Text Generation:

Creating Advertisements from Product Images

Sheryll Martutartus

Contents

1. Introduction

1.1 Description

In this project, our goal was to design our own text generation task to implement our learnings from the course. Though the focus was not on building a model with the best accuracy, we were to focus more on the implementation of a text generation task and evaluating the steps in the process. For my project, I attempted to implement a model that could create product descriptions/advertisements based on an image of the product itself. Though this is similar to image captioning, I believe it focuses more on creating a relationship between the words and image to create a product description/advertisement rather than simply describing what is seen in the image. After searching through Google Scholar for similar text generation tasks, I was not able to find one relating to product description/advertisement creation from images, highlighting the novelty and uniqueness of the idea. However, I believe that this type of task could be of importance to the online marketplace. By utilizing text generation, companies could utilize a wide-range of advertisements and product descriptions that have been successful in past sales and help them generate new ones given images of the product. It would save time and efficiency (of course fine tuning afterwards would be needed), and by incorporating past successful products and their advertisements, it could help them increase their visibility and sales of the product.

1.2 Goals

- Gain experience and knowledge of what text generation tasks exist
- Brainstorm new text generation tasks based on their need and importance
- Given the task, find or build an appropriate dataset to implement the task
- Learn how to extract features from images
- Learn how to corelate product descriptions with images to train the model
- Research which models are best/appropriate for such a task
- Despite the accuracy, still be able to obtain some results/product description based on an image
- Research and get a better understanding of how text generation models learn from datasets to make new predictions
- Based on my results (good or bad), evaluate my process and generate ideas on how I can improve in the future

2. Methods

2.1 Dataset

For my task, I wanted to be able to generate an advertisement based on the product's image. After extensive research, it was difficult to find a pre-existing dataset of advertisements and given that I would need an extensive number of samples, it would be difficult to generate thousands of advertisements manually. Therefore, I decided to aim my goal to creating product descriptions in hopes that once I have a baseline model, I could improve my input to eventually create persuasive text or advertisements. With that, I found that the Amazon Product Dataset 2020 [1] was a widely used dataset in many research papers and included the features that would be necessary in assisting my model training. The dataset contains around 10,000 samples, and I utilized an 80-20 train-test split of the data when training and evaluating its performance.

Though it had about 28 features, not all were useful for my text generation task. I utilized the product URL to create a script to extract the image and save it as a jpg into my designated dataset folder. I mainly utilized the product ID, product name, about the product, and technical description for each of the product's description. However, these descriptions also required some cleaning beforehand. I wrote a script to remove any unnecessary information that was generic and added multiple of the descriptions such as "Select the shipping method", "Ship it!", "Go to your orders and start the return", etc. in which these were included information that stayed when the dataset creator scrapped it from the website (this information would be useless and possibly harmful to the description generator).

2.2 Algorithm

In terms of implementing my task, I needed to understand what type of models would work best with the kind of dataset and input I had. Given that the main input for my task is of images, I utilized a Convolutional Neural Network because this model works well with scanning over images and extracting important features. Now that we extracted and interpreted the features from the images, we needed to align them with the product descriptions to be jointly used as the input into another model. The next model I choice is a type of Recurrent Neural Network, a Long-Short Term Memory model (LSTM), which is often used in NLP to generate the next word/sequences in text prediction tasks. By utilizing a LSTM, we are able to incorporate and maintain the past information/history for a longer amount of time, which aids in effectively predicting the next sequence. According to [2], LSTMs are advantageous because of their ability to handle long-term dependencies, and they are less susceptible to the vanishing gradient problem, making them an efficient and effective choice in predicting complex sentences. The diagram below shows the flow of the image and product description throughout the models and how they are all incorporated to predict a new description [3].

