Image captioning with flickr8k dataset

**Flickr8k dataset**

* The Flickr8k dataset is a collection of 8,092 images with up to five captions for each image. The dataset is available in English and can be used for tasks such as image-to-text and text-to-image. The captions are provided by five different people, as an image can be described in multiple ways.
* In Flickr\_8K dataset, all the images of training, validation and test set are in one folder.
* It contains 3 different files i.e *Flickr\_8k.trainImages.txt*,*Flickr\_8k.testImages.txt*, *Flickr\_8k.devImages.txt* corresponding to each type of dataset i.e train, test and validation set, each file having file\_name of images conatined in each dataset.
* Each image is given 5 different captions by 5 different humans. This is because an image can be described in multiple ways.These captions are stored in *'Flickr8k.token.txt'*.

Dependencies used in this project

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow

import keras\_preprocessing.text as kpt

from tqdm import tqdm

from keras.applications import vgg16

from keras.preprocessing import image

from keras.applications.vgg16 import preprocess\_input

from tensorflow.keras.applications.resnet50 import ResNet50

from keras.models import Model

from keras\_preprocessing.text import one\_hot

from keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.utils import to\_categorical

from nltk.translate.bleu\_score import sentence\_bleu

from numpy import array

import pickle

import numpy as np

import os

import cv2

Data visualization and pre-processing

Visualizing one of the train image using following code

img=cv2.imread("all\_images/Flicker8k\_Dataset"+ "/" +train\_image\_names.img\_id[0])

plt.imshow(cv2.cvtColor(img,cv2.COLOR\_BGR2RGB))

Transforming data into dictonary mapping of image\_id to encoded captions

This can be achieved by following code,

train\_encoded\_captions={}

for img\_id in tqdm(train\_captions):

  train\_encoded\_captions[img\_id]=[]

  for i in range(5):

    train\_encoded\_captions[img\_id].append([words\_to\_indices[s] for s in train\_captions[img\_id][i].split(" ")])

**Preprocessing**

Some elements of caption sentences do not carry any meaning. Examples include uppercase alphabets, punctuation and numbers (at least for our purposes). First we will clean each sentence of these entities. Then, we will add two special tokens at the start and end of each sentence, to indicate the beggining and ending of the sentence (startseq and endseq respectively). These help the model know when to start and stop predicting. Once that is done, we tokenize all the lines using the Tokenizer class from keras.preprocessing.text submodule.

## Embeddings

Integers are not the best way to capture information about words and their meanings.

## **Data Generator**

I know this has gone too long, but here's one last thing before we start coding. Typically the datasets used for these will have very large sizes. Plus, for computing crossentropy, your predictions and targets will be one hot vectors of size 8372. If you try to pass the entire dataset in this form to your machine, it might run out of memory soon (and hang). To prevent this, we use a **data generator**.

We are first going to build the data generator for our model. Then, we will move on to building the model itself.

# **Training the model**

1. For every epoch, initialize the data generator and loss counter.
2. Perform len(all\_captions) // batch\_size number of iterations on the generator. This will ensure that we go over each image and all of its captions in the training dataset.
3. Perform a learning step for the batch generated during an iteration and update relevant objects.
4. Perform some console outputs so you can track its progress.
5. Save the model.

**Greedy search**

Greedy search is a simple and intuitive algorithm used in problem-solving and optimization tasks. The core idea is to make the most favorable choice at each step with the hope of finding a global optimum.

**Property**: At each step, the algorithm chooses the best option available at that moment, without considering the consequences for future steps.

# Evaluation

Now that we have a trained model, we'll see how well it does. We will write a translation function, and another function to quantify its performance using a performance metric called BLEU score.

### BLEU scores

To compare model performance for seq2seq tasks (ones where it generates a sequence and we have to match it with reference sequences to see how well it has done), we use **BLEU scores**.

**Predicting Captions on Test Set using Greedy Search**

