**Table of Contents:**

**1. Image Caption Generator**

[**a. Create a show and tell model using our sports data.**](https://github.com/kaphc/CS5542_Big_Data_and_Analytics_ICP/wiki/LAB-3#q1a)  
[**b. Generate new captions for the sports data using the trained model.**](https://github.com/kaphc/CS5542_Big_Data_and_Analytics_ICP/wiki/Lab-3#b-generate-new-captions-for-the-sports-data-using-the-trained-model)  
[**c. Measure the accuracy of the generated captions in BLEU, CIDER, METEOR, and ROGUE.**](https://github.com/kaphc/CS5542_Big_Data_and_Analytics_ICP/wiki/Lab-3#c-measure-the-accuracy-of-the-generated-captions-in-bleu-cider-meteor-and-rogue)

**2. Data Analytics based on Unsupervised Learning**

[**Perform analytics on the sports data using machine and deep learning based upon unsupervised learning.**](https://github.com/kaphc/CS5542_Big_Data_and_Analytics_ICP/wiki/LAB-3#2-data-analytics-based-on-unsupervised-learning-1)

**1. Image Caption Generator**

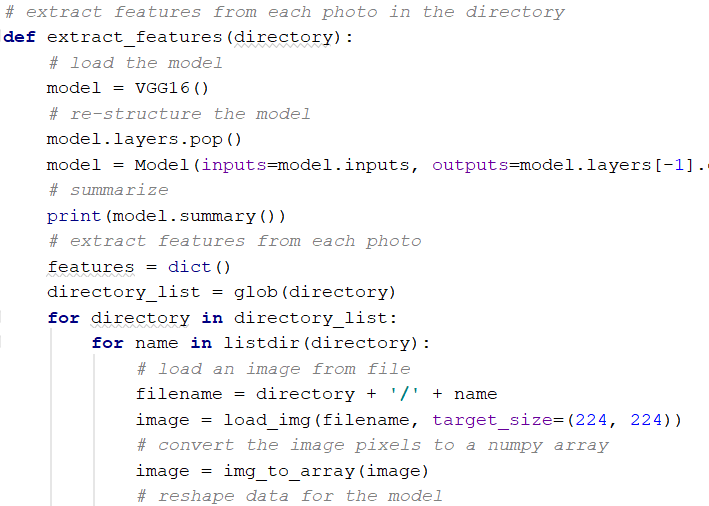
Create/generate captions for the images.

**a. Create a show and tell model using our sports data.**

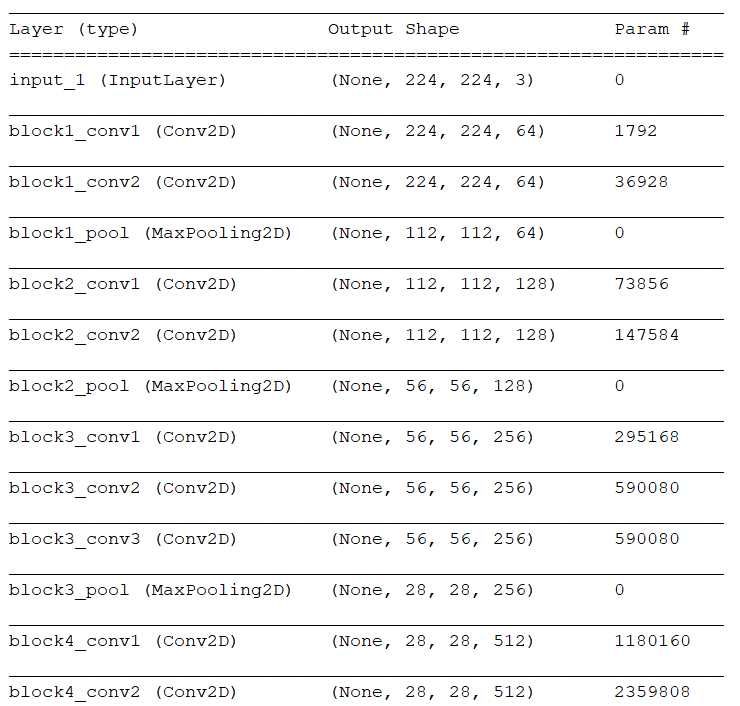
**i. Preparing the image data**

input -> set of sports images   
output -> features.pkl (all the features of the given images)

1. At first, we have used a pre-trained model to interpret the contents of the images. We have used [VGG model](https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/) to do that.
2. We have removed the last layer from the loaded model. Since we are interested only in the internal representation that is featured.
3. extract features in the below code will load the image, prepare it for VGG and store the predicted features from the VGG model.
4. At last, save this predicted feature to "features.pkl".