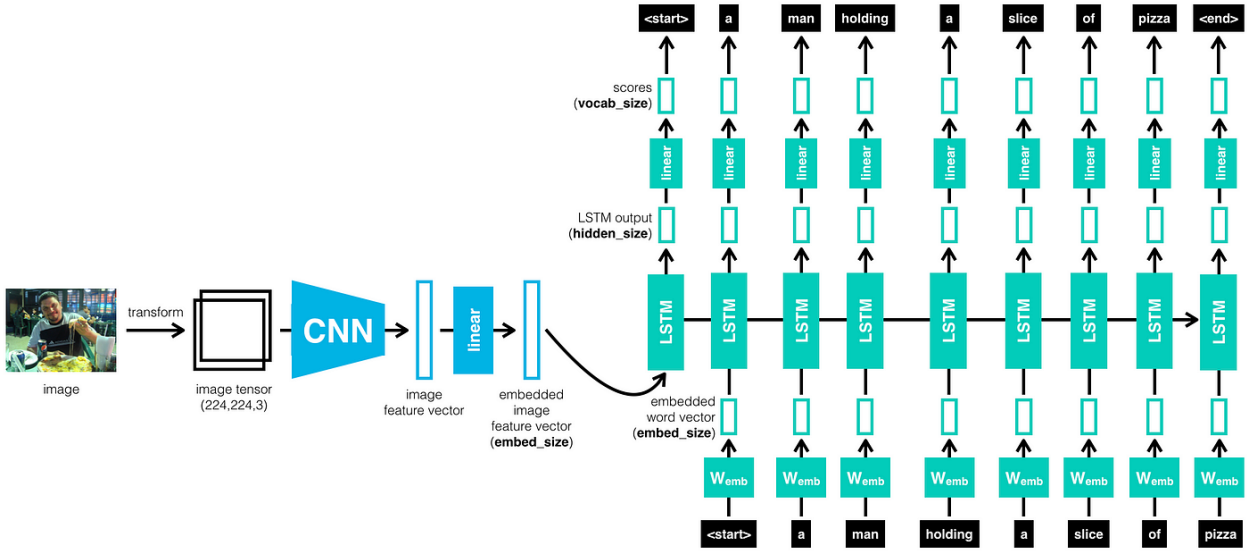

Figure 1: Steps and flow of input when generating a product description from its image.

Again, to summarize the process from the above diagram, I will train the models by passing in the training set of images through the CNN to extract the feature vector, and by keeping the fully connected model (eliminating the softmax last layer because we don't have a target/need a final prediction yet), we will then pass the embedded image into the LSTM. Within the LSTM, we will then utilize the features of the images along with the embedded words of the product description to train the model on how to predict the next sequence of words. Though the input to the CNN is just the image itself (with applied normalization and setting them all to the same image size), we needed a way to represent each of the words in each product description. Therefore, we utilized Keras' tokenizer library to eliminate the punctuation, change all letters to lowercase, and then apply a value/index to each word. This index would then be used to create a tokenized library of all the words, so that we have a numerical value to identify the word by and pass into the LSTM. I also added an <end> sequence to all the descriptions, which aided in detecting and predicting when the end of each description was hit (when to stop predicting).

For the training process, we utilized the Cross Entropy Loss function and the Adam optimizer as these were found to be the most commonly used in similar image captioning tasks. We then would train the model for about 10 epochs and save the model's parameters for ease in future use. Finally, we would evaluate the model on the testing data to obtain its predictions for each image. One way of evaluating, though by no scale of effectiveness, is by the human eye and interpretation of how close the description is in describing the product (if the words form a sentence/make sense or if they accurately describe the product). However, this is not a statistical means of evaluating the effectiveness, so I chose to utilize the ROUGE metric of evaluating text generation tasks. ROUGE generates a score for the predicted text vs. the ground-truth through the calculation of F1-score, precision and recall [4]. It operates by counting and matching the number of n-grams in each passage to give an idea of how well the model made its prediction and how well the prediction matches the ground-truth description.

3. Evaluation

After experimenting with different parameters of my model, I am disappointed in the outcome of this task's performance. My main issue I experienced was that in the testing/evaluation phase, the model kept outputting the same description for each of the different samples. I tried to pinpoint what could be causing this issue whether it was in the testing phase and evaluating on the same images/descriptions, yet they were all different inputs to the model. So, then I deduced it could be a result of the training process, and I could possibly be training on the same input (images/descriptions) too. From this, I was able to spot the problem; it wasn't related to the model or training/testing process, it was the fact that all the images were read as arrays of 1s or values very close to 1s. This factor then contributed to the fact the model was training on similar data (not a variety of inputs), so the weights all let to the same output and text sequence generated in the testing phase. Given that the issue was with the image vector, I traced it back to how each image was being read in. Despite finding multiple ways in Python to read images and convert them to arrays, it still returned an array of all 1s for almost every image. I attempted to debug this situation and research what could be the cause, but I was unsuccessful in finding a solution, and therefore, was stuck with a poor performing model.

