iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH
IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 Issue: X Month of publication: October 2022

DOI: https://doi.org/10.22214/ijraset.2022.47195

Image Captioning Using Deep Learning and NLP Techniques

Junaid Ahmad Wani[1], Sahilpreet Singh[2]
[1]M. Tech Scholar, [2] Assistant Professor, Department of Computer Science and Engineering, RIMT University, Mandi Gobingarh, Punjab, India

Abstract: The use of text to explain an image is known as picture captioning. It's usually used in applications that require information from a certain image to be automatically created in a textual format. This research tackles limitations by developing a vocabulary for interpreting visuals so that complicated, unified stories may be told. It uses an LSTM model with an NLP method to discover linguistic regions in pictures and builds a model that decomposes both photographs and captions into their constituent elements. It also demonstrates how to apply the LSTM Method with enhanced features for better results. The Flicker 8K dataset is used in this project

Keywords: Captions, NLP, LTSM, Deep learning

I. INTRODUCTION

The purpose of photo captioning is to describe the objects, actions, and aspects in an image using natural language. The majority of picture captioning research has focused on one-sentence captions; however this format has limited descriptive capacity; a single syllable can only fully describe a small area of an image. Recent study suggests that image paragraph captioning, with the purpose of constructing a (usually 5-8 sentence) paragraph that explains a picture, should be utilized instead. When compared to single-sentence captioning, paragraph captioning is a newer technique. The largest paragraph captioning dataset is the Visual Genome corpus, which was provided by Krause et al (2016). On this dataset, strong single sentence captioning algorithms produce repeated paragraphs that are unable to describe a range of visual attributes. Even when beam search is used, the generated paragraphs repeatedly repeat a minor variation of the same material.

Similarly, various methods for generating paragraphs are used, including Long-Term Recurrent Convolutional Network: The input might be a single image or a series of pictures from a video frame. The image is put into CNN, which identifies visual activity and develops a vector representation, which is then fed into the LSTM model, which generates a word and a caption [4].

This method requires generating a paragraph that is both coherent and thorough. A few semantic regions in an image are recognized using an attention model, and sentences are produced one after the other, resulting in a paragraph [14]. A recurrent neural network (RNN) is a form of neural network that can handle a sequence of data with a time stamp index of t ranging from 1 to t. For occupations that need sequential inputs, such as speech and language, RNNs are often preferable. If you want to predict how an NLP problem will turn out,

The gated recurrent unit (GRU) was introduced by Cho et al. as a novel concept. Similar to the LSTM unit, the GRU has gating units that control data flow inside the unit, but without the discrete memory cells. To manage the flow of data via each hidden unit, the Gated Recurrent Unit (GRU) generates two gates: updating and reset gates. Each hidden state at each step t is calculated using the formulae below. The concealed state of the previous time step is used to compute the updated gate. This gate controls how much of each new and old memory segment is included in the final memories. The reset gate is described by the equation below, but with a different set of variables

II. LITERATURE REVIEW

Picture captioning [4] and visual question answering (VQA) [12] are two examples of difficulties involving the integration of image and language comprehension that continue to spur substantial research at the confluence of computer vision and natural language processing. To create high-quality results in any of these tasks, fine-grained visual processing or even several layers of reasoning are typically necessary. These methods, which are currently based on deep neural network architectures, improve performance by learning to focus on the image's most significant features. A top-down attention mechanism and a faster R-CNN LSTM are used to achieve this

A natural image usually has a lot of semantic information and may be seen from several angles. Existing picture description methods are constrained by a restricted number of biased visual paragraph annotations and do not cover a broad variety of underlying semantics. This study provides a semi-supervised paragraph generation system that can synthesize various and semantically consistent paragraph descriptions by reasoning over local semantic domains and leveraging language input. Sentence Discriminator, Topic Transition Discriminator, and Adversarial Training Scheme are used to get the desired results.

III. METHODOLOGY

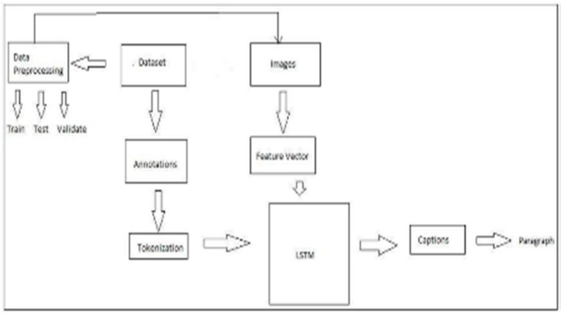


Figure 1System Architecture

The Visual Genome dataset, which contains 24,000,00 lakh photos, is used in this study. On these photos, data pre-processing is performed, dividing the dataset into train, test, and validate sets.

A. *Algorithm Steps*

1) *Step 1:* Download and pre-process the Visual Genome Dataset.
2) *Step 2:* Download the spacy English tokenizer and convert your text to tokens.
3) *Step 3:* Extract image attributes using the LSTM object detector.
4) *Step 4:* Tokenization generates features for LSTM to learn from and captions.
5) *Step 5:* A paragraph is generated by combining all of the captions.

Describe an image refers to the challenge of constructing a human-readable written representation of an image, such as a snapshot of an object or scene. The issue is also known as "photo tagging" or "automatic image annotation." It's an easy task for a human, but it's a challenging one for a machine.

B. *Proposed System*

The goal of picture clause captioning is to generate descriptions from images. This uses a hierarchical approach for text production. To begin, identify the objects in the image and make a caption for each one. The captions are then combined to form the final product.

