공학석사 학위 논문

Detecting Parts of Speech from Image for Caption Generation

영어 품사 정보를 활용한
이미지 캡션 생성 모델

2019년 8월

서울대학교 대학원
컴퓨터공학부
강 필 구

Detecting Parts of Speech from Image for Caption Generation

영어 품사 정보를 활용한
이미지 캡션 생성 모델

지도교수 김 형 주

이 논문을 공학석사 학위논문으로 제출함
2019년 12월

서울대학교 대학원
컴퓨터공학부
강 필 구

강필구의 석사 학위논문을 인준함
2020년 1월

위 원 장 이 상 구 (인)
부위원장 김 형 주 (인)
위 원 문 봉 기 (인)

Abstract

Detecting Parts of Speech from Image for Caption Generation

Phil Goo Kang
Department of Computer Science & Engineering
School of Engineering
Seoul National University

The capability to generate a description about the content of an image is becoming more important with the integration of smart devices and reliance on AI into our daily lives. In this paper, we propose a novel approach that utilizes multiple CNN models that have been specially trained to detect features related to the parts of speech (PoS) such as noun, verb, pronoun, adjective, preposition and conjunction. Using the PoS based CNN models, we extract features that the language model uses to generate high quality captions. We validate our finds by using Flickr8k, Flickr30k and MSCOCO dataset through multiple human surveys and several popular automatic text metrics.

Keywords: Image caption, Picture description languages, Image

Processing and Computer Vision

Student Number: 2018-22788

Table of Contents

List of Figures

List of Tables

List of Functions

Chapter 1. Introduction

In the recent years, research on image captioning, the process of generating a description for an image, has garnered attention from researchers due to its wide range of useful applications – generation of automatic product descriptions, guidance for the visually impaired, and interaction between human and computer to name a few.

Image captioning essentially relies on visual and linguistic understanding, requiring precision in the detection of visual features and the selection of words to generate semantically and syntactically correct captions. The *whats* and the *hows* of these actions have their respective challenges; combining them to cooperate as a single entity engenders a whole new set of problems. For example, an extracted visual feature, despite its importance to the whole image, may not align with what is considered useful for the language model to generate an effective sentence. This type of misdetection would trigger a misguided selection thereby formulating an incorrect caption, deeming the ability to extract visual features that qualify as both useful and important indispensable.

Numerous models aiming to compensate for these limitations and challenges have been proposed. The most renown, by Vinyals el at [35], is an approach whereby a CNN model for extracting object–related features is used in combination with a

long short term model (LSTM) for generating caption sentences. Another approach, by Fang el at [11], utilizes multiple instance learning (MIL) models to detect features from various locations within an image. Others employ an additional model to detect image attributes [40] alongside the feature detector model.

A commonality discerned among these methods is the lack of address for issues related to grammar. A "good" sentence is one that is semantically and syntactically correct: the rules of grammar are aptly applied. Unfortunately, the aforementioned models implementing the CNN model for detecting visual features and attributes rely solely on the language model to resolve all grammar-related issues. As such, limitations are inevitable due to the vastness of grammar itself. We resolve that if the CNN model were to simultaneously engage with the language model in tackling grammar-related issues, the quality of generated captions would improve.

In order for the CNN model to successfully interact with the language model, it would need to be able to accurately detect grammatical features. We first examined the building blocks of grammar – the basis being different parts of speech – so that we may be able to identify the features related to the various parts of speech from an image, essentially detecting grammatical features. Moreover, because all visual features, when classified, result in key words that belong to a part of speech group, we would consequently be able to distinguish parts of speech features from an image that would enable the detection of both

visual features and grammatical features at the same time.

Our paper presents a novel approach that utilizes a parts of speech based CNN model called PCR. We integrate multiple CNN models that have been trained to extract features related to each of the English parts of speech (PoS): noun, pronoun, verb, adjective, adverb, conjunction, preposition, and interjection.