I believe the structure of the model would have been efficient, yet my main cause of poor performance was the issue with the reading in of images. However, despite this setback, I continued with evaluating if/how different parameters could affect the model to the best of my ability. My first experimentation was with how different data size inputs would affect the training of the model and its prediction outputs. Initially, I would've thought that the more data exposed to the model, the better/longer the generated text sequence would be. Normally, more data = better results, yet in this case, more data resulted in more reading of images with 1s in its array and limited the output greatly. Below shows a comparison between the dataset size used and the corresponding output we received (keep in mind, due to the error, we received this output for all images):

Dataset Size (Train/Test)	Generated Sequence
500 (400/100)	"apunzel acrylic key ring end warrior we have you covered with diamond particles use with 481 collet or 4486 dremel chuck end 2 ply paper napkins high quality puzzle 14 marvel repair clamps type for hoses end for fast affordable three sizes and colors and shapes sticko stickers are acid"
1,000 (800/200)	"end may be slightly longer and novelty fully customized android system brings your lionel featuring the characters we know and love anna elsa kristoff olaf sven and marshmallow this line of dolls offers movie inspired details and who love a clear resilient coating to make cleanup easier black with red"
2,000 (1,600/400)	"end table set includes 1 trunk 1 top 1 skirt and gadgets that will crack them to your age now you can sit back relax and lose yourself

	in hours of coloring heaven end from infant toys 20 minutes shake the first business since 1946 whitmore has been dedicated star”
5,000	“end your party is”

Table 1: Dataset size vs. the output sequence obtained

As seen in the table above, the dataset of 5,000 produced the smallest output, while the smaller datasets produced a longer description, yet despite some errors, does make some makes when reading. However, again, this was the same description for each image, so it is difficult to evaluate its effectiveness in describing each and every individual image. It’s also interesting to note how every run incorporate “end” at the beginning of the sentence (which does not exist in the beginning of any description, yet the smallest dataset size of 500 did not. I did implement a ROGUE function to test and evaluate the predicted sequence, yet with the issue of receiving the same output for every image, it would be useless and impractical to utilize this metric.

Another experiment I tried implementing is seeing how the number of epochs during training affect the model’s output. Again, one would think that the more epochs, or exposure to the data, the better the model will be at identifying features and patterns to perform well. The table below shows a relationship between epochs and output with a dataset size of 1,000 samples. It’s difficult to pick out any trends or comparisons in the number of training epochs given that the size of the sequence is relatively the same and that they all begin with “end” again. With epochs of 1 and 3, they both seemed to mention “dc comic originals”, which shows the model was able to find a relationship between these words. It’s also interesting that in the sequence from 10 epochs, it also found a relationship between “anna elsa kristoff olaf sven”. Because the model is able to group similar words together like this, we can believe that it is, on some level, making relationships between words themselves.

Number of Training Epochs	Generated Sequence
1	"end your favorite licensed entertainment merchandise and original artworks dc comic originals flash logo sticker end for a variety of the newest icons joins the best armored tank in the world of the war the manufacturer aurora world's fantasy unicorn giddy up for a cake topper make a fun end"
3	“end the perfect gift for the first order stormtrooper imagine charging into a variety of colors and wholesale distributor for licensed entertainment merchandise and original artworks dc comics originals batman logo end exposes your child to the world since 1950 as you are printed with the dye sublimation process moveable”
5	“end standing height super soft microfiber sheet set includes a fitted sheet flat sheet measures 66 x 96 with coordinating fabrics color matching and attention to details to ensure that we are specialist in creative apparel designs with years of experience in trending and popular designs of various themes this”
10	“end may be slightly longer and novelty fully customized android system brings your lionel featuring the characters we know and love anna elsa kristoff olaf sven and marshmallow this line of dolls offers movie inspired details and who love a clear resilient coating to make cleanup easier black with red”

Table 2: Number of epochs vs. the output sequence obtained

Overall, due to the error we experience in loading the images, it is difficult to assess the performance of the model since all testing inputs received the same text generated output. However, we were still able to experiment with different parameters (dataset size, number of epochs) to see how they could impact the model and give us insight into what would be the optimal values for a successful model. I do regret that I was unable to find a solution to loading in the images and having different array values for them because then we would be able to analyze the model’s performance and draw a relationship between the images and the given descriptions.

4. Discussion

Though my implementation did not prove to be as successful as anticipated, I was able to learn more about text generation tasks, especially about the importance of the right dataset and features being inputted into the model itself. Because I am more familiar with creating classification models, this was my first time experimenting and implementing a text generation model, and I still found it to be a success on my end, not a success based on the model's performance, but rather a success in the sense that I have a better understanding of word embeddings and the process of text generation tasks.