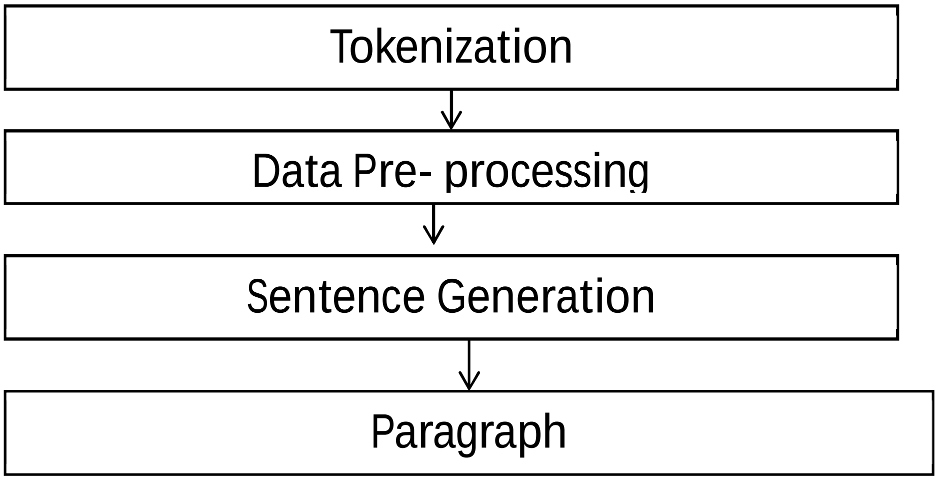


Figure 2 Proposed System Architecture

Tokenization, which separates character streams into tokens for use in data(paragraph) preparation, is the first module in this endeavour. It's the process of separating a string of letters into tokens like words, keywords, symbols, and other elements. The tokens are saved in a file and used as necessary. [15]. The act of eliminating duplicates from data and obtaining it in its purest form is known as data preparation. Data refers to images that need to be enhanced and saved in a dataset. The dataset is separated into three parts: train, test, and validate files, each of which includes the dataset's index of 14575,2489,2487 image numbers.

Object identification [4,13] is the second module in this endeavour, in which things are discovered to make the researcher's task simpler. This is done using the LSTM Model. Figure 1 depicts the process of execution. Initially, a photograph is uploaded. The first step is to look for activity in the image. The returned characteristics are then sent into an LSTM, which creates a phrase by extracting a word associated with the object characteristic. It then advances to the Intermediate level, when it generates several phrases and a paragraph...

Sentence creation is the final step in this endeavor. Objects are detected in the object characteristic, and tokens from file names are used as Captions to build words. To construct a phrase, each word is joined with the one before it.

The paragraph is the project's last module.

The produced sentences are arranged logically, giving them a distinct meaning.

The intended output is obtained as a consequence.

C. Pre-Processing Step

1) Data Pre-processing — Images

Only images are accepted as input (X) in our model. As you may know, every input to a model must be in the form of a vector.

Every image must be transformed to a fixed-size vector before being fed into the neural network as input. For this, we employ the InceptionV3 model (Convolutional Neural Network) developed by Google Research, as well as transfer learning.

This model was trained to do picture classification on 1000 different classes of images using the Visual Genome Corpus dataset. Our goal isn't to categorise the image, but rather to obtain a fixed-length informative vector for each image. This is referred to as automated feature engineering**.**

2) Data Pre-processing — Captions

It's important to remember that captions are something we want to anticipate. As a result, during the training phase, captions will be the target variables (Y) that the model will learn to predict. The whole caption, on the other hand, is not expected at the same time as the image. We'll try to predict the caption exactly. As a result, each word must be encoded into a vector of fixed size.

The dictionaries "Wordtoix" (pronounced "word to index") and "ixtoword" (pronounced "index to word")

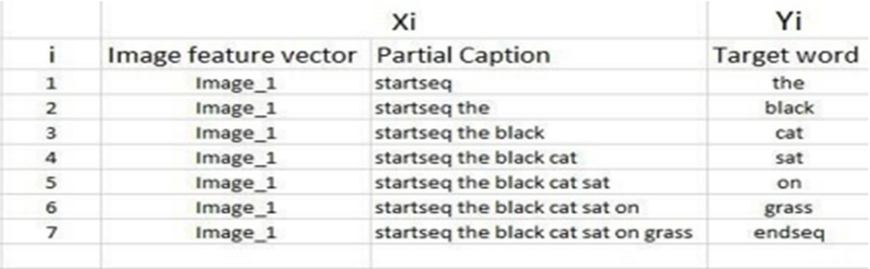
3) Data Preparation using Generator Function

Consider the first image vector Image 1 and its accompanying caption: "startseq the black cat sat on grass endseq." Keep in mind that the picture vector is the input, and the caption is what we need to forecast. However, we believe the caption will be as follows:

In the first case, we provide the picture vector and the first word as input and attempt to predict the second word using the following formula: Output = 'the'; Input = Image 1 +'startseq'

Then, as follows, we submit an image vector and the first two words as input and attempt to predict the third word: Output = 'cat'; Input = Image 1 +'startseq the'.and so forth...

As a consequence, the data matrix and description for one image



	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	startseq	the
2	Image_1	startseq the	black
3	Image_1	startseq the black	cat
4	Image_1	startseq the black cat	sat
5	Image_1	startseq the black cat sat	on
6	Image_1	startseq the black cat sat on	grass
7	Image_1	startseq the black cat sat on grass	endseq

Table 1 Data points corresponding to one image and its caption

4) Statistical Algorithm

The LSTM (Long Short-Term Memory) layer is nothing more than a customized Recurrent Neural Network that handles sequence input (partial captions in our case).

D. LSTM

It is Hochheimer and Schmid Huber that discover it. It's like an RNN. It's usually used to move data from one cell to another and to form a complete word. An input gate, an output gate, and a forget gate are the three gates in total. The input gate receives a value and distributes it to the rest of the cells. The forget gate is used to determine how much a value will be utilized in the future, based on past experiences. The output gate collects all of the data and generates a single word as an output. In addition to photos, it may handle a variety of data, such as video or sound. The LSTM equations are as follows:

$it= \sigma (xtU^{i} + ht\text{-}1W^{i})$ (1)

$ft= \sigma (xtU^{f} + ht\text{-}1W^{f})$ (2)

$ot= \sigma (xtU^{o}+ ht\text{-}1W^{o})$ (3

$\mathit{C}t=tanh(xtU^{g} +ht\text{-}1W^{g})$ (4)

$Ct= \sigma (ft*Ct\text{-}1 + it*\mathit{C}t)$ (5)

The LSTM architecture is a deep learning architecture based on artificial recurrent neural networks (RNNs). The LSTM contains feedback connections. It can handle both single data points (such as photographs) and whole data streams (such as speech or video). A typical LSTM unit consists of a cell, an input gate, an output gate, and a forget gate. The three gates regulate the flow of data into and out of the cell, and the cell remembers values for long periods of time.

