Image Captioning using Flickr8K Dataset

Aditya Kismatrao, Manas Mishra, Pratik Pandey, Yash Srivastava
Robert H. Smith School of Business
University of Maryland
College Park, MD 20740
{amkismat, manasm, pratikp, yashsri}@umd.edu

Abstract

Image Captioning is an important research topic at the intersection of computer vision and natural language processing. With recent advances in deep learning including the development of convolutional and recurrent neural networks, great strides have been made in image captioning techniques. In this project, we aim to showcase the image captioning methodology using deep learning and the Flickr8K dataset which provides images and it's captions. We have presented a vanilla architecture to generate captions and discuss the results and observations during our experiments.

1 Introduction

1.1 The problem

Image Captioning is a field of artificial intelligence that is swiftly becoming an important aspect of computer vision. Image indexing is one application of Image Captioning and image indexing is an important part of Content-based image retrieval (CBIR) which has its applications in military, education, digital libraries, and bio-medicine to name a few.

Human-computer interaction can be an important thing when it comes to developing solutions for visually impaired people and image captioning could enable them to talk to the world more efficiently. Through image retrieval and video captioning, visually impaired people could experience things that are normal for a person with eyesight. Our project aims at alleviating some of their problems. These are some problems we hope to solve through our solution.

1.2 Significance of the problem

At first, image captioning might look like it has its most important application in social media due to the billions of images being circulated on Facebook, Twitter and Instagram. However, this is not the case. Image captioning can be a revolutionary subject in changing the way human-computer interaction occurs.

Data storage and retrieval has advanced and nowadays we have large data sets full of images of different types. Military and law authorities would benefit from having systems that help with automatic fingerprint identification and prevention of crime. Current systems are laborious and time-consuming as they use the traditional keyword-based approach for image retrieval.

Biodiversity Information Systems need to be automated as researchers keep on discovering new types and species of living organisms. This increases the scope of implementation of Image Captioning significantly. Almost all important fields like medicine and education can benefit from this application in some way or the other.

Visually impaired people could have more access to comprehensive digital content. This is important because a lot of information that flows today is digital. These people would benefit greatly from this implementation. Image indexing would help create and maintain digital libraries more efficiently.

1.3 Challenges faced by current methods

Image captioning is a sophisticated artificial intelligence mechanism. Many of its applications are dependent on large data sets for information retrieval. The alternative to CBIR has always been the usual way of image retrieval i.e. keyword-based searches through large data sets. This is not efficient and has higher infrastructural demands. It also does not provide speedy retrieval.

These types of searches are also limited to search the metadata that is tagged to an image or video. If the text queried is not annotated to the image or video being searched, nothing is returned. This can be frustrating. Things that are not so important in an image might not have been added to the metadata as description. This would also result in a failed or incorrect search.

1.4 Proposed solution and its capabilities

We plan to use a combination of images and captions to be fed into our model. This will generate human-like captions after running in iterations. This approach can help create more comprehensible captions that are more relevant to the image being processed and that is the aim of this project - to be able to come up with accurate captions for images that may help people from various domains.

2 Related Work

The issue of generating description from visual graphics has always been the subject of curiosity over the years but it was mostly focused on computer vision for video. The method primitively used involved visual recognizers coupled with structured formal language via rule based systems to generate video captions. However, scope was fairly limited, the system was almost hand-designed and demonstrated mainly on sports domain.

The concept of still image description came into picture more recently after significant developments in deep learning [4]. We already saw keyword-based methods and its drawbacks for describing the image. Below are the comparisons of existing methods in the domain of computer vision for describing images.

2.1 Template-based method

In this method there is a fixed sentence template. One example of this template is subject-verb-object template [2]. First the object detection takes place, related surrounding attributes are identified and then they are filled in the assigned template. Although this method works well with out-of-the-box visual components, the sentence generation is fairly rigid. The fixed template makes the caption look less natural.

2.2 Transfer-based method

Transfer based method uses image retrieval process for captioning the image. Firstly, the images which are visually similar are retrieved and the captions are transferred to image query. This method makes the captions look less computerized. However, the flexibility to update words based on content of the image is drastically reduced because of strong dependency on retrieved images from training data.

