Final Project

Anqi Chen

December 18 2020

1 Introduction

To generate photo captions, generally, we make the Input as an image and transform the visual information into text captions to describe it as an output. It is a popular but difficult topic as it combines computer vision and natural language processing.

The project is using a large volume of image data and corresponding caption text data to train a prediction model. It involves the modules we have discussed in the classes, such as data prepossessing of images and texts, building a model for training, validation and testing, evaluating the results using appropriate solutions, and data visualization of showing data set information and model progress.

The reason why I chose this topic is because of the study of the Seq2Seq model in one of the lectures and also the last assignment. First, I thought of using the Seq2Seq model to solve the translation problem, but it is quite difficult to understand both languages and analyze the predicted results. So, I choose to build a natural network model to generate captions for images in the project.

Different than the assignments I did before, the part of data processing that deals with image data is way more difficult than with texts, as it takes more time and effort to extract the features. The important aspect of generating captions of images can be identified as the 5 Vs characteristics of big data. There is no doubt that all kinds of images can be posted on any social media and news applications in real-time. Besides classifications among those images, automatically image captioning can be a valuable research area that machine learning can do with big data.

2 Possible Solutions

There are several approaches to generate captions automatically by deep learning models. According to Patel and Varier (2020) [6], different encoder-decoder models such as CNN + LSTM and CNN + Transformer architectures are used. Moreover, they even compared experimental results for varying encoder in each architecture with matrix. Vinyal et al.[3] from Google introduced a model called Neural Image Caption (NIC) NIC based on a convolution neural network that encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence.

The paper named Image Captioning: A Comprehensive Survey [7] summaries the captioning methods based on deep neural networks into three categories: retrieval and template based methods, multimodal learning based approaches, and encoder-decoder framework based ways. The most commonly used models like what we discussed before are belonging to encoder-decoder frameworks. We can also get an idea that CNN-RNN framework technique are effective as they use both the characteristics of feature representation in CNN and the ability to deal with input sequences in RNN. However, in this survey [7], two drawbacks are identified. The first one is that equal importance are assigned to each caption and each word. The second drawback is during caption generation objects may not be correctly recognized. To deal with the first drawback, we can add a new layer in the model which is the attention layer. This is a mechanism to avoid equal pay more attention on the related regions like human thinking ways. Like what we learned in the lectures, this is belong to one-to-many situation in Seq2Seq learning model. To deal with the second drawback, Devlin et al. [4] uses Nearest-Neighbors approaches to improve that novel captions generated for automatic image captioning is quite important because if NN images are diverse, the chosen captions are likely to be generic.

Here in the project, I use the CNN-RNN architecture in two models, one is without attention layer and the other is with attention layer. The results showing the one with attention layer showing a better performance to predict correct captions for a large number of images. The ideas are first to extract image features through a pretrained image classification model. We can get image contexts from one of the convolution layers in the model. For example, the layer in vgg16: block5_conv3 (Conv2D) has the representation of original image (14, 14, 512), which means we now have 14 x 14 = 196 small slices for every original image and each with 512 Division features. Use these contexts and LSTM generate possible word one by one, we can generate a complete description for each image. When generate every single word, we can lay particular stress and different attention among 196 blocks. The implementation steps can be found in code and the next section.

3 Implementation

3.1 Whole Process

The whole structure of the project is formed as below. The input images are put into a Convolutional Neural Network model to extract the features. This is the first part of the implementation as shown below.

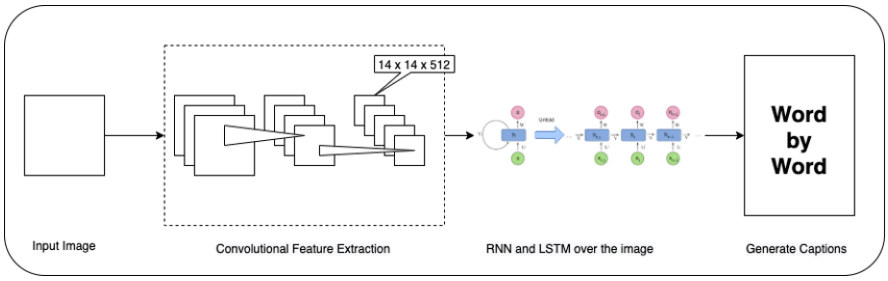

Figure 1: Basic Structure of Image Captioning

To prepare text data for our model, we need to clean it as in the prepossessing step and then summarize the size of vocabulary in the whole set. This step is quite similar to what we did in previous assignments.