Time series data is well-suited to LSTM networks for classification, processing, and prediction.

Input_1 -> Partial Caption Input_2 -> Image feature vector

Output ->The following is how to test (infer) the model by feeding in new photographs, i.e. how to create a caption for a new test image:

Remember that in the example when we learned how to prepare data, we only used the first two images and descriptions. Let's look at the third image and attempt to figure out how we want to make the caption.

The third image vector and caption were as follows:

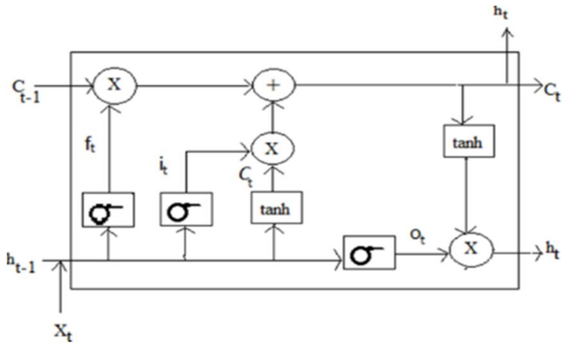


Figure 3. LSTM Architecture

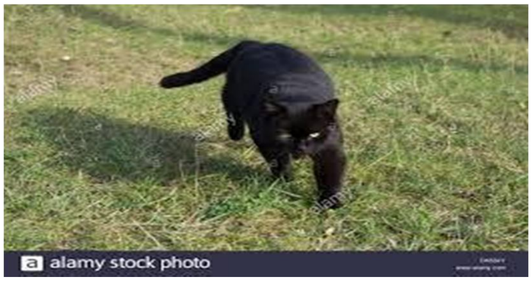


Figure. 3 Test image

The black cat is wandering through the grass. The sample also included the following vocabulary: vocab = black, cat, endseq, grass, is, on, road, sat, startseq, the, walking, white

1) Iteration 1

Image vector + "startseq" as input (as partial caption) Expected "the" is the output word.

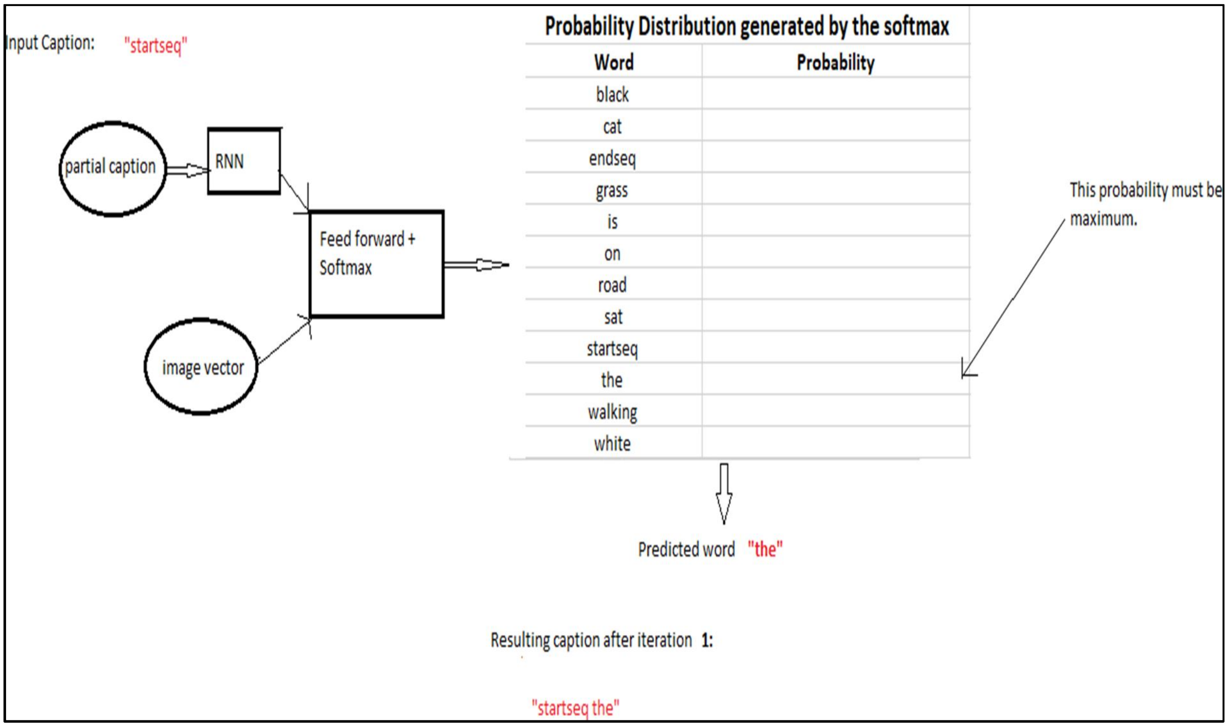
(At this point, you should understand why the token'startseq' is used as the first partial caption for every image during inference.)

However, the model generates a 12-length vector (in the case),

In the original situation, the probability distribution across all of the terms in the lexicon is a 1652-long vector. As a consequence, given the feature vector and partial caption, we chose the most likely word greedily.

We may expect the probability of the word "the" to be the highest if the model has been appropriately trained.:

Maximum Likelihood Estimation (MLE) is a method of selecting the most likely word for a given input based on the model. Because we choose the term with the highest likelihood, this strategy is also known as Greedy Search.


