Swarup Sahu

Richard Xie

DATS 6202

4/30/2018

Image Captioning Model – Project Report

**Project Introduction (Problem Statement)**

The advent of new methods in deep learning has made previously impossible tasks, such as image captioning, feasible. A priori, neural networks are an effective way to process image files. In the same vein, methods in natural language processing allow individuals to effectively tokenize, or create a grouped sequence of characters, sentences which could later be used to create captions. This project *attempts* to combine both methods to process a dataset of images.

**Data Description**

The Flickr8K[[1]](#footnote-1)[[2]](#footnote-2) dataset is publicly available and provided by the University of Illinois at Urbana Champaign’s Dept. of Computer Science. The images within the .zip file were chosen from 6 Flickr groups, and represent a variety of variegated images. Within the entire dataset, there are 8,092 png files each with 5 *captions* describing each image. Hence the total *corpus* has 40,460 sentences within a .txt file (these sentences are generated by human readers). Furthermore, the dataset also includes annotations where a human reader ranks captions from 1-4 based on how well it describes an image[[3]](#footnote-3). The authors of this research project have further divided the dataset into a training, test, and validation dataset[[4]](#footnote-4).

**Preprocessing and Data Cleaning Theory**

Before creating the deep learning model, we first transform the images into a “machine readable format” and clean the text file for natural language processing. Each image is a uniquely created matrix of *pixel values* which describes how deeply shaded or colored a particular pixel is. The most common format (and the one we convert to in this project) is the *grayscale image*; the number is stored as an 8-bit integer giving a range of possible values between 0 and 255. Ultimately after conversion, one is left with an *m* long by *n* wide matrix which resembles a standard data matrix.

Color images add an extra dimension to this format. Separate red, green, and blue components are specified for each pixel which creates a 3D matrix where the 3rd dimension is the RBG component.This matrix is then recombined to create a 1 by *n* vector. Hence, we can run a for loop on directory of images and to extract each png files’ vector. After the data processing step, we are left with a dictionary of key-value pairs where the key represents the png file and the values represent each pixel within the image.

The text cleaning process has two components. First, we transform all sentences to lower case to avoid any bias in tokenization. Second, we remove all punctuation, numbers, and stop words. After the cleaning the data is reshaped so that it can be read into the model as a data matrix. By the end of the cleaning process there are approximately 8,700 unique words which will be used for tokenization, and the maximum length of any given text sequence is 34 words.

**Neural Network and NLP Model**

The initial motivation of the project was to create a neural network and natural language processing model without the use of computing packages. This however eventually proved to be infeasible due to time constraints and the specificity of the project’s requirements. Instead we use the *keras* packages in Python because it can help generate a LSTM recurrent neural network model and can help tokenize a corpus.

First, we load the VGG model from *keras*, a pre-trained model which will help “kickstart” the training process. The primary motivation for using this is to create a computationally efficient model which can help train future models. This pre-trained model is needed due to the limitations posed by the computational hardware at hand (4GB RAM). We next run the for loop that loads each image and use a keras method to transform the images into a standardized vector of 1 by 4096.

After the preprocessing step we generate a recurrent neural network with 256 hidden layers and the rectified linear unit (“relu”) activation function. This activation function is currently one of the most used functions in the industry because of its ability to side-step the “Vanishing Gradient Problem”[[5]](#footnote-5). What tends to happen when using probabilistic activation functions such as the hyperbolic tangent or the sigmoid function is that their gradients, or their derivates, tend to decrease exponentially (they resemble gaussian distributions with different 𝛿s). This ultimately effects the error function and causes it to decrease exponentially in deep neural networks causing the layers to either slow or completely stop training. Unlike these functions, the “relu” is either 0 when x < 0 or 1 when x > 0. Hence, the activation function does not face the same gradient problem as its counterparts. However, one weakness of this activation function is that negative inputs turn into zero and their subsequent gradients causing a level of bias within the model.

