Data Mining Project

Image Caption Generator Using CNN and RNN(LSTM).

Content

Introduction

1.The Purpose of the project is to generate Image Caption of images.

2.Image captioning aims at generating captions of an image automatically using deep learning techniques such as CNN and RNN(LSTM).

Datset Used

- **Flicker8k_Dataset –** Dataset folder which contains 8091 images.

- **Flickr_8k_text –** Dataset folder which contains text files and captions of images.

Performing data cleaning

- First we import necessary packages like numpy,os,tokenizer,etc which we will use for building our model

- Cleaning the dataset-Our file flicker8k.tokens contains image name and it’s description seperated by a line(\n),but description lines contain many unwanted characters like numbers and ‘-” symbols,etc,which are required to be clean.So we will remove numbers,punctuations and symbols from description and store the updated description in new file

What is CNN?

- Convolutional Neural networks are specialized deep neural networks which can process the data that has input shape like a 2D matrix. Images are easily represented as a 2D matrix and CNN is very useful in working with images.

- It scans images from left to right and top to bottom to pull out important features from the image and combines the feature to classify images. It can handle the images that have been translated, rotated, scaled and changes in perspective.

Extracting feature vector from images

- This technique is called transfer learning.In this,we use the pre-trained model that have been already trained on large datasets and extract the features from these models and use them for our tasks.
- We are using the Xception model which has been trained on imagenet dataset that had 1000 different classes to classify.
- Xception model takes 299*299*3 image size as input so we will remove the last classification layer and get the 2048 feature vector.
- The function **extract_features()** will extract features for all images and we will map image names with their respective feature array. Then we will dump the features dictionary into a “features.p” pickle file.

What is LSTM?

- LSTM means **Long short term memory**, they are a type of RNN (**recurrent neural network**) which is well suited for sequence prediction problems
- Based on the previous text, we can predict what the next word will be. It has proven itself effective from the traditional RNN by overcoming the limitations of RNN which had short term memory
- LSTM can carry out relevant information throughout the processing of inputs and with a forget gate, it discards non-relevant information