Input Caption: "startseq"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "the"
Resulting caption after iteration 1:
"startseq the"

Probability Distribution generated by the softmax

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 4 iteration 1

2) Iteration 2

Input: Image vector + “startseq the” Expected Output word: “black”

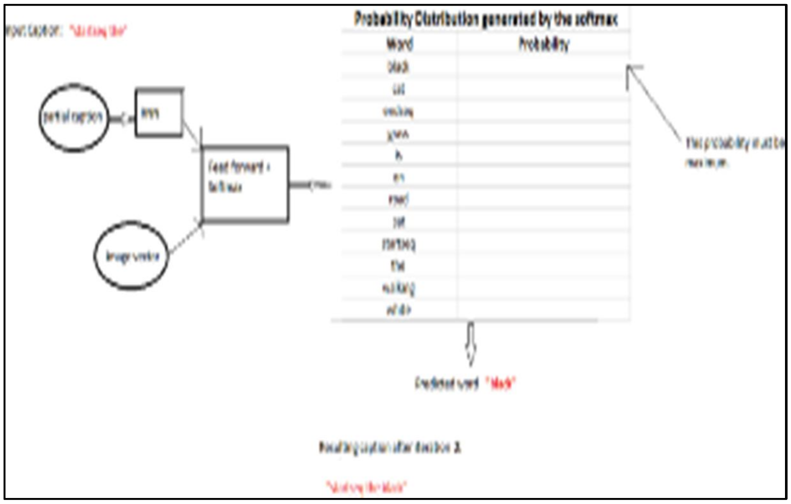
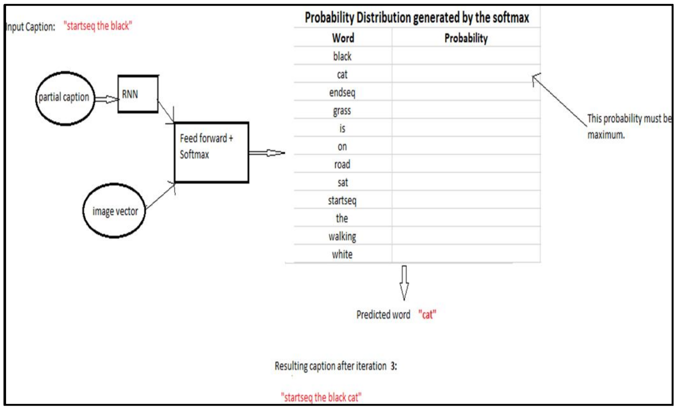


Figure 5 iteration 2

3) Iteration 3

Input: Image vector + “startseq the black” Expected Output word: “cat”

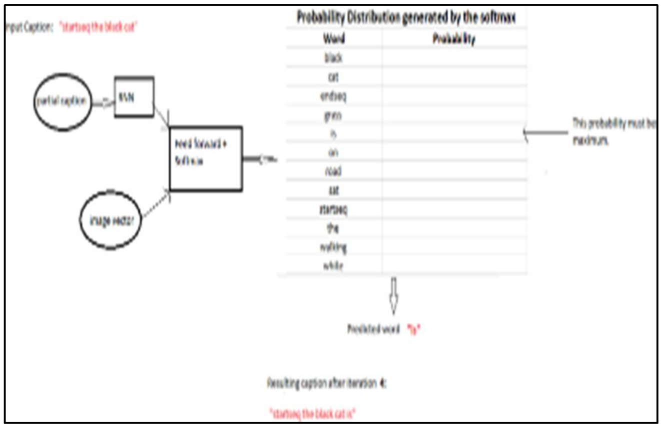

Input Caption: "startseq the black"
partial caption
RNN
Feed forward + Softmax
image vector
Probability Distribution generated by the softmax
This probability must be maximum.
Predicted word "cat"
Resulting caption after iteration 3:
"startseq the black cat"

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure. 6 iteration 3

4) Iteration 4

Input: Image vector + “startseq the black cat” Expected Output word: “is”

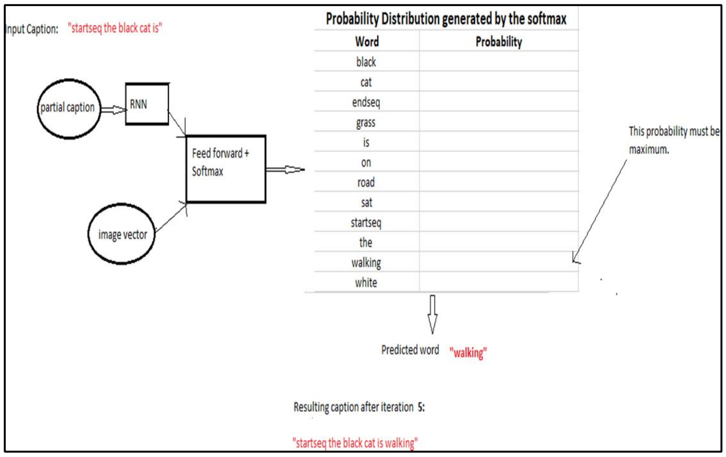

Input Caption: "startseq the black cat"
partial caption
RNN
Feed forward + Softmax
image vector
Probability Distribution generated by the softmax
This probability must be maximum.
Predicted word "is"
Resulting caption after iteration 4:
"startseq the black cat is"

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 7 Iteration 4

5) Iteration 5

Input: Image vector + “startseq the black cat is” Expected Output word: “walking”

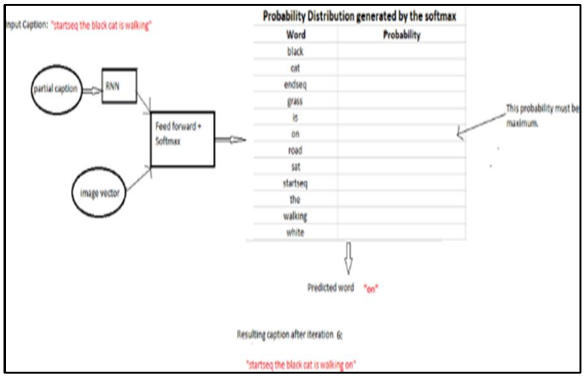

Input Caption: "startseq the black cat is"
partial caption
RNN
Feed forward + Softmax
image vector
Probability Distribution generated by the softmax
This probability must be maximum.
Predicted word "walking"
Resulting caption after iteration 5:
"startseq the black cat is walking"