2.3 Multi-linear model

Recently many researchers have started using purely statistical models for describing images. These models use natural language processing for predicting captions. One such model is multi-modal log bi-linear which uses probability distribution of words conditioned on an image. This is similar to the model which we adopted but use of the LSTM model gives better accuracy as compared to the non-recurrent neural network models.

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312

Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0

Figure 1: VGG16 Feature Extractor pretrained on ImageNet

3 Methodology and Architecture

The methodology used can be explained in the following parts: generation of image features, modelling captions for supervised learning, caption processing and tokenization, and combining image and text features using a sequential model. These processes are discussed in this section.

3.1 Generation of Image Features

Since captions describe a lot of the information regarding the objects in an given image, we need to extract the important features from the image using a Convolution Neural Network (CNN) using the Transfer Learning approach.

In this regard, we use the VGG16 [6] architecture pre-trained on the ImageNet [1] dataset. The ImageNet task aims to classify the image into thousand (1000) classes or objects. But for the image captioning task, we don't aim to classify the image into broad categories. Hence, we remove the last classification layer from the architecture, yielding a feature vector of size 4096.

The VGG16 feature extractor can be seen in Figure 1[1]. Image features are generated for both training and validation images.

[1]None stands for the batch size in the figure

Image Feature Vector	Input String	Target Variable
Image_1	startcap	the
Image_1	startcap the	fish
Image_1	startcap the fish	is
Image_1	startcap the fish is	in
Image_1	startcap the fish is in	the
Image_1	startcap the fish is in the	pond
Image_1	startcap the fish is in the pond	endcap

Table 1: Caption modelling for the caption: The fish is in the pond

3.2 Modelling Captions for Supervised Learning

The primary objective of image captioning is to output a caption for a given image. Hence, we have used a supervised learning methodology where the image is the input and caption is used as the target. The modelling of the caption as a target variable is described here.

- First, the captions are added with strings "startcap" and "endcap" at the start and end of the caption. These words help to indicate the start and end of the caption string.
- During training, the model is supplied with image feature vector and the first word of the caption sequence i.e. "startcap" with the next word in the caption as the target variable.
- In the next iteration, the input for the model is the image feature and the sequence: "startcap" + target variable from the previous iteration. This process continues till it reaches the end of the caption i.e. "endcap".
- For predictions, the model input is the image feature vector and the starting sequence "startcap" with the trained model yielding the caption string ending with "endcap".

An example of the caption modelling as a supervised learning problem is presented in Table 1.

3.3 Caption Processing and Tokenization

The captions were processed to remove any special characters (@, #, etc.) along with alpha-numeric terms (CR7, DM10, etc.) from the sequence. All the captions were converted to lower-case characters.

To feed the captions into the model, the captions appended with the "startcap" and "endcap" strings were tokenized using the Keras tokenizer. The vocabulary size of the caption dataset after tokenization is 8693 and the maximum length of a caption string was found to be 34 words.

3.4 Combine Image and Text Features

The image feature vector obtained from VGG16 feature extractor and the text vector generated from caption of length 34 (since that's the maximum possible length of a caption) are fed to the model architecture as shown in Figure 2[2].

The image vector is passed through a Dropout layer and a fully-connected layer reducing the size of the vector to 256. Similarly, the caption vector is embedded to a vector of size 256 using the Embedding layer and then passed through a Dropout and a LSTM layer.

The image and text features are combined together and subsequently passed through two fully-connected layers leading to an output vector containing the indexed word vector of length 8693 (the vocabulary size).

[2]Question mark denotes the batch size.