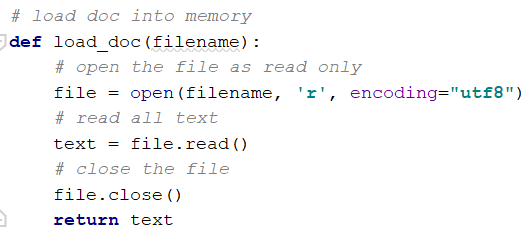
**OUTPUT**



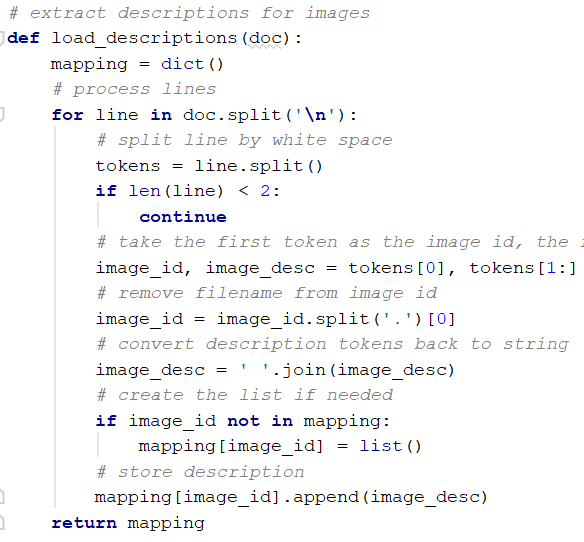
**ii. Preparing the captions data**

input -> sports\_captions.txt (captions in uppercase, punctuations, etc.,)  
output -> description.txt (cleaned captions)

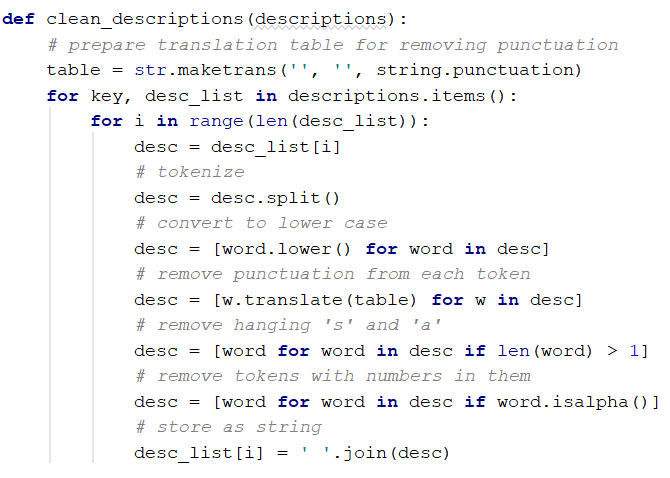
1. At first, we load the captions dataset in load\_doc().



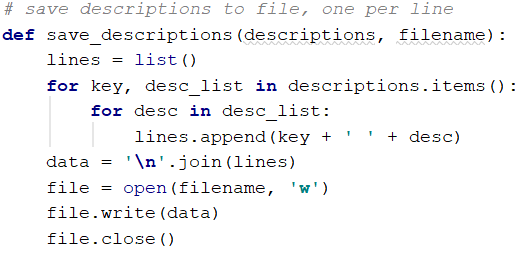
1. Next, we have created a dictionary containing filenames and its corresponding captions in load\_descriptions.



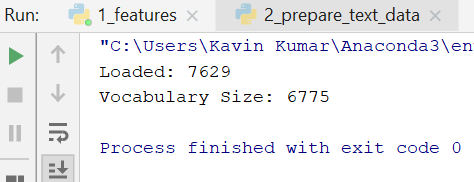
1. We cleaned the descriptions that are converting all words to lowercase, remove all punctuations, remove all words less than one or equal to one character and remove words with numbers in clean\_description().