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 8 iteration 5

6) Iteration 6

Input: Image vector + “startseq the black cat is walking” Expected Output word: “on”

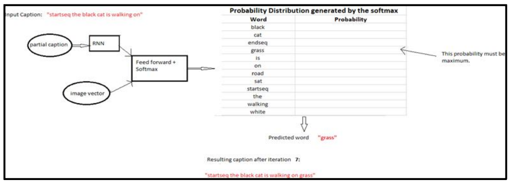

Input Caption: "startseq the black cat is walking"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "on"
Resulting caption after iteration 6:
"startseq the black cat is walking on"

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 9 Iteration 6

7) Iteration 7

Input: Image vector + “startseq the black cat is walking on” Expected Output word: “grass”

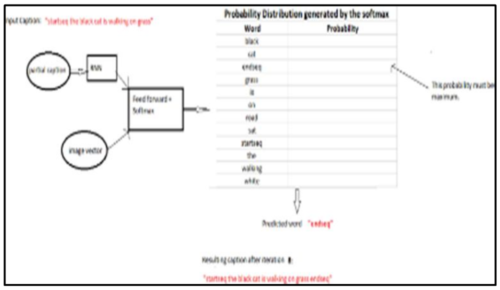

Input Caption: "startseq the black cat is walking on"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "grass"
Resulting caption after iteration 7:
"startseq the black cat is walking on grass"

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure . 10 Iteration 7

8) Iteration 8

Input: Image vector + “startseq the black cat is walking on grass” Expected Output word: “endseq”


Input Caption: "startseq the black cat is walking on grass"
partial caption
RNN
Feed forward + Softmax
image vector
This probability must be maximum.
Predicted word "endseq"
Resulting caption after iteration 8:
"startseq the black cat is walking on grass endseq"

Word	Probability
black	
cat	
endseq	
grass	
is	
on	
road	
sat	
startseq	
the	
walking	
white	

Figure 11 iteration 8

The iterations come to a standstill at this point.

As a result, when one of the following two requirements is satisfied, it comes to a halt:

- It encounters a 'endseq' token, which denotes that the model believes the caption ends here. (By now, you should understand what the 'endseq' symbol means.)
- The number of words generated by the model is approaching a limit.

IV. SYSTEM ARCHITECTURE

A. Dataset

With 8,000 photographs linked with five separate captions that give comprehensive descriptions of the essential aspects and events, a new benchmark collection for sentence-based picture description and search has been created. … The photographs were chosen from six different Flickr groups and don't contain any well-known people or places, but they do depict a variety of events and circumstances.

When it comes to photo captioning, the Flickr8K dataset is a great place to start.

```
image = load_img('990890291_afc72be141.jpg', target_size=(224, 224))
```

The rationale for this is that it is realistic and small enough to download and utilize on a workstation with a CPU to generate models. "Framing Image Description as a Ranking Task: Data, Models, and Evaluation Metrics," a 2013 paper, describes the dataset in detail.

The dataset is described as follows by the authors:

We provide a new benchmark collection for sentence-based picture description and search, consisting of 8,000 photographs paired with five separate captions that provide clear descriptions of the main entities and events.

The photographs were hand-picked from six different Flickr groups and don't contain any well-known people or places, but they do depict a variety of events and situations...

- Flickr8k_Dataset.zip (1 Gigabyte) An archive of all photographs.
- Flickr8k_text.zip (2.2 Megabytes) An archive of all text descriptions for photographs.

B. How to Prepare a Photo Caption Dataset for Training a Deep Learning Model

A model is given an image and is asked to write a human-readable description.

It's a demanding artificial intelligence problem that calls for both image interpretation and text generation from the domains of computer vision and natural language processing.

You may now develop your own picture caption models using deep learning and publicly available datasets of photos and their descriptions.

1) Learn about the Flickr8K dataset, which contains over 8,000 pictures with up to five descriptions per snap, after completing this course.
2) Importing and preparing picture and text data for deep learning models in general.
3) How to store data for two different types of deep learning models in Keras.

Start with my new book, Deep Learning for Natural Language Processing, which contains step-by-step instructions and Python source code files for all scenarios.

Let's get going.

- *November 2017 update:* Minor mistakes in the code in the "Whole Description Sequence Model" section were fixed. Thank you, Matthew and Moustapha Cheikh.
- *February 2019 update:* Direct connections to the Flickr8k Dataset dataset were added when the official site was taken down.

C. Python Environment

This course requires that you have the Python 3 SciPy environment installed. Python 2 can be used, however some of the examples may need to be modified.

Keras 2.0 or above is required, as well as the TensorFlow or Theano runtimes.

You must also have scikit-learn, Pandas, NumPy, and Matplotlib installed in order to follow the lesson.

D. How to Load Photographs

In this section, we will develop some code to load the photos for use with the Keras deep learning library in Python.

The image file names are unique image identifiers. For example, here is a sample of image file names:

```
990890291_afc72be141.jpg
99171998_7cc800ceef.jpg
99679241_adc853a5c0.jpg
997338199_7343367d7f.jpg
997722733_0cb5439472.jpg
```

Keras provides the *load_img()* function that can be used to load the image files directly as an array of pixels.

```
from keras.preprocessing.image import load_img
image = load_img('990890291_afc72be141.jpg')
```

The pixel data needs to be converted to a NumPy array for use in Keras.

We can use the *img_to_array()* keras function to convert the loaded data.

```
from keras.preprocessing.image import img_to_array
image = img_to_array(image)
```

We may want to use a pre-defined feature extraction model, such as a state-of-the-art deep image classification network trained on Image net. The Oxford Visual Geometry Group (VGG) model is popular for this purpose and is available in Keras.

The Oxford Visual Geometry Group (VGG) model is popular for this purpose and is available in Keras.

