram: 0.000000
Consider using lower n-gram order or use SmoothingFunction()
```

```
  4) a white plate topped with meat , potatoes and veggies . (p=0.000279)
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
Blue cumulative 1-gram: 0.161348
how many N-gram overlaps of lower order it contains.
Blue cumulative 2-gram: 0.000000
Consider using lower n-gram order or use SmoothingFunction()
Glue score for this sentence: 0.09523809523809523
  warnings.warn(_msg)

Process finished with exit code 0
```

**Sequence of steps required for show and tell model:**

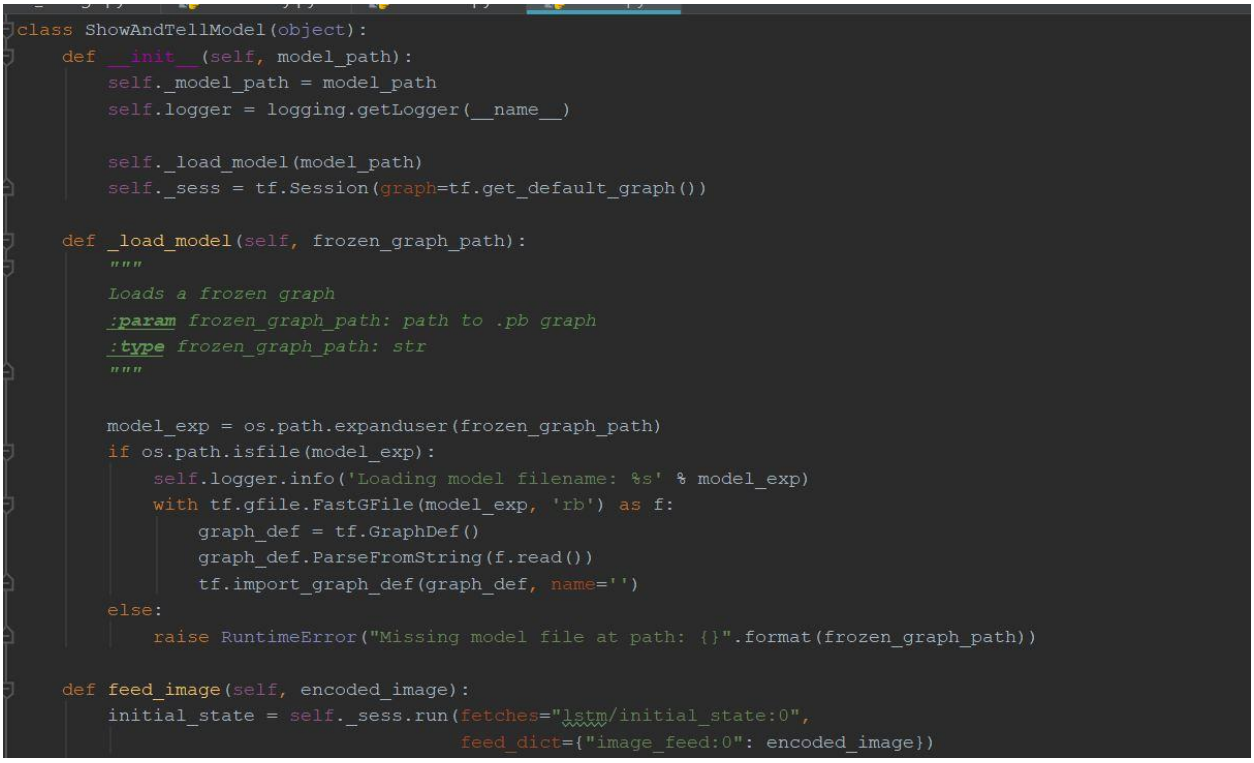
- A .pb file for the model is generated when the model is first executed which contains the model parameters
- After creating a model, we need to train the model with the necessary vocabulary file i.e., word count file.
- After training the model, testing is done with different images and then captions are generated.

**The screenshots of the show and tell model are as follows:**

The requirements of the show and tell model

```
#required libraries
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import nltk
import nltk.translate.gleu_score as gleu
import numpy
import os
try:
  nltk.data.find('tokenizers/punkt')
except LookupError:
  nltk.download('punkt')
import nltk

try:
  nltk.data.find('tokenizers/punkt')
except LookupError:
  nltk.download('punkt')
from nltk.translate.bleu_score import sentence_bleu

import logging
import math

import tensorflow as tf
```

The model functionality is described by the following snippet.