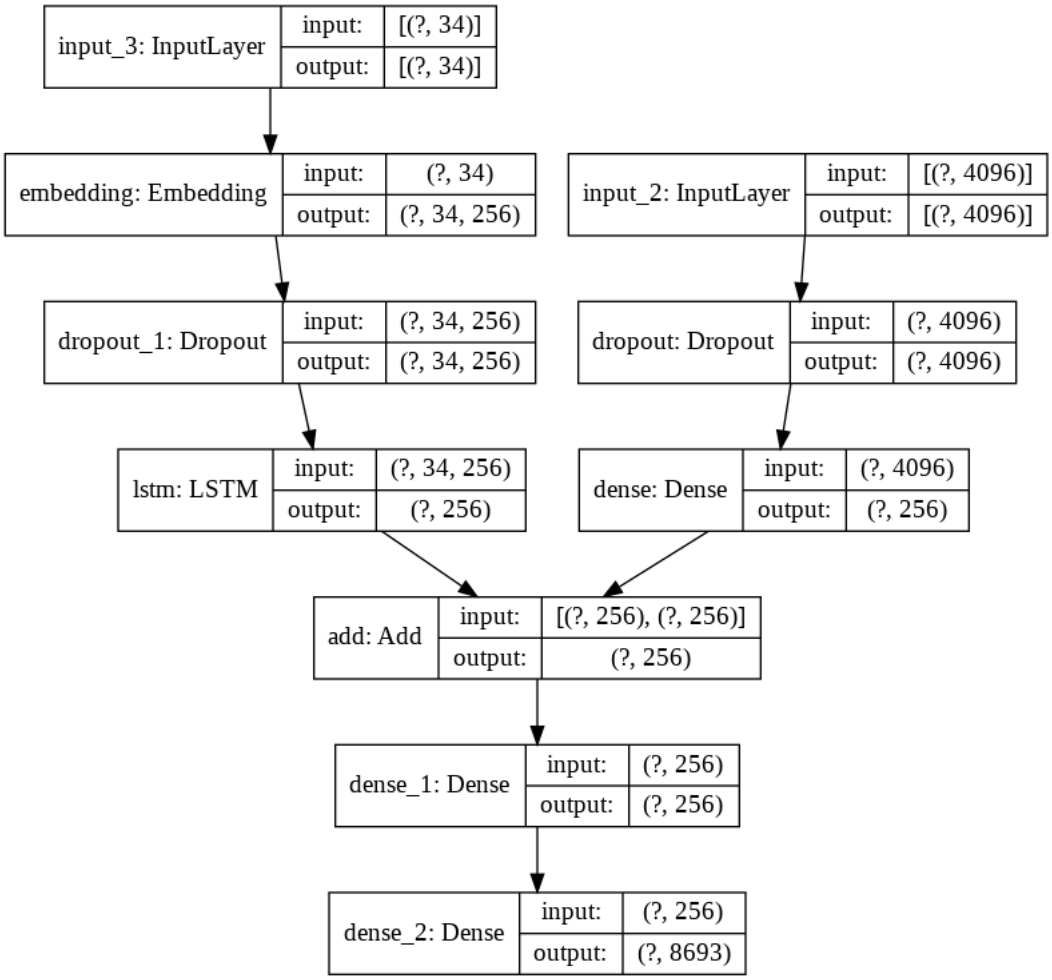


Figure 2: Model Architecture combining Text and Image feature vectors

4 Flickr8K Dataset

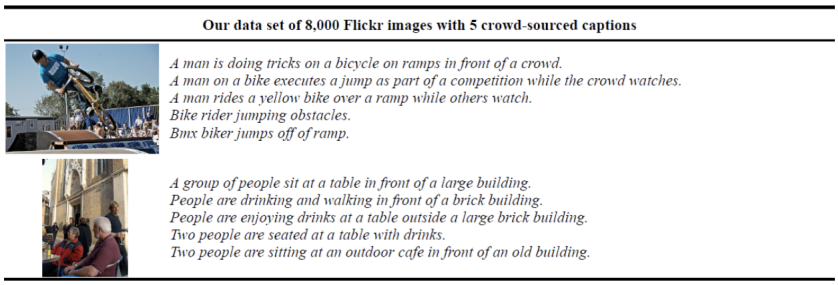


Figure 3: Example from Flickr8K dataset [3]

Based on research on image description related to image libraries, we can define three types of image captions for a single image- Conceptual image description, Non-Visual image description and Perceptual image description. We are interested in the conceptual description of the images. Conceptual description identifies exactly what is depicted in the image. It does not give us the background of the image like the mood portrayed, if the image is a painting or a drawing, what are

the most dominant shapes or additional information like the situation, time and location in which the image was taken. While the image may be abstract, we simply look at the image objectively and try to describe it.