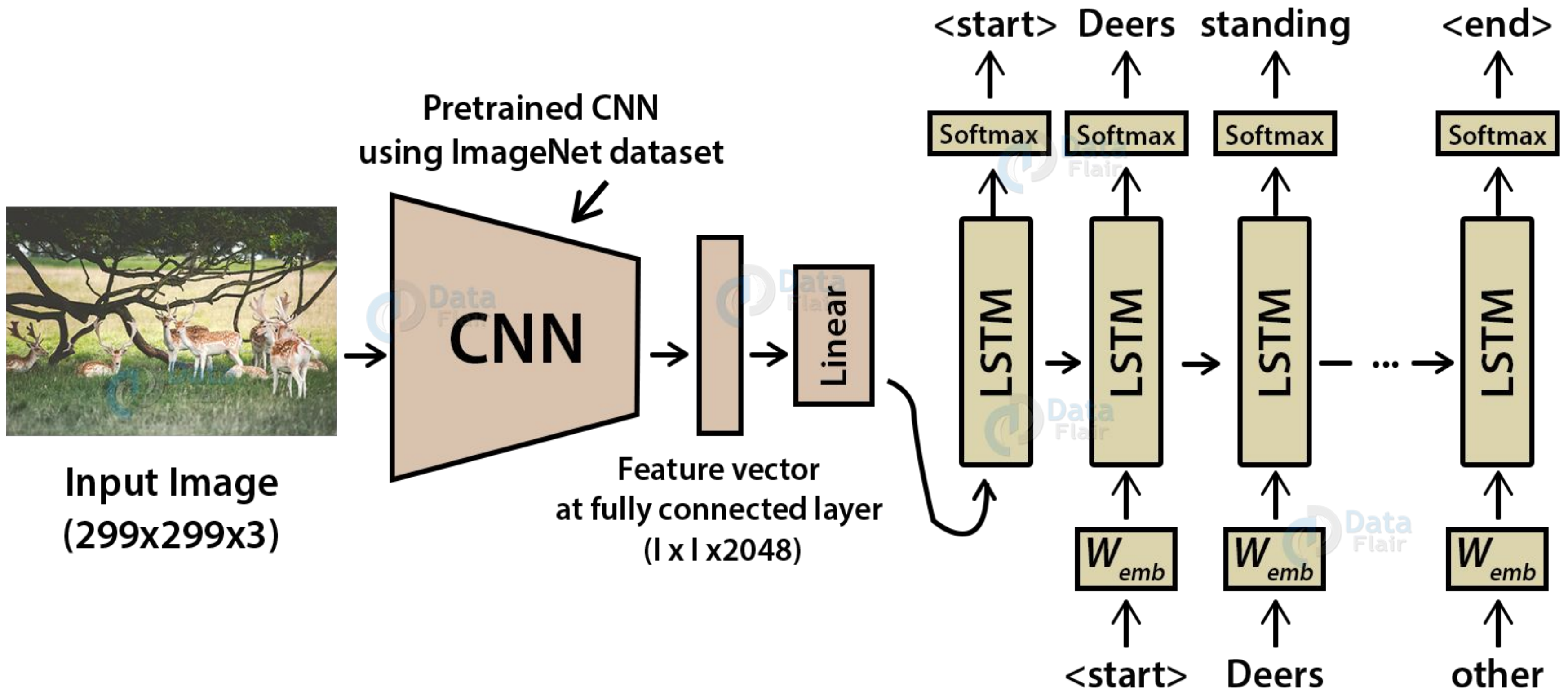
Our Approach

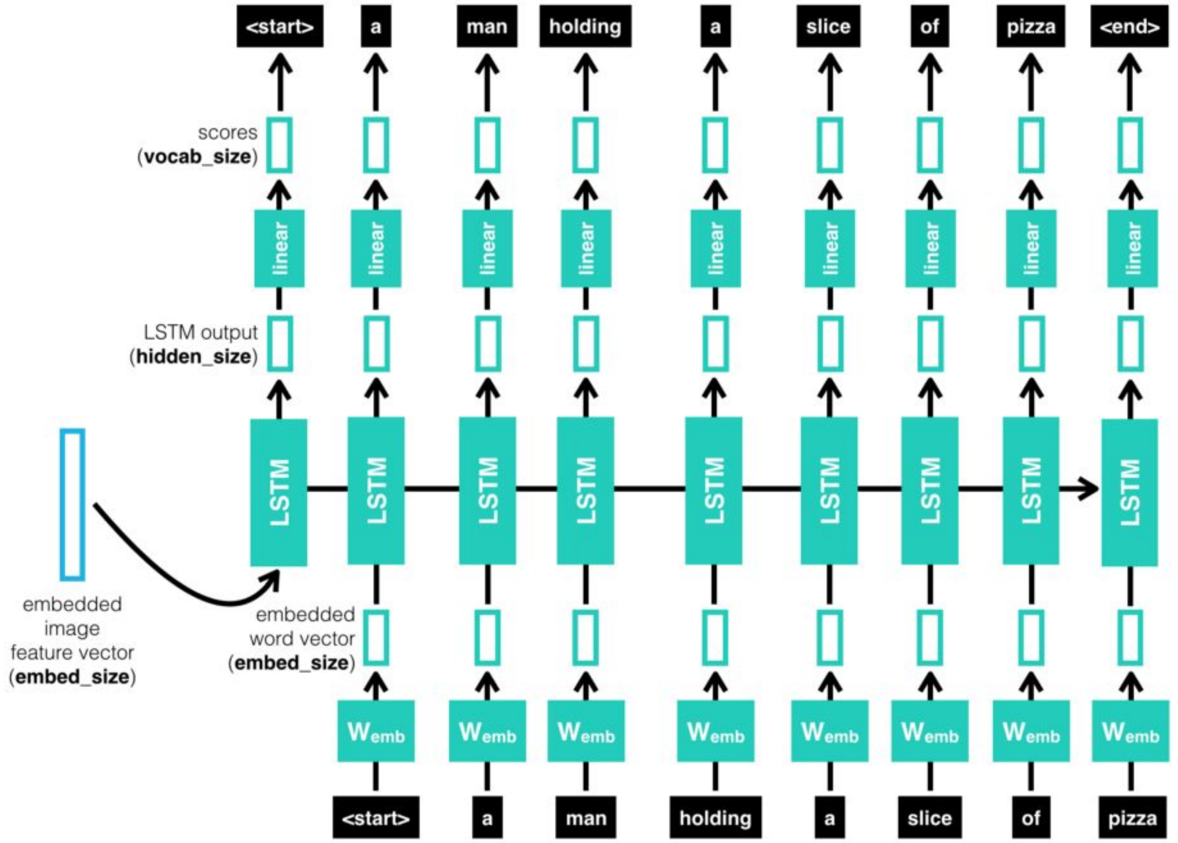
To implement our image caption generator model,we will combine CNN AND LSTM architecture