If we decide to use this pre-trained model as a feature extractor in our model, we can preprocess the pixel data for the model by using the *preprocess_input()* function in Keras, for example:

```
from keras.applications.vgg16 import
preprocess_input

# reshape data into a single sample of an image
image = image.reshape((1, image.shape[0],
image.shape[1], image.shape[2]))
 # prepare the image for the VGG model
 image = preprocess_input(image)
```

We may also want to force the loading of the photo to have the same pixel dimensions as the VGG model, which are 224 x 224 pixels. We can do that in the call to *load_img()*,

We may wish to obtain the picture filename's unique image identification. This may be accomplished by separating the filename string by the '.' (period) character and getting the first member of the resultant array:

```
filename.split('.') image id[0]
```

We can tie everything together by writing a function that will load and pre-process all of the images for the VGG model and return them in a dictionary keyed on their unique image identifiers, given the name of the directory holding the photos..

It takes a few minutes to run.

```
Loaded Images: 8091
```

If you do not have the RAM to hold all images (about 5GB by my estimation), then you can add an if-statement to break the loop early after 100 images have been loaded, for example:

```
if (len(images) >= 100):
	break
```

E. Pre-Calculate Photo Features

It is possible to use a pre-trained model to extract the features from photos in the dataset and store the features to file.

This efficiency implies that the language element of the model, which converts picture characteristics into textual descriptions, may be trained separately from the feature extraction model. The advantage is that the extremely large pre-trained models do not need to be loaded, stored in memory, and used to process each photo while the language model is being trained.

The feature extraction model and the language model may then be combined to make predictions on fresh photographs..

We'll expand the previous part's photo loading approach to load all photographs, extract their aspects using a pre-trained VGG modelling, and save the extracted features to a new file that can be loaded and used to train the language model in this section.

The VGG model must be loaded first. This model is included in Keras and may be loaded in the following way. Note that this will take a few minutes to download the 500-megabyte model weights to your PC.

```
from keras.applications.vgg16 import VGG16
# load the model
in_layer = Input(shape=(224, 224, 3))
model = VGG16(include_top=False, input_tensor=in_layer, pooling='avg')
print(model.summary())
```

This will load the VGG 16-layer model.

Include top=False removes the two Dense output layers as well as the categorisation output layer from the model. The output of the last pooling layer is used to extract the image's features.

Then, like in the previous section, we can go through all of the photos in the directory of images and apply the predict() method on the model for each prepared image to acquire the extracted features. The features may then be saved in a dictionary with the picture id as the key..
The complete example is listed below.

V. SIMULATION AND RESULTS

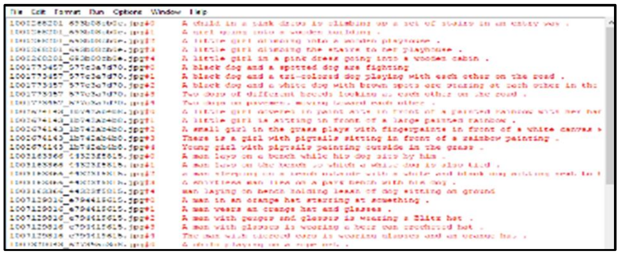
A. First, we Import all the Necessary Packages

```
import string
import numpy as np
from PIL import Image
import os
from pickle import dump, load
import numpy as np
from keras.applications.xception import Xception, preprocess_input
from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.layers.merge import add
from keras.models import Model, load_model
from keras.layers import Input, Dense, LSTM, Embedding, Dropout
# small library for seeing the progress of loops.
from tqdm import tqdm_notebook as tqdm
tqdm().pandas()
```

B. Getting and Performing data Cleaning

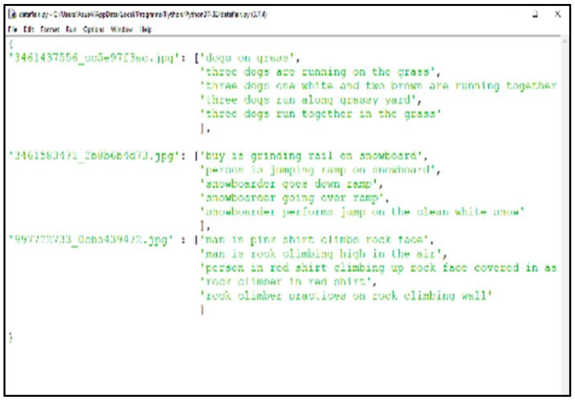
Flickr8k.token is the primary text file in our Flickr 8k text folder that contains all image descriptions.

1) Take a look at the Document. –



```
1000268201_693b08cb0e.jpg#0     A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg#1     A girl going into a wooden building .
1000268201_693b08cb0e.jpg#2     A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg#3     A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg#4     A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg#0     A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg#1     A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg#2     A black dog and a white dog with brown spots are staring at each other in the
1001773457_577c3a7d70.jpg#3     Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg#4     Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg#0     A little girl covered in paint sits in front of a painted rainbow with her har
1002674143_1b742ab4b8.jpg#1     A little girl is sitting in front of a large painted rainbow .
1002674143_1b742ab4b8.jpg#2     A small girl in the grass plays with fingerpaints in front of a white canvas w
1002674143_1b742ab4b8.jpg#3     There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_1b742ab4b8.jpg#4     Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg#0     A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg#1     A man lays on the bench to which a white dog is also tied .
1003163366_44323f5815.jpg#2     a man sleeping on a bench outside with a white and black dog sitting next to h
1003163366_44323f5815.jpg#3     A shirtless man lies on a park bench with his dog .
1003163366_44323f5815.jpg#4     man laying on bench holding leash of dog sitting on ground
1007129816_e794419615.jpg#0     A man in an orange hat starring at something .
1007129816_e794419615.jpg#1     A man wears an orange hat and glasses .
1007129816_e794419615.jpg#2     A man with gauges and glasses is wearing a Blitz hat .
1007129816_e794419615.jpg#3     A man with glasses is wearing a beer can crocheted hat .
1007129816_e794419615.jpg#4     The man with pierced ears is wearing glasses and an orange hat .
1007320043_627395c3d8.jpg#0     A child playing on a rope net .
```