Both the input models use some mechanisms like dropout to reduce overfitting in the training dataset.

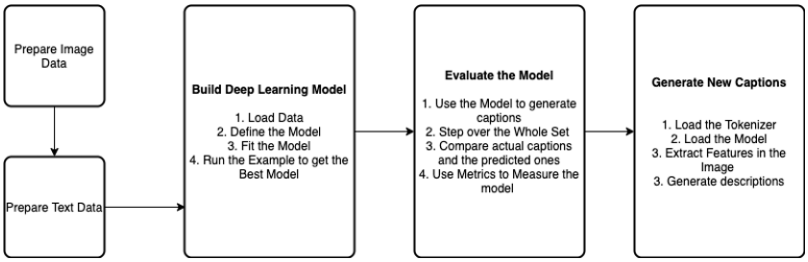

Figure 2: The Flow Chart of Steps

3.2 Model Details

The model building step can be divided into main parts:

Feature Extractor - To be more specific in this part, VGG16 and VGG19 are used in two models in the project, can be loaded in Keras using the VGG class. The last layer in each model is removed as we do not need the classification for those images. The features of images are more important in the project. The

image features are extracted as a one-dimensional 4,096 element vector. These elements are then processed by a Dense layer to generate a 256-element vector for the photo.

Text Sequence Processor - This is to process the sequence of text input by word embedding and followed by a LSTM RNN layer. The input sequences went through the embedding layer to ignore padded values and processed with an LSTM layer to general 256 vectors which is the same number of units as images.

Decoder - The decoder part merges the vectors from two input models and fed to a dense layer to make a softmax prediction over the entire vocabulary for the next word in the description of images.

The information of model1 is plotted as below to better visualize the network structure. It shows how two streams of input data be processed and put together in the training model.

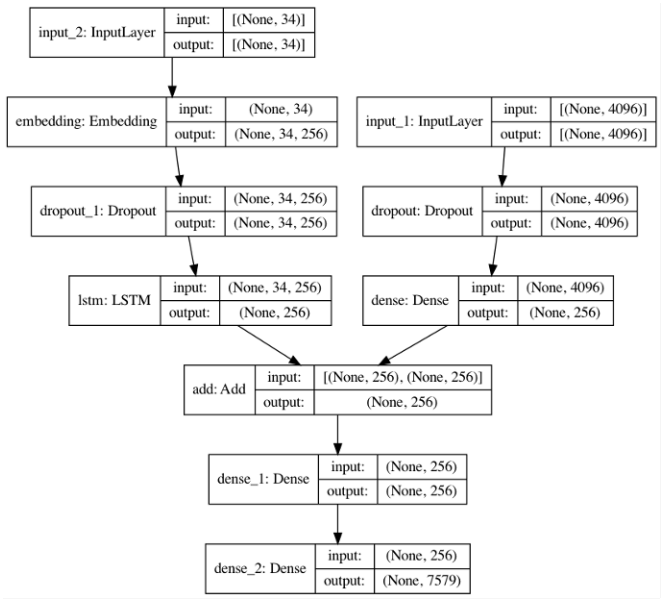

Figure 3: Flow and Structure of the First Model

While training the models, we write the loss into events, so that after training, historical training data can be plotted on the tensorboard. Since model 1 uses early stopping, it only has 3 or 4 epochs in each training process. We saved all the models along the way and remember the best one with the lowest loss value

for later evaluation. The model 2 runs for whole 20 epochs during training and took almost 10 hours.

The loss value changes for both model:

Figure 4: Loss Changes in Model 1

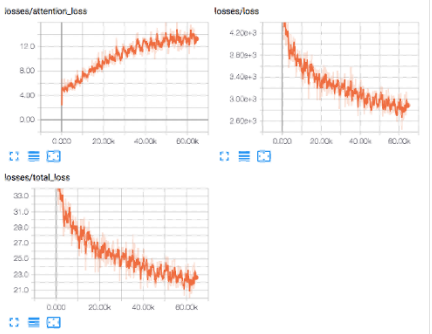

Figure 5: Loss Changes in Model 2

3.3 Cross Validation

The sample image shown below is from the Interne, not in our training datasets. It is used to test the model prediction as the final step.

