EE452 Computer Vision (Spring 2020)

Assignment 2

Image Captioning

Release date: Friday, 03 April 2020

Due date: Friday, 24 April 2020

Maximum Marks: 100 (Weight in Final grade: 8%)

Instructions:

- This assignment is individual task where only collaboration in terms of discussing and idea sharing is allowed.
- A report is to be submitted on LMS describing the different phases of the work also including what worked, what did not work, what you learned, what extra things you tried and how to proceed with this etc.

Description: The purpose of this assignment is to train an image captioning model using a manageable dataset. Image captioning refers to the task of generating a short textual description given an image as input. Some examples of image captioning can be seen in Figure 1:

Figure 1: Examples of captions generated by an Image Captioning model [1]

The aim of this assignment is not to design a perfect image captioning model, but to expand our knowledge of deep learning and computer vision by applying it to an immensely exciting task.

Tasks:

1. Download the Flickr8k Image Captioning dataset using the following two links [2]:

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

You can download and unzip the data in your Google colab notebook using the following code:

```
!wget https://github.com/jbrownlee/Datasets/releases/download/Flickr8k/Flickr8k_Dataset.zip
!wget https://github.com/jbrownlee/Datasets/releases/download/Flickr8k/Flickr8k_text.zip
!unzip Flickr8k_Dataset.zip
!unzip Flickr8k_text.zip
```

The relevant files for this assignment are the following: Flickr8k.token.txt, Flickr_8k.trainImages.txt, Flickr_8k.devImages.txt, and the Flicker8k_Dataset folder which contains all of the images. The Flickr_8k.token.txt contains the name of each image followed by "#{digit}" followed by its caption. The digit refers to the fact that for each image there are multiple captions.

2. Our training data for this task will refer to those images mentioned in the "Flickr_8k.trainImages.txt" and their corresponding captions which can be found in the "Flickr8k.token.txt". The input at any particular instant to the network will consist of two components: a representation of the image and a representation of a sequence of words; the job of the network will be to predict the next word given this context.

Each description in the training data would have to be padded with special "START" and "END" tokens because once we provide the network with an input image, we would also provide it with the start token as its only context. The network will then generate the next word, which would then be added to the context sequence, and so on until it reaches the end token or a set maximum length.

In order to represent the images, we can use an approach similar to the transfer learning approach we used in the last assignment i.e. we can use a pre-trained model (such as VGG-16, VGG-19 or ResNet-50), and use one of its fully connected layers (except for the final one) as a feature representation of our image.

In order to represent words, we shall resort to using a vector representation of words. This can be realized by using an embedding layer in our network, which will have a word id to vector correspondence. Now, for the embedding layer (i) we can either train it during our actual training, (ii) we can use pre-trained embeddings such as Word2Vec embeddings, (iii) or we can train a Word2Vec model on our data beforehand, and then utilize those in our model. For (ii) and (iii), one can refer to the fourth and fifth part of the following tutorial:
https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/
Note: Only one of the above methods is required.

All of the preprocessing we do on our training data will also be repeated for our validation data, which can be referred to from "Flickr_8k.devImages.txt".

3. The architecture we would be implementing can be seen in Figure 2:

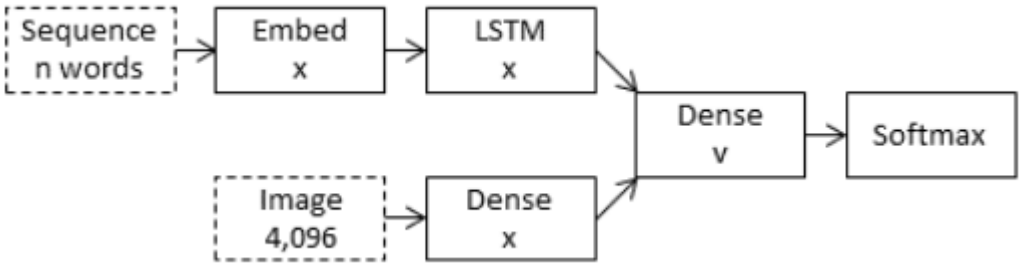

Figure 2: The architecture we shall be following [3]

For every step, a sequence of "n" words is being passed as input. This might appear confusing at first due to the mention of a fixed length sequence, however, this can be achieved by padding all sequences less than that fixed length with some form of zero padding. So for example if the input sequence is "START", the input sequence to the model shall be: {word id for "START"} 0 0 0 0 . . . 0, where 0 is repeated "n-1" times. Similarly, if the input sequence is "START the", the input sequence to the model shall be: {word id for "START} {word id for "the"} 0 0 0 0 . . . 0, where 0 is repeated "n-2" times. Also, while the image representation in the figure is mentioned to be 4096, we can always change it depending on our pre-trained image classification model.

While training, we can use our validation data to choose the best model based on an appropriate criterion.

4. Once we have a trained model, it can be used to generate captions for unknown images. In this part, some examples of generated captions along with their corresponding input images need to be provided.

References:

[1] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
[2] Jbrownlee, "jbrownlee/Dataset," GitHub. [Online].
Available: https://github.com/jbrownlee/Datasets/releases/.
[3] M. Tanti, A. Gatt, and K. Camilleri, "What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?," *Proceedings of the 10th International Conference on Natural Language Generation*, 2017.

Submission Guidelines:

The entire assignment is to be done as a python notebook. Once you are done, you should upload the notebook to LMS. Create sections within the notebook in order to answer each part. Parts 1 and 4 are distinct while parts 2 and 3 can be done together.

It is preferred that you do this assignment on google colab: https://colab.research.google.com/. Colab allows you to utilize its GPU, which can be accessed as follows: Runtime => Change runtime type => Change Hardware accelerator to GPU. Once you have completed your assignment on colab, you can download it as a notebook as follows: File => Download.ipynb. Also, in case you are doing this assignment in Keras, you might find it helpful to downgrade tensorflow (Keras' backend) as follows:

```
!pip3 install tensorflow==1.14
```

This is because the default version on colab is now tensorflow 2.0 which might lead to unwanted results.