After we have processed the image files and built the RNN model, we create code for tokenization (i.e. create an integer – word mapping). This process is done recursively where we train the model by first giving it the image and the first word in a sentence describing the word. We then provide the image again, except we allow for two words to be passed in. This process continues until all the words are fed to the model (a 6-word phrase would have 7 iterations).

Finally, due to the computational constraints of the project, the model innovatively trains data in “batches” in order to avoid loading the entire dataset into a computer’s working memory (note that one *epoch*, or cycle, takes two hours to run). The *keras* package conveniently has a function called *fit\_generator[[6]](#footnote-6)* that allows for a model to train on a sub-sample of images while saving each iteration of model to be later aggregated.

**Model Evaluation**

The BLEU indicator, or the bilingual evaluation understudy, is an algorithm used for evaluating the “quality” of the text. It computes a precision metric between the actual description of the image and the predicted description based on n-grams. The theory behind the indicator is that it is meant to show the correlation between machine generated text and human text and is averaged over the entire corpus to give a metric. This code utilizes up to the 4th n-gram to evaluate the performance of the model[[7]](#footnote-7). Our predicted outputs generate BLEU scores of .55, .28, .17, and .07 (this is for a unigram, bigram, trigram, and 4-gram estimates).

 After having trained and tested the model, we try to validate the success of this image caption on 4 different png image files from different sources. The images are shown below:



One can notice that out of the 4 images, only one caption properly explains the data (‘dog is running through water’) whereas the “dog is playing on the grass” and “two men are playing in the water” has the subject of the image as wrong. The last image (bottom-right) is completely incorrect as the caption reads “two men are playing on the grass”. Ultimately the model has mixed success in accurately outputting captions.

**Conclusion, and Improvements**

The results of this project are mixed. As evidenced by the four random predictions made on images, the model does not generate robust descriptions for every image. Instead, this model captures the functionality of image captioning and succeeds in creating descriptions for certain images in certain environmental scenarios (for example: water, grass, etc.). This mixed success is due in part to the fact that computationally, we could only run one epoch without using up all memory and crashing the system. Future developers of this project should first ensure that their computing system is up to the task before attempting it.

Furthermore, the complexity of this project demands more knowledge and collaboration with other individuals. For a project such as this to succeed, a team of data scientists with different expertise in machine learning topics is strongly advised. With such a team, a more ambitious goal of creating a model without the use of packages becomes more achievable. Yet, if we keep in mind the scope of the DATS 6202 class and the time and computational constraints of this project, the model and its subsequent results can be deemed a moderate, but not a whole, success.

1. To access the dataset, an individual must register through this link: <https://forms.illinois.edu/sec/1713398> [↑](#footnote-ref-1)
2. The authors of the paper and dataset have requested they be cited in any projects using their data:

   M. Hodosh, P. Young and J. Hockenmaier (2013) "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics", Journal of Artifical Intellegence Research, Volume 47, pages 853-899 [↑](#footnote-ref-2)
3. For the methods used in the project we do not use this information of word rankings, though it is encouraged for a future developer to incorporate this information into their deep learning model. [↑](#footnote-ref-3)
4. We train on 6,000 images and test on 1,000 images (this has been conveniently created for us by the developers) using the BLEU indicator which will be discussed more in depth in later sections. [↑](#footnote-ref-4)
5. Sepp Hocrieiter identified this phenomena in his thesis paper. The opposite problem can be true where if a activation function has an increasing gradient can cause an “Exploding Gradient Problem”. [↑](#footnote-ref-5)
6. A guide to implementing this method can be found here: <https://medium.com/@fromtheast/implement-fit-generator-in-keras-61aa2786ce98> [↑](#footnote-ref-6)
7. For more information on the background and theory of the BLEU indicator, please see this paper’s methodology section here: <https://arxiv.org/abs/1703.09137> [↑](#footnote-ref-7)