Figure 13 expected output

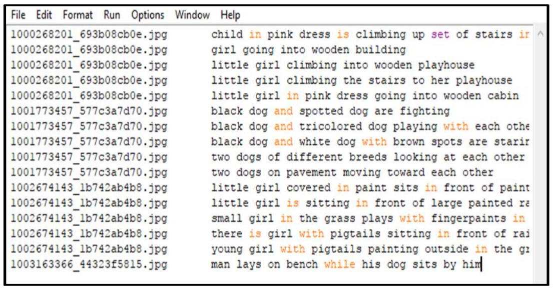

datafile.py - C:\Users\Asus\AppData\Local\Programs\Python\Python37-32\datafile.py (3.7.4)
File Edit Format Run Options Window Help

```
{
'3461437556_cc5e97f3ac.jpg': ['dogs on grass',
                              'three dogs are running on the grass',
                              'three dogs one white and two brown are running together
                              'three dogs run along grassy yard',
                              'three dogs run together in the grass'
                              ],

'3461583471_2b8b6b4d73.jpg': ['buy is grinding rail on snowboard',
                              'person is jumping ramp on snowboard',
                              'snowboarder goes down ramp',
                              'snowboarder going over ramp',
                              'snowboarder performs jump on the clean white snow'
                              ],
'997722733_0cb5439472.jpg' : ['man in pink shirt climbs rock face',
                              'man is rock climbing high in the air',
                              'person in red shirt climbing up rock face covered in as
                              'rock climber in red shirt',
                              'rock climber practices on rock climbing wall'
                              ]


}
```

2) *Cleaning Text (descriptions)* – This function clears out all of the descriptions. When coping with data sets, this is a critical stage when we choose the type of text cleaning we want to do based on our objectives. In our example, we'll remove punctuation, convert all text to lowercase, and delete phrases that contain digits. Therefore, captions such as "A guy in a three-wheeled wheelchair" will be renamed "man in three-wheeled mobility scooter."
3) *Text Vocabulary (Descriptions)* – This is a fundamental function that produces the vocabulary by separating all lexical aspects from all explanations.
4) *Save Descriptions (Descriptions, Filename)* – This method will save a list of all pre-processed summaries to a file. All of the captions will be saved in a descriptions.txt file. In looks, it will approach this.



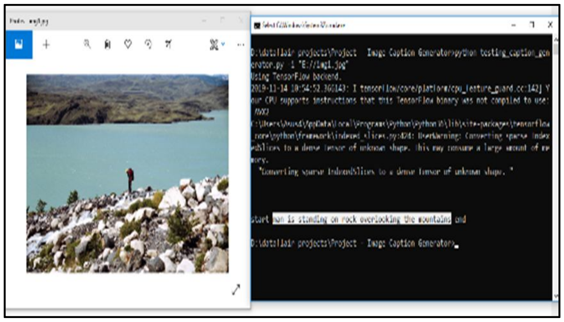
```
1000268201_693b08cb0e.jpg       child in pink dress is climbing up set of stairs ir
1000268201_693b08cb0e.jpg       girl going into wooden building
1000268201_693b08cb0e.jpg       little girl climbing into wooden playhouse
1000268201_693b08cb0e.jpg       little girl climbing the stairs to her playhouse
1000268201_693b08cb0e.jpg       little girl in pink dress going into wooden cabin
1001773457_577c3a7d70.jpg       black dog and spotted dog are fighting
1001773457_577c3a7d70.jpg       black dog and tricolored dog playing with each othe
1001773457_577c3a7d70.jpg       black dog and white dog with brown spots are starir
1001773457_577c3a7d70.jpg       two dogs of different breeds looking at each other
1001773457_577c3a7d70.jpg       two dogs on pavement moving toward each other
1002674143_1b742ab4b8.jpg       little girl covered in paint sits in front of paint
1002674143_1b742ab4b8.jpg       little girl is sitting in front of large painted ra
1002674143_1b742ab4b8.jpg       small girl in the grass plays with fingerpaints in
1002674143_1b742ab4b8.jpg       there is girl with pigtails sitting in front of rai
1002674143_1b742ab4b8.jpg       young girl with pigtails painting outside in the gr
1003163366_44323f5815.jpg       man lays on bench while his dog sits by him
```

C. *Extracting the Feature Vector from all Images*

Because we don't have to do everything ourselves, this method is also known as transfer learning. Instead, we use pre-trained frameworks that have been tested on large amounts of data to extract features and reapply them to our tasks. The Xception model was trained on an imagenet dataset containing 1000 discrete classes to sift through. The keras.applications package includes this prototype, which can be readily imported. Make sure you have an internet connection because the weights are downloaded instantly. Because the Xception model was originally designed for Imagenet, we will make small tweaks to it for interaction with our model. Keep in mind that the Xception model requires a picture with dimensions of 299*299*3.. We'll acquire the 2048 feature vector by removing the last classification layer.

include top=False, pooling='avg') model = Xception(include top=False, pooling='avg')

The function extract features () will extract features for all photos, and image identifiers will be mapped to their corresponding feature arrays. The features dictionary will then be dumped into a "features.p" pickle file.

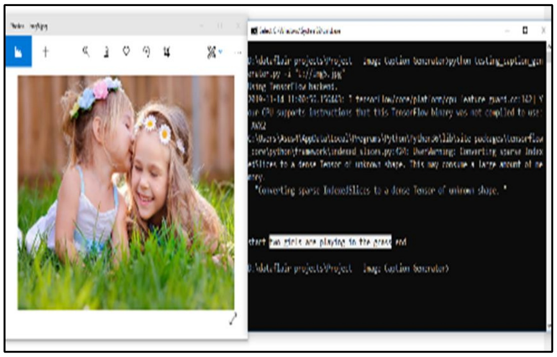


```
D:\dataflair projects\Project - Image Caption Generator>python testing_caption_gen
erator.py -i "E://img1.jpg"
Using TensorFlow backend.
2019-11-14 10:54:52.360143: I tensorflow/core/platform/cpu_feature_guard.cc:142] Y
our CPU supports instructions that this TensorFlow binary was not compiled to use:
 AVX2
C:\Users\[illegible]\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow
_core\python\framework\indexed_slices.py:424: UserWarning: Converting sparse Index
edSlices to a dense Tensor of unknown shape. This may consume a large amount of me
mory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "



start man is standing on rock overlooking the mountains end

D:\dataflair projects\Project - Image Caption Generator>
```