The dataset has been gathered from a large diverse set of images scraped from photo hosting website, Flickr.com. This data set contains around 8,092 images of people and animals performing some task or action. These pictures are not taken in a popular location and do not contain any well known celebrity. To increase the diversity of the set, images are handpicked and not randomly selected. The captions used to train our model are provided by people from the United States who have passed a certain spelling and grammar test. To increase the variance, five or more descriptions are used to describe a single image. The same picture can be depicted in various ways, for example someone may describe a building as an office and someone else may simply call it a building. On an average, the length of each caption in the set is around 11.8 words. Also, in our Flickr 8K data set, 11% of it does not contain verbs and 10% of it contains commonly used verbs like wear, look, sit and stand.

5 Experimental Setup, Results, and Observations

The Flickr8K dataset consists of 6000 training, 1000 validation and 1000 test images. The settings used to train the model architecture are mentioned in Table 2.

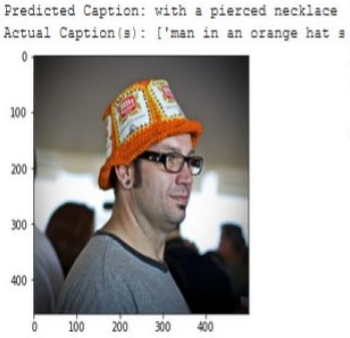
Parameters	Values
Loss Function	Categorical Cross-Entropy
Optimizer	Adam
Number of Epochs	10
Batch Size	32
Dropout rate	0.5

Table 2: Train Settings

The evaluation metrics considered for our experiment are the training and validation loss. The values obtained for these metrics after 10 training epochs are given in Table 3.

Loss	Value
Training	2.5230
Validation	4.1712

Table 3: Loss Metrics



Predicted Caption: with a pierced necklace is smiling with a drink in the background and a crowd of people in the background is wearing a blitz hat and a red headband and a red mohawk

Actual Caption(s):

1. man in an orange hat staring at something
2. man wears an orange hat and glasses
3. man with gauges and glasses is wearing a blitz hat
4. man with glasses is wearing a beer can crochet hat
5. man with pierced ears is wearing glasses and an orange hat

Figure 4: Sample output of the model with Predicted and Actual captions

Training and validation losses depict the difference between the actual output and the predicted output. Training loss is the loss obtained on training samples whereas validation loss is the loss that is obtained on unseen data and hence a good indicator of the performance of the model.

The sample output of the model is shown in Figure 4.

The baseline comparison of the model with template-based and transfer-based models is discussed below.

5.1 Comparison with Template-based model

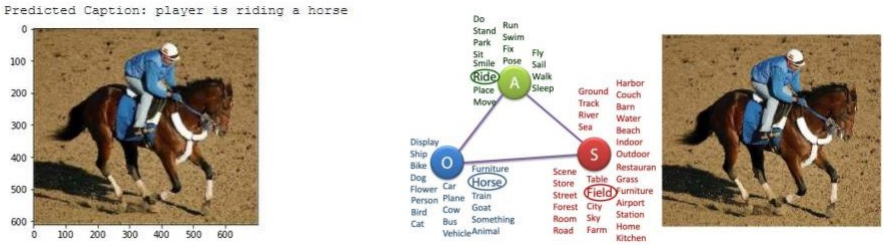


Figure 5: Comparison between our model and template-based model. Images from [2]

On comparing our model performance with the first baseline approach viz. Template-based model, it is evident that caption generated by our model was more detailed and natural as compared Farhadi et al [2] model which follows a particular template.

For detailed comparison we used an image from the paper and generated the caption as shown in Figure 5. Template used by the authors is the "Object-action-Object", hence the output generated was 'Man ride Horse'. In contrast, our model produced the caption 'player is riding a horse' along with other strings is more detailed and doesn't feel like a machine generated sequence of words.

In our model, we have used validation loss as the most important parameter. Other researchers have used parameters like BLEUScore, Tree F-1 Measure, etc. Since our model bases its performance on other metrics and hence, like-for-like baseline comparison could be misleading and less workable. This doesn't affect the output of our model and comparisons can be made very efficiently.