```
class ShowAndTellModel(object):
    def __init__(self, model_path):
        self._model_path = model_path
        self.logger = logging.getLogger(__name__)

        self._load_model(model_path)
        self._sess = tf.Session(graph=tf.get_default_graph())

    def _load_model(self, frozen_graph_path):
        """
        Loads a frozen graph
        :param frozen_graph_path: path to .pb graph
        :type frozen_graph_path: str
        """

        model_exp = os.path.expanduser(frozen_graph_path)
        if os.path.isfile(model_exp):
            self.logger.info('Loading model filename: %s' % model_exp)
            with tf.gfile.FastGFile(model_exp, 'rb') as f:
                graph_def = tf.GraphDef()
                graph_def.ParseFromString(f.read())
                tf.import_graph_def(graph_def, name='')
        else:
            raise RuntimeError("Missing model file at path: {}".format(frozen_graph_path))

    def feed_image(self, encoded_image):
        initial_state = self._sess.run(fetches="lstm/initial_state:0",
                                       feed_dict={"image_feed:0": encoded_image})
```

- *load model* function in the above screenshot is for loading the model. Try and catch block is used inorder to handle exceptions.
- *feed_image* function generally feeds an image to LSTM model for predicting the next word in the caption generation.
- *inference_step* function is available which is a softmax function implementation which is a final stage.

The screenshot of the Beam score is as follows:

```
class CaptionGenerator(object):
    """Class to generate captions from an image-to-text model.
    This code is a modification of https://github.com/tensorflow/models/blob/master/research/im2txt/im2txt/infe
    """

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

        self.vocab = vocab
        self.model = model

        self.beam_size = beam_size
        self.max_caption_length = max_caption_length
        self.length_normalization_factor = length_normalization_factor
```

- Beam size generally defines the number of captions to be generated for each image. From the above screenshot as beam size is 4, it generates 4 captions.

*Next feature is BLEU score which generally determines a metric for evaluating the generated sentence which varies between 0 and 1.*

The screenshot is as follows:

```
    generator = CaptionGenerator(model, vocab)
    for filename in filenames:
        with tf.gfile.GFile(filename, "rb") as f:
            image = f.read()
        captions = generator.beam_search(image)
        print("Captions: ")
        for i, caption in enumerate(captions):
            sentence = [vocab.id_to_token(w) for w in caption.sentence[1:-1]]
            sentence = " ".join(sentence)
            temp = "  %d) %s (p=%f)" % (i+1, sentence, math.exp(caption.logprob))
            print(temp)
            comp = [sentence.split()]
            # Calculating The Blue Score
            print('Blue cumulative 1-gram: %f' % sentence_bleu(comp, candidate, weights=(1, 0, 0, 0)))
            print('Blue cumulative 2-gram: %f' % sentence_bleu(comp, candidate, weights=(0.5, 0.5, 0, 0)))
            # Glue Score
            G = gleu.sentence_gleu(comp, candidate, min_len=1, max_len=2)
            print("Glue score for this sentence: {}".format(G))
```

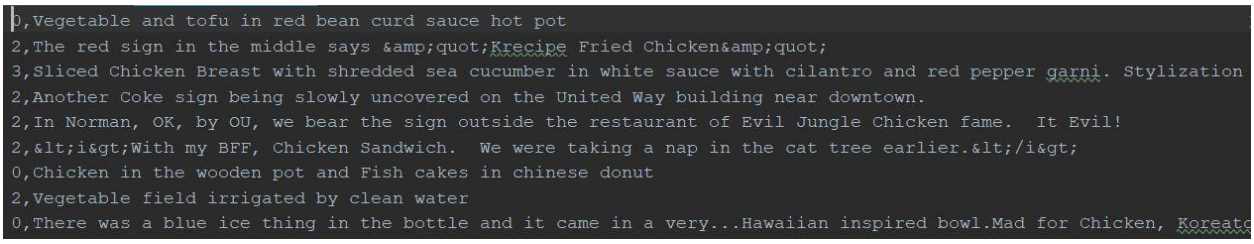
**Explanation of Objective-2:**

**Sequence of steps for Data Analytics based on Unsupervised Learning**

- The major goal is to implement various clustering techniques where we have been implemented KMeans and EM clustering.
- This is implemented on unsupervised data.

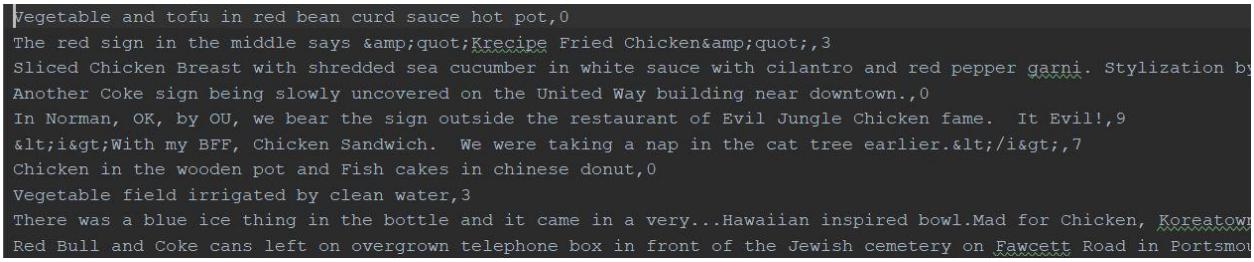
**The screenshots of the techniques are as follows:**

The output of KM_clustering is as follows:



```
0,Vegetable and tofu in red bean curd sauce hot pot
2,The red sign in the middle says &amp;quot;Krecipe Fried Chicken&amp;quot;
3,Sliced Chicken Breast with shredded sea cucumber in white sauce with cilantro and red pepper garni. Stylization
2,Another Coke sign being slowly uncovered on the United Way building near downtown.
2,In Norman, OK, by OU, we bear the sign outside the restaurant of Evil Jungle Chicken fame.  It Evil!
2,&lt;i&gt;With my BFF, Chicken Sandwich.  We were taking a nap in the cat tree earlier.&lt;/i&gt;
0,Chicken in the wooden pot and Fish cakes in chinese donut
2,Vegetable field irrigated by clean water
0,There was a blue ice thing in the bottle and it came in a very...Hawaiian inspired bowl.Mad for Chicken, Koreato
```

The output of EM_clustering is as follows:



```
Vegetable and tofu in red bean curd sauce hot pot,0
The red sign in the middle says &amp;quot;Krecipe Fried Chicken&amp;quot;,3
Sliced Chicken Breast with shredded sea cucumber in white sauce with cilantro and red pepper garni. Stylization by
Another Coke sign being slowly uncovered on the United Way building near downtown.,0
In Norman, OK, by OU, we bear the sign outside the restaurant of Evil Jungle Chicken fame.  It Evil!,9
&lt;i&gt;With my BFF, Chicken Sandwich.  We were taking a nap in the cat tree earlier.&lt;/i&gt;,7
Chicken in the wooden pot and Fish cakes in chinese donut,0
Vegetable field irrigated by clean water,3
There was a blue ice thing in the bottle and it came in a very...Hawaiian inspired bowl.Mad for Chicken, Koreatown
Red Bull and Coke cans left on overgrown telephone box in front of the Jewish cemetery on Fawcett Road in Portsmou
```

The code snippet for the KM_clustering technique:

```
object kM_Clustering {
  def main(args: Array[String]): Unit = {
    System.setProperty("hadoop.home.dir", "Desktop\\winutils")
    val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
    val sc = new SparkContext(sparkConf)

    val features=sc.textFile( path = "data/Extracted_tokens.txt")
      .map(f=>{
        val str=f.replaceAll( regex = ",", replacement = "")
        val ff=f.split( regex = " ")
        ff.drop(1).toSeq
      })
```

- The above screenshot represents the input data along with setting the hadoop property.

```
    val hashingTF=new HashingTF()

    val tf=hashingTF.transform(features)
    val kMeansModel=KMeans.train(tf,10, maxIterations = 1000)

    val WSSSE = kMeansModel.computeCost(tf)
    println("Within Set Sum of Squared Errors = " + WSSSE)

    val clusters=kMeansModel.predict(tf)
    val out=new PrintStream( fileName = "data\\results_kM.csv")
    features.zip(clusters).collect().foreach(f=>{
      out.println(f._2+","+f._1.mkString(" "))
    })
  }

}
```

- The code represents pushing the captions into a hash map.
- After storing them into map respective clustering is done and results are stored in a csv file.