1. At last, we stored these cleaned descriptions to descriptions.txt in save\_descriptions().



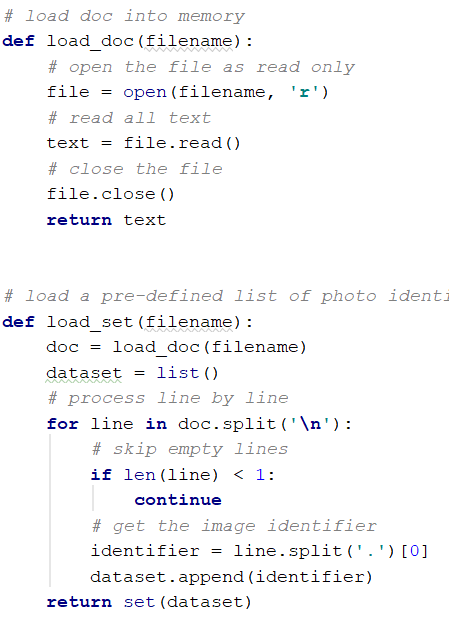
**OUTPUT**



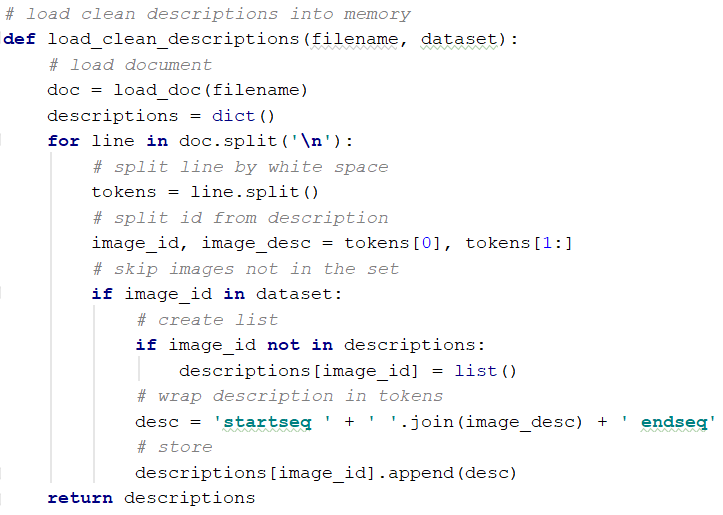
**iii. Train the Deep Learning Model**

input -> train\_images, features.pkl (created features) and descriptions.txt  
output -> H5 model

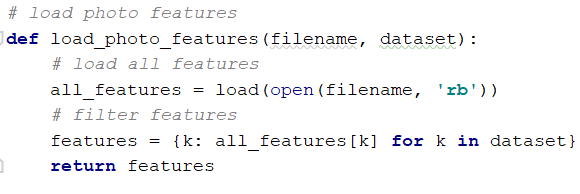
1. Loading data  
   a. load\_set() will load the all the set of filenames from the passed text file name



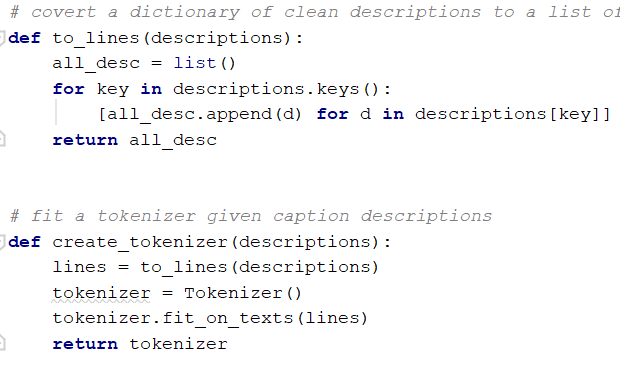
b. load\_clean\_descriptions() will now load the cleaned descriptions from "description.txt" and return the dictionary of identifiers to the list of text captions. we also add "startseq" and "endseq" as the end model will need first word to kick-off the text generation process and the last word to signal the end of the process.



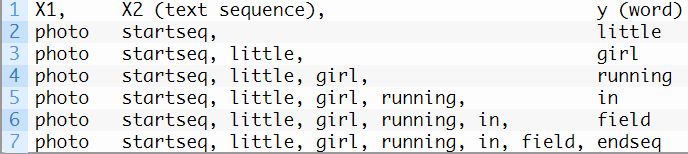
c. load\_photo\_features() will load the photo features and filter only what we needed