Figure 6: Sample Image Used in Both Model

The caption generated by model 1 is shown as below:

```
startseq two dogs are running through the grass endseq
```

Figure 7: Caption generated by Model 1

The caption generated by model 2 is shown as below:

```
a brown and white dog running in the grass .
```

Figure 8: Caption generated by Model 2 (part 1)

Figure 9: Caption generated by Model 2 (part 2)

With the same testing sample image, we can see both models made the prediction similar to the ground truth, but compared to model 2, the caption generated by model 1 is more generic.

4 Comparisons

4.1 Dataset

I have explored with two datasets, one is MSCOCO dataset[4], the training dataset contains more than 80,000 images and validation dataset has more than 40,000 images, and each image also has more than one corresponding captions. The other one is Flickr dataset[2], which is a commonly used dataset in image captioning area. I choose these two datasets for experimentation as they are stable and well-prepared for image captioning. The datasets contain both images and text in separate folders and even already be divided into training, validation and testing.

Flickr8k_Dataset: Contains 8000 images in JPEG format.

Flickr8k_text: Contains different sources of descriptions for the images.

The dataset has a pre-defined training dataset (6,000 images), validation dataset (1,000 images), and test dataset (1,000 images).

4.2 Evaluation

BLEU[5]is used as an effective way to measure the similarity between a set of reference texts and the model generated descriptions through the use of n-grams. The output of BLEU is the geometric mean of n-gram score with the brevity penalty to discourage shorter translation. I also use the library[1], a caption evaluation tool, to evaluate ROUGE_L and CIDEr for model 2. These schemas are commonly used for automatically score the correlation between generated text and original text, based on different attributes in words.

Model	**Model1(Without Attention)**	**Model2(Attention)**
BLEU-1	0.428795	0.687801
BLEU-2	0.227460	0.479960
BLEU-3	0.152939	0.334669
BLEU-4	0.064423	0.235521
ROUGE_L	/	0.530459
CIDEr	/	0.729266

As the table above demonstrate, we can see that Model 2 has a better performance and more similar to reference texts so that we can say model 2 is better for image caption generation.

5 Conclusion

5.1 Results

In conclusion, the image captioning problem is effectively solved by two models used in the project and model 2 is better than model 1 by adding an attention layer in the model over input images. In this project, we implemented and compared two models. It is obvious the one using RNN with attention layer over images are better than the one without attention for several reasons. First of all, the loss is more trainable. Then, the sample prediction of model 2 is more specific than that in model 1. Most importantly, compared by BLEU, a principle in this area to evaluate the prediction of captions, model 2 is way better than model 1 as shown in the evaluation results table.

5.2 Future Work

As we have seen in previous research, we can try to improve the accuracy of a future model in several ways. First of all, to improve the datasets, we can make

the features in images more diverse, the reference text more similar to those generated by common readers. Then, we can think of the models, like how to optimize the parameters, structures and algorithms used in them. With the datasets big enough, the model should always have a stable performance even using ensemble methods to improve the evaluation results on diverse data.

References

[1] salaniz/pycocoevalcap github. https://github.com/salaniz/pycocoevalcap. Accessed: 2020-12-18.

[2] X. Chen al. Microsoft coco captions: Data collection and evaluation server. *[cs], Apr*, 2015, Dec. 2020.

[3] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artifical Intellegence Research*, 47:853–899, 2013.

[4] 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, 2013.

[5] Kishore Papineni et al. Bleu: a method for automatic evaluation of machine translation. 2002.

[6] A. Patel and A. Varier. Hyperparameter analysis for image captioning. *[cs], Jun*, 2020, Dec. 2020.

[7] H. Sharma, M. Agrahari, S. K. Singh, M. Firoj, and R. K. Mishra. Image captioning: A comprehensive survey. In *2020 International Conference on Power Electronics IoT Applications in Renewable Energy and its Control (PARC*, pages 325–328, February 2020.