- CNN will be used for extracting feature of image.We will use the pre-trained model Xception.

- Then LSTM will use the features extracted by CNN to generate description of image for our Image caption generator model

Model - Image Caption Generator





<start>
a
man
holding
a
slice
of
pizza
<end>
scores
(**vocab_size**)
linear
LSTM output
(**hidden_size**)
LSTM
embedded
image
feature vector
(**embed_size**)
embedded
word vector
(**embed_size**)
W_{emb}
<start>
a
man
holding
a
slice
of
pizza

Loading dataset for training model

In our **Flickr_8k_test** folder, we have **Flickr_8k.trainImages.txt** file that contains a list of 6000 image names that we will use for training.

For loading the training dataset, we need more functions:

- **load_photos(filename) –** This will load the text file in a string and will return the list of image names.
- **load_clean_descriptions(filename, photos) –** This function will create a dictionary that contains captions for each photo from the list of photos. We also append the <start> and <end> identifier for each caption. We need this so that our LSTM model can identify the starting and ending of the caption.
- **load_features(photos) –** This function will give us the dictionary for image names and their feature vector which we have previously extracted from the Xception model.

Tokenizing the vocabulary

- Computers don’t understand English words, for computers, we will have to represent them with numbers. So, we will map each word of the vocabulary with a unique index value.

- Keras library provides us with the tokenizer function that we will use to create tokens from our vocabulary and save them to a **“tokenizer.p”** pickle file.

Create data generator

We first need to see how the input and output of our model will look like. To make this task into a supervised learning task, we have to provide input and output to the model for training. We have to train our model on 6000 images and each image will contain 2048 length feature vector and caption is also represented as numbers. This amount of data for 6000 images is not possible to hold into memory so we will be using a generator method that will yield batches.

The generator will yield the input and output sequence.

Defining the CNN-RNN model

For defining our model we will use keras model from functional API.It basically consist of three parts-:

1. **Feature Extractor –** The feature extracted from the image has a size of 2048, with a dense layer, we will reduce the dimensions to 256 nodes.
2. **Sequence Processor –** An embedding layer will handle the textual input, followed by the LSTM layer.
3. **Decoder –** By merging the output from the above two layers, we will process by the dense layer to make the final prediction. The final layer will contain the number of nodes equal to our vocabulary size.

Training and testing the model

- To train the model, we will be using the 6000 training images by generating the input and output sequences in batches and fitting them to the model using model.fit_generator() method.

- The model has been trained, now, we will test by loading the model and generate predictions. The predictions contain the max length of index values so we will use the same tokenizer.p pickle file to get the words from their index values.