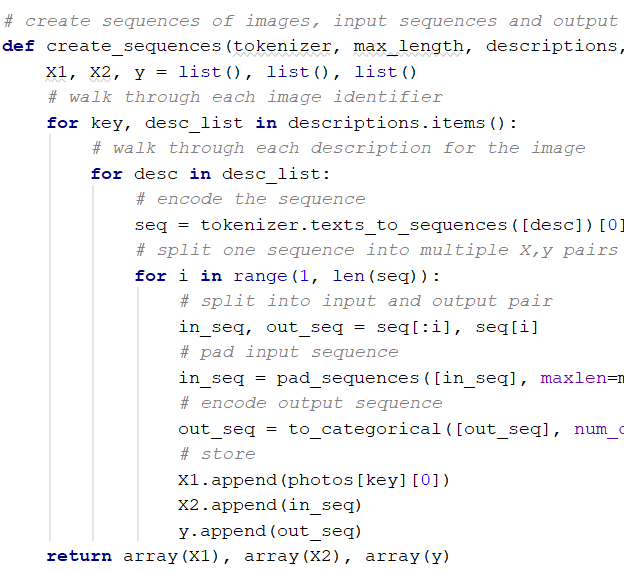
d. to\_lines() will convert the dictionary into a list of strings and create\_tokenizer to fit a tokenizer.



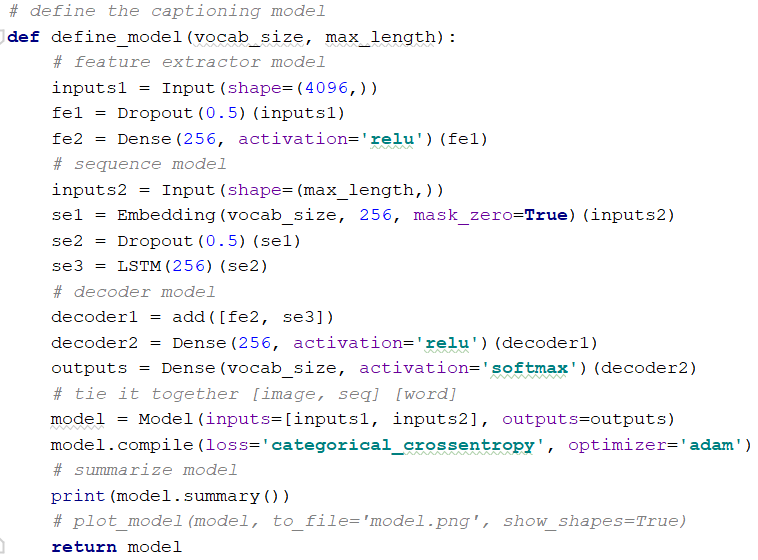
e. for each of the caption we split them into words like this



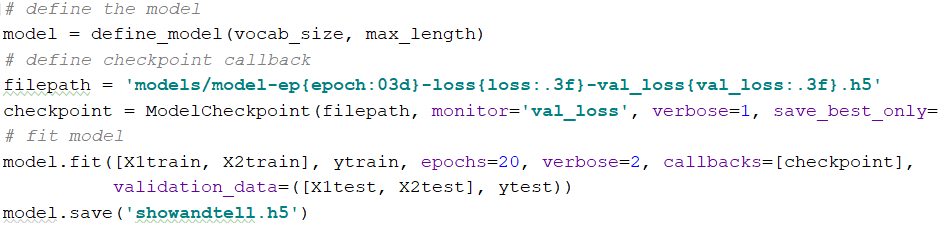
that is creating tuples with an image and first word, then image with first two words.

f. create\_sequences() will transform the data in the above format that is two input arrays. one for photo feature and other is for the encoded text. Output for the next word in sequence.   
Input text is encoded as integers ad fed into word embedding layer. photo features will be given directly. The model will now output a prediction as a probability distribution over other words in the dictionary. output data is encoded using one-hot  


1. Defining the model  
   The photo feature extractor model needs input photo features of a vector of 4096 elements. The sequence processor model needs input sequences with a predefined length which is fed to embedding layer and followed by an LSTM layer with 256 memory units.  
   Both of the above model produce 256 element vector and 50% dropout to avoid overfitting.  
   At last decoder model combines both using an addition operation and fed to a dense 256 neuron layer that can predict the entore output vocabulary for the next word.