5.2 Comparison with Transfer-based model

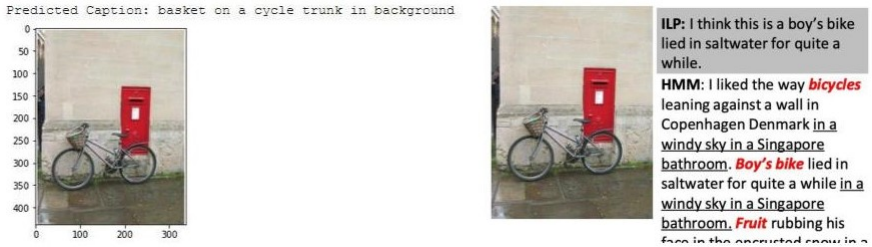


Figure 6: Comparison between our model and transfer-based model. Images from [5]

We also compared our model with another method viz. Transfer-based model which supposedly used image retrieval approach for captioning process. Although this process generated captions which felt very natural but it added way too many redundant words.

As can be seen from Figure 6, although the caption generated by Kuznetsova et al. [5] "I think this is a boy's bike lied in saltwater for quite a while" seems human-like, it seems presumptuous. On the contrary, our model keeps it simple and generates relevant caption(s).

Again, some of the parameters used by researchers were different from the metrics used in our model. Hence, we prioritized comparing outputs rather than trying to map all performance metrics of the two models.

6 Conclusion

Image captioning is one of the most sophisticated artificial intelligence tools today. Inspired by recent advances in deep learning, neural machine translation models have lately led to drastic improvements in image captioning domain. It has applications in multiple domains as listed in our report. We believe that the model we have built can apply to a wider gamut of problems, not limited to fields mentioned in the paper. When we compare our results to baseline architectures, our concerns about image captioning's reliability are alleviated.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[2] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 15–29, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15561-1.

[3] Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47: 853–899, 05 2013. doi: 10.1613/jair.3994.

[4] MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. *ACM Comput. Surv.*, 51(6), February 2019. ISSN 0360-0300. doi: 10.1145/3295748. URL `https://doi.org/10.1145/3295748`.

[5] Polina Kuznetsova, Vicente Ordonez, Alexander C. Berg, Tamara L. Berg, and Yejin Choi. Collective generation of natural image descriptions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, page 359–368, USA, 2012. Association for Computational Linguistics.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

Appendix

README

The code has been written using the Keras deep-learning library with Tensorflow backend. Following libraries are required to run the code:

- os
- requests
- zipfile
- io
- numpy
- datetime
- matplotlib
- pickle
- nltk
- re
- BeautifulSoup (bs4)
- Keras
- Tensorflow

The above libraries either come pre-installed or can be installed using the pip command in a Python virtual environment (virtualenv).

The Flickr8K dataset can be downloaded from the following links:

- Images: https://github.com/jbrownlee/Datasets/releases/download/Flickr8k/Flickr8k_Dataset.zip
- Text: https://github.com/jbrownlee/Datasets/releases/download/Flickr8k/Flickr8k_text.zip

According to the location of the files, the folder and data paths mentioned in the code need to be modified.

The whole code is accessible on Google Colaboratory with UMD Smith Login via the following link:

https://colab.research.google.com/drive/1ZOA195yLGo2OtVDMALI_pc5mq2lSeMVg?usp=sharing

Code

Image Captioning using Flickr8K Dataset

BUDT758 - Big Data and Artificial Intelligence

Team Members: Aditya Kismatrao, Manas Mishra, Pratik Pandey & Yash Srivastava

Google Colaboratory Link:
https://colab.research.google.com/drive/1ZOA195yLGo2OtVDMALI_pc5mq2lSeMVg?usp=sharing

Import Modules

In [ ]:
```
import os
import requests
import zipfile
import io
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
from pickle import dump, load

import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

import re
from bs4 import BeautifulSoup


import tensorflow as tf
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.models import Model, load_model
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.text import Tokenizer
from keras.utils import plot_model
from keras.layers import Input
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
from keras.layers import Dropout
from keras.layers.merge import add
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.callbacks import ModelCheckpoint
```

In [ ]:
```
%load_ext tensorboard
%matplotlib inline
```

Flickr8K Dataset

In [ ]:
```
images = "https://github.com/jbrownlee/Datasets/releases/download/Flickr8k/Flickr8k_Dat
text = "https://github.com/jbrownlee/Datasets/releases/download/Flickr8k/Flickr8k_text.

if not os.path.exists("/content/FlickrImages/"):
```

```
  r = requests.get(images)
  z = zipfile.ZipFile(io.BytesIO(r.content))
  z.extractall("/content/FlickrImages/")

if not os.path.exists("/content/FlickrText/"):

  r = requests.get(text)
  z = zipfile.ZipFile(io.BytesIO(r.content))
  z.extractall("/content/FlickrText/")
```