We take the outputted features from the PoS CNN models and feed them directly into a language model based on LSTM [15]. By detecting the different parts of speech presented in the image, using the PoS CNN models, we are able to generate a caption that is similar in quality to human generated captions.

To evaluate our approach, our model was trained using Flickr8k [16], Flickr30k [41], and MS-COCO [24] dataset, which is publicly available online. We evaluated our model with two different types of human surveys and several popular automatic text comparison metrics such as BLEU [27], CIDer [30], ROUGE [23], and SPICE [2]. For BLEU-4 metrics on MS-COCO [24] dataset, our model scored a value of 34.27 which outperforms previous works. Using these evaluation methods, we attempt to prove that our approach has the capability of generating high quality captions through the use of PoS based CNN-RNN models.

This paper is organized as follows: Section 2 we explain prior research methods and juxtapose their approach. Section 3 we give an overview of our model and explain our PoS CNN, and Section 4 we evaluate our model and in section 5 we conclude

our research.

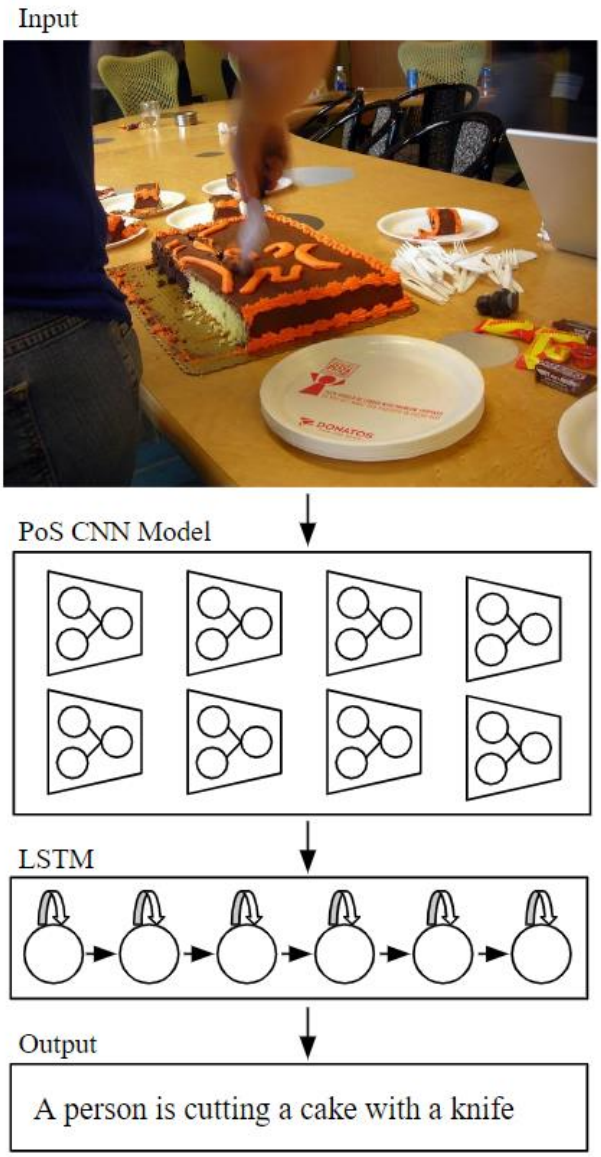


Figure 1. Our model utilizes multiple CNN models to detect different parts of speech related features from an image and feed them to a LSTM model to generate an image captions.

Chapter 2. Related Work

Image caption research falls into two main dimensions: retrieval based and generation based. There is a recent survey by Dao at el [3] who presents a comprehensive report about the two different dimensions.