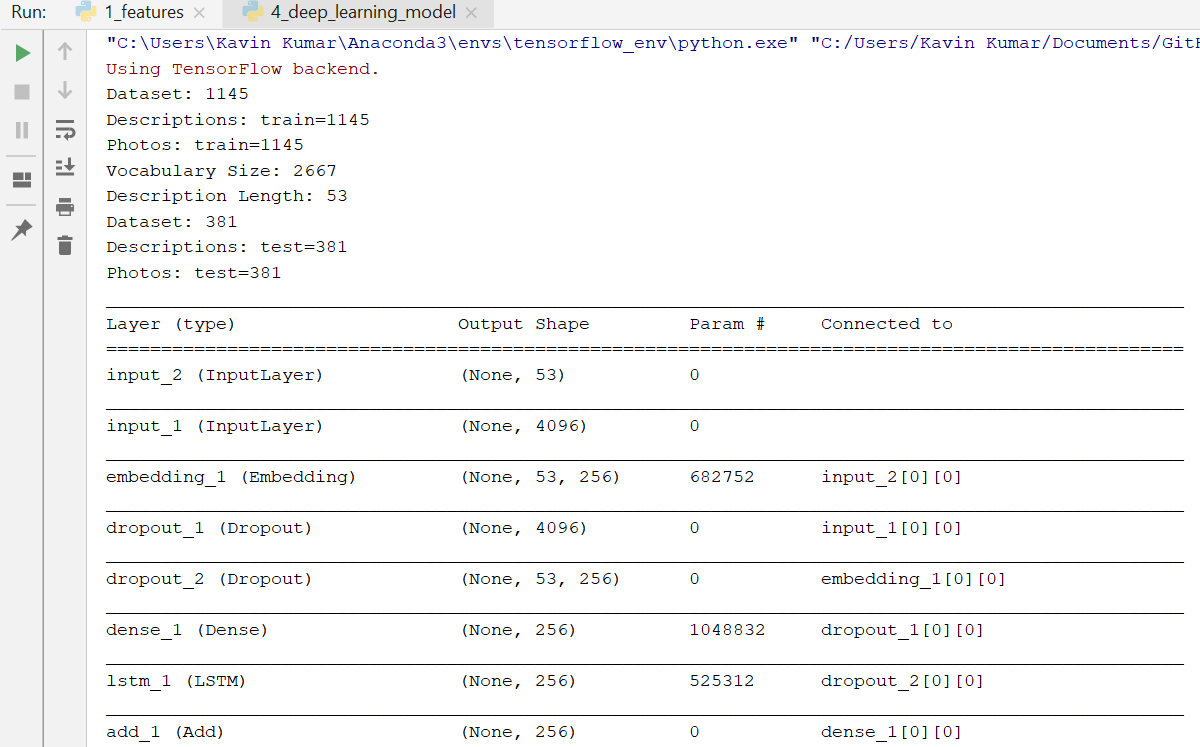


1. Fit the defined model  
   As our model learns fastly which can lead us to overfitting. so we have dev data to push out the model whenever the model has minimum loss on the validation dataset.



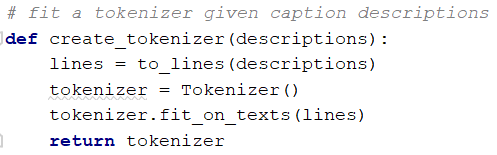
At last we have created the show and tell model.

**OUTPUT**

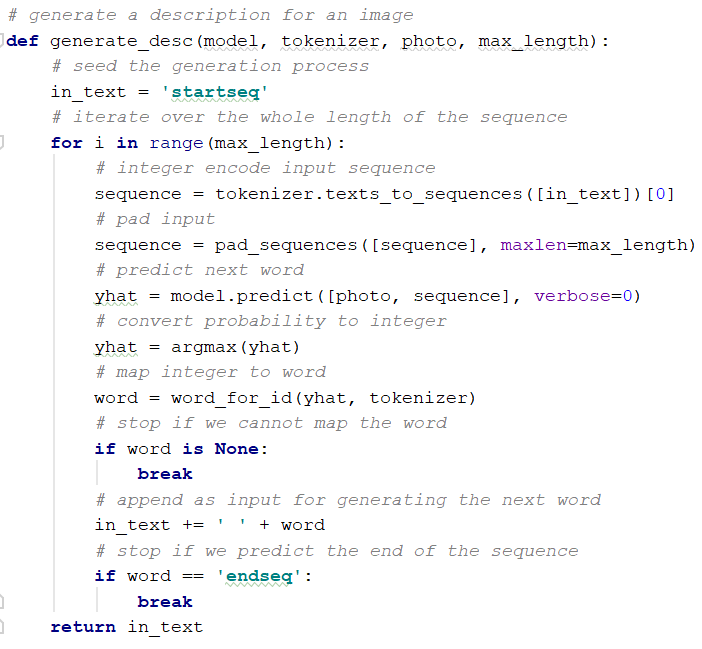


**b. Generate new captions for the sports data using the trained model.**

input -> descriptions.txt intermediate output -> tokenizer.pkl(contains all set of words in the descriptions.txt file) final output -> generated captions Now to generate the new captions we need tokenizer and save them to tokenizer.pkl



At last we create or generate the sequence with startseq for our test image.