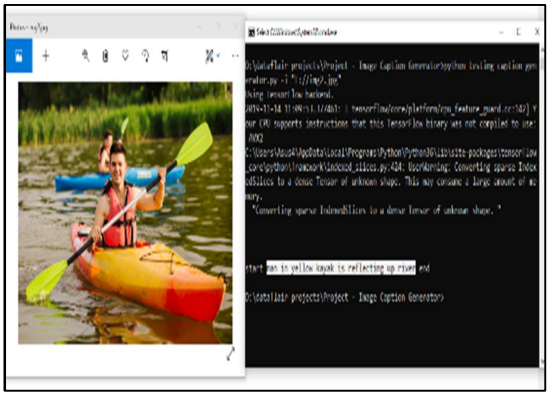


```
D:\dataflair projects\Project - Image Caption Generator>python testing_caption_gen
erator.py -i "E://img5.jpg"
Using TensorFlow backend.
2019-11-14 11:00:[illegible]: I tensorflow/core/platform/cpu_feature_guard.cc:142] Y
our CPU supports instructions that this TensorFlow binary was not compiled to use:
 AVX2
C:\Users\[illegible]\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow
_core\python\framework\indexed_slices.py:424: UserWarning: Converting sparse Index
edSlices to a dense Tensor of unknown shape. This may consume a large amount of me
mory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "



start two girls are playing in the grass end

D:\dataflair projects\Project - Image Caption Generator>
```



```
D:\dataflair projects\Project - Image Caption Generator>python testing caption gen
erator.py -i "1://img2.jpg"
Using TensorFlow backend.
2019-11-14 11:09:51.172461: I tensorflow/core/platform/cpu_feature_guard.cc:142] Y
our CPU supports instructions that this TensorFlow binary was not compiled to use:
 AVX2
C:\Users\Asus4\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow
_core\python\framework\indexed_slices.py:424: UserWarning: Converting sparse Index
edSlices to a dense Tensor of unknown shape. This may consume a large amount of me
mory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "



start man in yellow kayak is reflecting up river end

D:\dataflair projects\Project - Image Caption Generator>
```

VI. CONCLUSION

We utilize human consensus scores to assess the accuracy. Human participants are given a triplet of descriptors, one reference and two candidate descriptions, and asked to pick the proposed description that most closely resembles the reference. A metric is valid if it provides the human subject's choice of description a higher score for being more like the reference caption. This study focuses mostly on text captioning, according to academic articles. Several captioning metrics are used to evaluate the machine-generated sentences. The scores reflect the accuracy with which the words were written. According to our findings, the LSTM method is efficient. This produces the best results on the Flicker 8K Dataset. There may be certain limitations on the output, such as a section having no more than 500 words or 4-5 lines. As a consequence, this article demonstrates how to build a paragraph from a photograph. The research is limited to the LSTM approach. The project's scope might be broadened in the future to allow all scholars to make better use of the system.

In this complicated Python project, we developed an image caption generator to generate a CNN-RNN model. It's vital to note that because our model is based on data, it won't be able to predict concepts that aren't in its lexicon. We have a small dataset of 8000 images to work with. In order to generate superior accuracy models, we need to train production-level models on datasets larger than 100,000 images

REFERENCES

[1] Marc‘AurelioRanzato, Sumit Chopra, Michael Auli,andWojciechZaremba. 2015.‖ Sequence level training with recurrent neural networks‖arXiv preprintarXiv:1511.06732.

[2] Peter Anderson, Xiaodong He, Chris Buehler, DamienTeney, Mark Johnson, Stephen Gould, and LeiZhang. 2017.‖ Bottom-up and top-down attention for image captioning and vqa‖. arXiv preprintarXiv:1707.07998.

[3] Steven J Rennie, Etienne Marcheret, Youssef Mroueh,Jarret Ross, and VaibhavaGoel. 2016‖. Self-critical sequence training for image captioning.‖

[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, JamesHays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence Zitnick. 2014.‖ Microsoft coco:Common objects in context.‖ In European conference on computer vision, pages 740–755. Springer.

[5] Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan,and Eric P Xing. 2017. ―Recurrent topic-transition for visual paragraph generation.‖

[6] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young,C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: ―Generating sentences from images.‖ InECCV,2010.

[7] Karpathy and L. Fei-Fei.‖ Deep visual-semantic generating image descriptions.‖InCVPR,2015.

[8] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg,and T. L. Berg. Baby talk: ―Understanding and generating image descriptions.‖InCVPR, 2011.

[9] J. Donahue,L. Anne Hendricks,S. Guadarrama,M. Rohrbach, S. Venugopalan, K. Saenko, and T.Darrell‖Long-term recurrent convolutional networks for and description‖. InCVPR, 2015.

[10] Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis,and Erkut Erdem. 2016. Re-evaluating automaticmetrics for image captioning.arXiv preprintarXiv:1612.07600.

[11] Oriol Vinyals, Alexander Toshev, Samy Bengio, andDumitru Erhan. 2014. Show and tell: A neural im-age caption generator.CoRR, abs/1411.4555.

[12] Ruotian Luo, Brian L. Price, Scott Cohen, and Gregory Shakhnarovich. 2018. Discriminability objective for training descriptive captions. CoRR,abs/1803.04376.

[13] Steven J Rennie,Etienne Marcheret,Youssef Mroueh,Jarret Ross,and Vaibhava Goel.2016“Self Critical Sequence Training for Image Captioning”.

[14] Tsung-Yi Lin,Michael Maire,Serge Belongie,James Hays,Pietro Peronal Deva Ramanan,Piotr Dollar, and C Larence Zitnick.2014, “Micrsococo:Common Objects in ConteAt”In European conference on computer vision pages 740-755.Springer.

[15] Xiaodan Liang,Zhiting Hu,Hao Zhang,Chuang Ggan ,and Eric P Xing.2017. “Recurrent Topic-Transition for Visual Paragraph Generation