Data Cleaning and Processing

Image-Caption Dictionary

In [ ]:
```
captions_path = "/content/FlickrText/Flickr8k.token.txt"

image_captions = dict()

with open(captions_path) as file:

  captions = file.read()

  for caption in captions.split('\n'):

    if caption != '':

      splits = caption.split(" ")

      image_id = splits[0].split(".")[0]
      string = ' '.join(splits[1:])

      if image_id in image_captions:
        image_captions[image_id].append(string)

      else:
        image_captions[image_id] = [string]
```

Training and Validation Data

In [ ]:
```
def read_data(path):

  data = list()

  with open(path) as file:

    images = file.read()

    for image in images.split("\n"):

      if image != '':
        data.append(image.split(".")[0])

  return data
```

In [ ]:
```
train_path = "/content/FlickrText/Flickr_8k.trainImages.txt"
val_path = "/content/FlickrText/Flickr_8k.devImages.txt"

train_images = read_data(train_path)
```

```
val_images = read_data(val_path)

print("Number of Training Samples:", len(train_images))
print("Number of Validation Samples:", len(val_images))
```

Clean Text

- Clean Text by removing bad symbols and stopwords
- Reference: https://github.com/radonys/Reddit-Flair-Detector/blob/master/Jupyter%20Notebooks/Reddit_Flair_Detector.ipynb

In [ ]:
```
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))

def clean_text(text):

    text = BeautifulSoup(text, "lxml").text
    text = text.lower()
    text = REPLACE_BY_SPACE_RE.sub(' ', text)
    text = BAD_SYMBOLS_RE.sub('', text)
    text = ' '.join(word for word in text.split() if word.isalpha())

    return text

for key in image_captions:
  image_captions[key] = [clean_text(caption) for caption in image_captions[key]]
```

Text Tokenizer

In [ ]:
```
def tokenize(captions):

  caption_list = list()

  for key in image_captions:
    for caption in image_captions[key]:
      caption_list.append(caption)

  tokenizer = Tokenizer()
  tokenizer.fit_on_texts(caption_list)

  vocab_size = len(tokenizer.word_index) + 1
  max_length = max(len(caption.split()) for caption in caption_list)

  return tokenizer, vocab_size, max_length
```

Start and End Identifiers for Captions

Start Identifier: "startcap"

End Identifier: "endcap"

In [ ]:
```
def startend(image_captions, keys):

  marked_captions = dict()

  for key in image_captions:
```

```
    if key in keys:

      for caption in image_captions[key]:

        caption = 'startcap ' + caption + ' endcap'

        if key in marked_captions:
          marked_captions[key].append(caption)

        else:
          marked_captions[key] = [caption]

  return marked_captions
```

Convolutional Neural Network: VGG16

In [ ]:
```
model_cnn = VGG16()
model_cnn = Model(inputs=model_cnn.inputs, outputs=model_cnn.layers[-2].output)
```

Save Train and Validation CNN Features

In [ ]:
```
def preprocess(image_path):

    img = image.load_img(image_path, target_size=(224, 224))

    img = image.img_to_array(img)

    x = img.reshape((1, img.shape[0], img.shape[1], img.shape[2]))

    x = preprocess_input(x)

    return x
```

In [ ]:
```
def encode(image):

    image = preprocess(image)

    features = model_cnn.predict(image)

    return features
```

In [ ]:
```
images_path = "/content/FlickrImages/Flicker8k_Dataset/"

train_images_encoded = dict()

for img in train_images:
  train_images_encoded[img] = encode(images_path + img + '.jpg')

with open("/content/train_image_features.pkl", "wb") as file:
    dump(train_images_encoded, file)
```

In [ ]:
```
val_images_encoded = dict()

for img in val_images:
  val_images_encoded[img] = encode(images_path + img + '.jpg')

with open("/content/val_image_features.pkl", "wb") as file:
    dump(val_images_encoded, file)
```