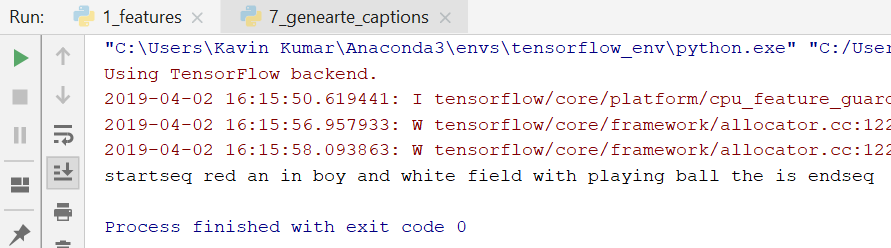


**OUTPUT**

Image

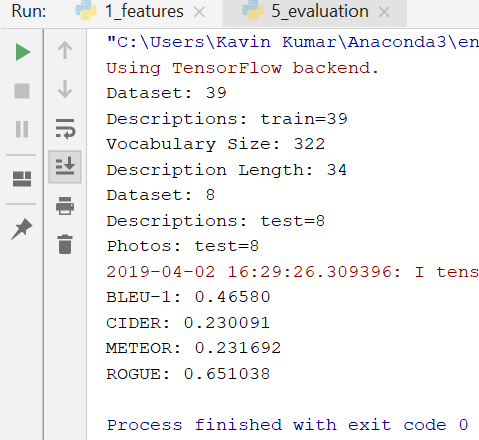


caption



**c. Measure the accuracy of the generated captions in BLEU, CIDER, METEOR, and ROGUE.**

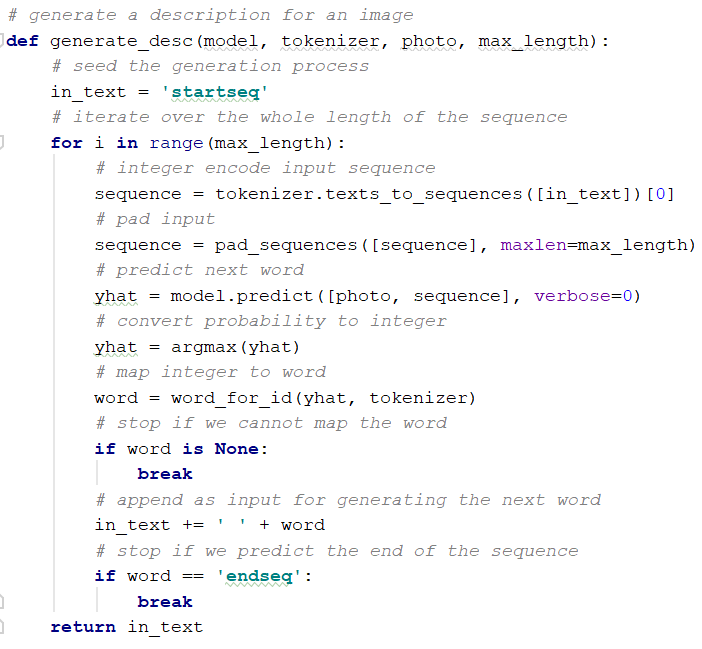
**OUTPUT**

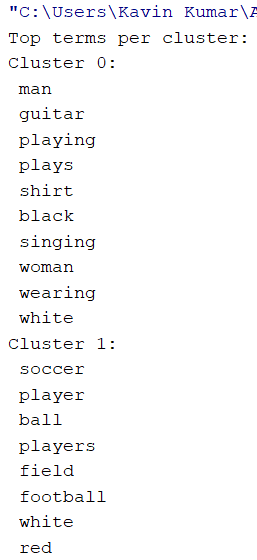


**2. Data Analytics based on Unsupervised Learning**

**a. Normal KMeans Clustering (only machine learning without deep learning)**

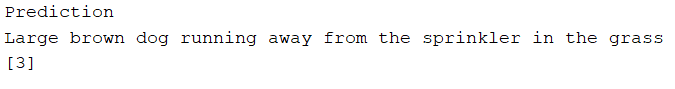
We have loaded the caption data file and created five clusters. We also printed the top words from each of these words. Predicted a sentence and found which cluster it belongs to.





**Inference**

We where able to cluster the words like one of our clusters had words related to sports and other to objects.



Belongs to third cluster.