For my task, it incorporated both images and corresponding product descriptions. Originally, it was difficult to brainstorm tasks and what their corresponding datasets would be like, yet I was lucky to come across the Amazon Product Dataset. Though it does contain around 10k samples (a sufficient size of input), I discovered throughout my project that in some cases, it is more about the quality of the data rather than the quantity. Many of the product descriptions contained irrelevant information, yet I believed I was able to clean out these pieces to the best of my ability prior to training. However, I believe it was difficult to generate product descriptions because of the wide range of data and variety of products. Because the product could be anything from a t-shirt to a toy to a science kit, the variety in products could have made it more difficult for the model to associate keywords to each specific item. My hope was that it would associate specific words like colors and shapes with each product to describe them and then extract common persuasive phrases in general to build a description, yet I believe that was too large of a task. I believe the quality of the dataset and its descriptions were one of the sole purposes that my task was not as successful as I'd hoped. For future attempts, I believe if I had a smaller set of images, yet with multiple descriptions for each product, it would aid the model in picking out certain descriptive words and associating it with the product image. Additionally, I could also narrow down the products to a specific category such as Clothing, Electronics, Toys, etc., which would again, help the model better associate certain words and phrases with aspects of the image. Since we wanted to focus more on the generation of the description, a dataset of products with less variety would better fit this task. I also originally wanted the task to be aimed towards advertisement generation, yet since it was difficult to find any datasets of ads for a product, this could be a future investigation or step after achieving a successful product description generation model.

Throughout my experimentation process, I experimented with the dataset size and train-test split to see if that was a factor of my model's performance. Of course, a dataset size of 100 would prove of no use, yet I experimented with the performance difference of 1,000 vs. 2,500 vs. 5,000 vs. all 10,000. Surprisingly enough, I discovered that the amount of data did not drastically improve the text generation predictions, which I again attribute to the wide variety of images and description topics. As well, I experimented with the train-test split of 80-20 vs. 70-30 as similarly found that the split did not improve the output of the model.

Another improvement I could foresee changes in the future is with how the descriptions and words are represented before used as input to the models. For my task, I analyzed multiple implementations of similar image captioning tasks, and for all of them, they utilized tokenizing the descriptions and just giving each word an index in the created set of words. Though it does keep track of each word's count, I believe this method gives us little to no information about the words themselves or their relationship to each other. To further this

task, I would focus on creating and improving word embeddings for each product description because this would draw similarities and relationships between the words and how they are used in the sentence (for example, POS tagging would be useful in constructing sentences). As opposed to the current method of just assigning a value to each word, word embeddings would help us to better understand each word and their role in the structure of the description and allow us to create a relationship between the key words in the description to appropriately describe the image (ex. Dinosaur, green, [size], toy would be useful when extracted from such image).

As previously mentioned, I believe the dataset, its quality, and its representation/vector input plays more of an important role in the model's performance in this case rather than the structure of the model itself. I utilized a pre-built model of PyTorch's ResNet50, which is a popular and widely used CNN model. Then, for the decoding portion, I utilized a LSTM, which is useful in incorporating long-term dependencies and past histories. However, one additional feature I would've liked to incorporate would be adding an attention mechanism within the decoding as it is often proved that attention within decoders/transformers increase the model's interpretability of the data and its performance in prediction. Due to lack of time and extent of the project, I believe the addition of an attention mechanism would also help improve the model's performance as it would highlight the important key words of the description that effectively describe/summarize the image.

Overall, though my task was proven unsuccessful due to an error in loading in the images, I believe this approach would be successful or at least provide us with further insight if I were to utilize a different dataset or set of images. I discovered a lot about how CNNs and LSTMs operate, their structures, their required input/output, and most interestingly, how we can combine them to generate text from images. In the future, I would like to obtain a more effective dataset to refine this task as well as experiment with other text generation tasks and see what the realm of NLP has to other.

5. Citations

[1] https://www.kaggle.com/datasets/promptcloud/amazon-product-dataset-2020?resource=download

[2] https://www.knowledgehut.com/blog/web-development/long-short-term-memory

[3] https://medium.com/@deepeshrishu09/automatic-image-captioning-with-pytorch-cf576c98d319

[4] https://medium.com/nlplanet/two-minutes-nlp-learn-the-rouge-metric-by-examples-f179cc285499

[5] https://towardsdatascience.com/how-to-build-an-image-captioning-model-in-pytorch-29b9d8fe2f8c

[6] https://medium.com/@jooheepark_78752/dl-memo-converting-simple-csv-files-into-pytorch-dataloader-deea64192409

[7] https://www.analyticsvidhya.com/blog/2021/12/step-by-step-guide-to-build-image-caption-generator-using-deep-learning/

[8] https://pypi.org/project/rouge-score/