Example image caption

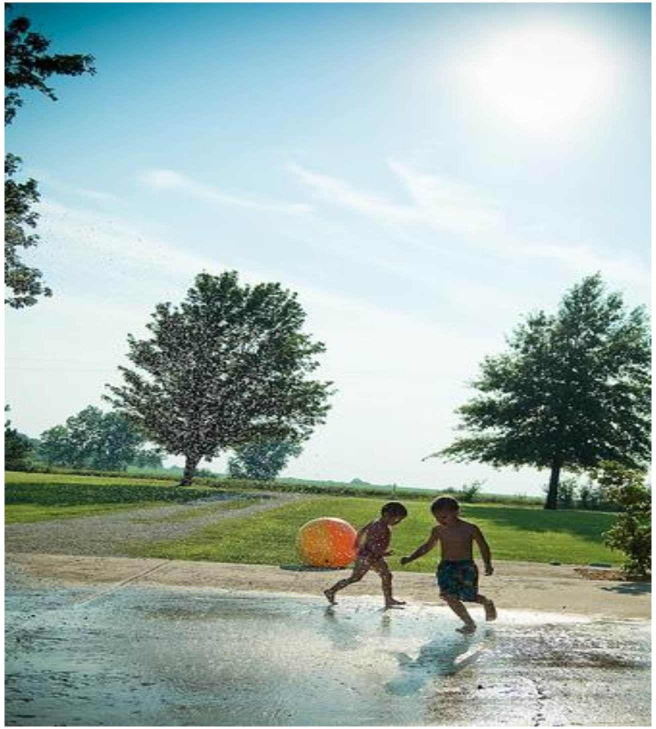


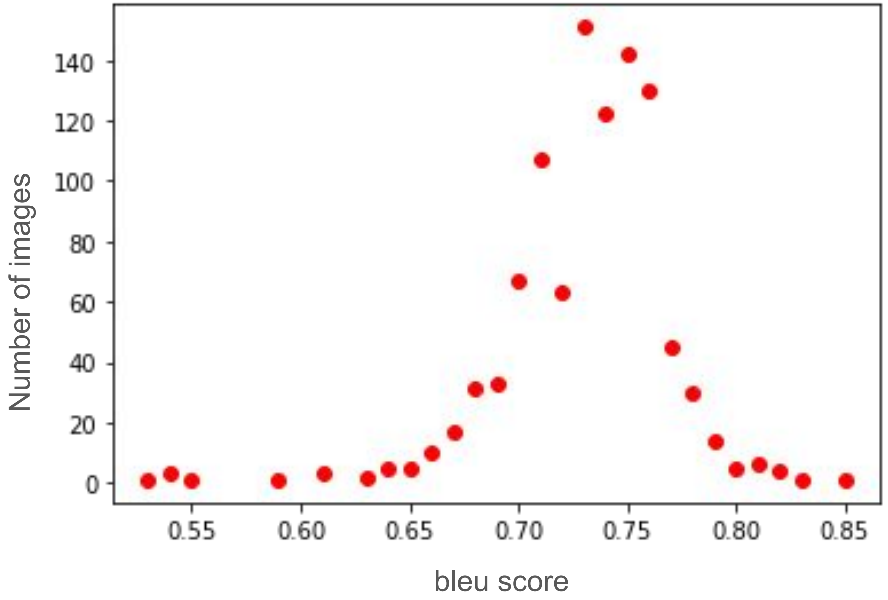
Image Caption :
two girls are sitting on the grass

Testing the model

- For checking the accuracy of the cations produced,we used BLEU score.
- The Python Natural Language Toolkit library, or NLTK, provides an implementation of the BLEU score that one can use to evaluate your generated text against a reference.
- BLEU score generates a number between 0 and 1 where 0 means no match and 1 denotes a complete match.
- ‘reference’ for us are the provided captions in the training dataset.
- We generated captions for 1000 images and calculated average bleu score for them.

Result

- BLEU score came out to be : 0.7316400000000023 after the model was trained on 5 epochs.
- We also visualised the distribution of bleu score in the following plot :



Credit

- Bharat Kumar (18075016)
- Ayush Singh (18075015)
- Asht Bhuja Prasad Rastogi (18075013)
- Aryan Agrawal (18075011)