Variable Declaration

In [ ]:
```
epochs = 10
batch_size = 32
```

Model Definition

In [ ]:
```
def image_captioning(vocab_size, max_length):

  #Feature Extractor (FE)
        input1 = Input(shape=(4096,))
        fe_drop = Dropout(0.5)(input1)
        fe_fc = Dense(256, activation='relu')(fe_drop)

  #Sequential Model (Captions)
        input2 = Input(shape=(max_length,))
        se_embed = Embedding(vocab_size, 256, mask_zero=True)(input2)
        se_drop = Dropout(0.5)(se_embed)
        se_lstm = LSTM(256)(se_drop)

  #Combine Features
        combine = add([fe_fc, se_lstm])
        combine_fc = Dense(256, activation='relu')(combine)
        output = Dense(vocab_size, activation='softmax')(combine_fc)

        model = Model(inputs=[input1, input2], outputs=output)

        return model
```

Data Generator

In [ ]:
```
def data_generator(captions, images, tokenizer, vocab_size, max_length, batch_size):

  X1, X2, y = list(), list(), list()
  counter = 0

  while 1:

    for key in captions:

      counter += 1

      for caption in captions[key]:

        caption_sequence = tokenizer.texts_to_sequences([caption])[0]

        for i in range(1, len(caption_sequence)):

          input_seq, output_seq = caption_sequence[:i], caption_sequence[i]

          input_seq = pad_sequences([input_seq], maxlen=max_length)[0]
          output_seq = to_categorical([output_seq], num_classes=vocab_size)[0]

          X1.append(images[key][0])
          X2.append(input_seq)
          y.append(output_seq)

      if counter == batch_size:
```

```
        yield ([np.array(X1), np.array(X2)], np.array(y))

        X1, X2, y = list(), list(), list()
        counter = 0
```

Model Compilation and Train/Validation Data

In [ ]:
```
#train_image_features = load(open("/content/train_image_features.pkl", "rb"))
train_captions = startend(image_captions, train_images)

tokenizer, vocab_size, max_length = tokenize(train_captions)

#val_image_features = load(open("/content/val_image_features.pkl", "rb"))
val_captions = startend(image_captions, val_images)
```

In [ ]:
```
model = image_captioning(vocab_size, max_length)

model.compile(loss='categorical_crossentropy', optimizer='adam')
```

Model Training and Validation

In [ ]:
```
train_steps = len(train_captions)//batch_size
val_steps = len(val_captions)//batch_size

logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)

for epoch in range(0, epochs):

  print("Epoch:", epoch)

  train_generator = data_generator(train_captions, train_images_encoded, tokenizer, voc
  val_generator = data_generator(val_captions, val_images_encoded, tokenizer, vocab_siz

  model.fit(train_generator, epochs=1, verbose=1, steps_per_epoch=train_steps, validati
```

In [ ]:
```
model.save("model_vgg.h5")
```

In [ ]:
```
%tensorboard --logdir logs
```

Model Output

In [ ]:
```
def int_to_word(integer, tokenizer):

  for word, index in tokenizer.word_index.items():

   if index == integer:
     return word

  return None
```

In [ ]:
```
def model_output(image_path, model, tokenizer, max_length):

  caption = 'startcap'
  image_feature = encode(image_path)
```

```
  for i in range(0, max_length):

    sequence = tokenizer.texts_to_sequences([caption])[0]

    sequence = pad_sequences([sequence], maxlen=max_length)

    y = model.predict([image_feature, sequence], verbose=0)
    y = np.argmax(y)

    word = int_to_word(y, tokenizer)

    if word is None:
      break

    caption += ' ' + word

    if word == 'endcap':
      break

  return caption
```

In [ ]:
```
model = load_model('/content/model_vgg.h5')
```

In [ ]:
```
image_path = "/content/FlickrImages/Flicker8k_Dataset/1000268201_693b08cb0e.jpg"
caption = model_output(image_path, model, tokenizer, max_length)

caption = caption.split(" ")[1:-1]
caption = " ".join(caption)

x = plt.imread(image_path)
plt.imshow(x)
print("Predicted Caption:", caption)
print("Actual Caption(s):", image_captions[image_path.split("/")[-1].split(".")[0]])
```