The first dimension, called retrieval based, is an approach that tries to match the input image to correlate with a sentence in the sentence pool. The sentence pool is a set of sentences created by humans. The result of this approach is almost always well-formed sentences. However, there exists several limitations. The first is the sentence pool itself. The different combinations and sentence patterns created are limited to the preexisting human generated sentences. The second problem is the incapability to handle unseen images. This approach requires the matching between sentence and image but if the image is an unseen new image sentence matching will not occur. Even with these limitations several proposals have been made. Farhadi et al [12] used triplets detected from an image to generate a sentence that best matches the detected triplets. Triplets are keywords that describe object, action, and scene from the input image. Kulkarni at el [21] used the objects, attributes, and prepositions detected from an image and a conditional random field to correctly fill in a sentence template with the most likely word. Yang at el [38] used a Hidden Markov Model (HMM) to

generate a template sentence based on selecting the objects, scenes, verbs, and prepositions with the highest log−likelihood ratio value.

Generation based approach designs a model that learns the probability distribution of the sequence of text. Unlike its counterpart, the retrieval based method, generation method is not limited to its training pool and can working with previously unseen images. Generation approach is more flexible to a wide variety of situation, especially for previously unseen images, thus it is the preferred approach. The most well−known and popular generation based approach is Vinyals et at [35]’ s neural image captioner (NIC). Vinyals et al [35] proposed method is to use a CNN based feature extractor where the fully connected layer of the CNN is fed directly into a LSTM model that will generate a sentence. Xu et al [36] extends the NIC model with the addition of an attention mechanism. The attention mechanism selects the features that will be fed in to the LSTM model focusing the target of the sentence. Yao et al [39] method extracted both features and attributes from the image and used different feeding combinations to boost the LSTM prediction. In You et al [40] paper, they employed a semantic attention model and a feedback loop to guide the recurrent neural network language model during each iteration when it creates the sentence. Jia et al [17] also guided the language model by adding a bias to words that are semantically linked to the content of the image. Dai et al [8] separated the semantic and caption problem

using two different LSTM models making the generation problem a two-step process. The first LSTM generates short phrases overcoming semantic issues and the second LSTM utilizes the short phrases to create the complete sentence. Most recently, He et al [14] extracted the PoS tag for each word and fed both the PoS tag and the feature vectors to the LSTM model to help guide and prevent overfitting. Other works utilizing this approach include [10], [11], [26], and [5].

Our work belongs to the generation based approach. We directly extend the work of Vinyals et at [35]' s NIC while receiving influence from He et al [14] and Fang et al [11]. Both He et al [14] and Fang et al [11] approach utilizes the PoS for each word but in our case, we attempt to detect features related to each PoS instead of using the tags themselves as input [14] or as a filtering mechanism [11].

Chapter 3. Proposed Model

Our model extends the architecture of [35] which is based upon the most widely used encoder–decoder framework [6]. In our approach, the encoder is eight separate CNN [22] models which extracts PoS related features from the image and the output of each of the eight CNN models are combined into a single tensor to be fed into the decoder. The decoder is a recurrent neural network (RNN) [18] which takes the CNN extracted features and generates a captions as illustrated in Figure 1.

3.1. Encoder

Attempting to design each layer of the features detecting CNN model is a difficult task and an inefficient approach. There already exist several popular and verified CNN object detection architectures that work very well such as [20], [30], [13], [33], [32] which are all used in prior methods introduced in section 2. Among these architectures, we have selected ResNet-152 [13] because it scored the highest accuracy in the Large Scale Visual Recognition Challenge [29] in 2015.

Training a CNN model to detect specific features in an image related to the PoS is a hard task. Approaches like multiple instance learning and unsupervised learning were tested to detect PoS related features, however, they showed less accuracy then directly training the CNN model through supervised learning. To explain the process of generating PoS CNN models, we will first introduce how we prepared our training dataset and then describe the training process.

3.1.1. Preparing PoS Dataset

Our first step, in creating PoS based CNN models, is to preprocess our training caption for Flickr8k [16], Flickr30k [41], and MS-COCO [24] using NLTK [4] to detect parts of speech for each word in the dataset. For each dataset, the number of words corresponding to each of the parts of speech are organized in Table 1. The rows in bold, for Table 1, represent the PoS groups that showed the best predicting results. Further details

are explained in section 4.6.

3.1.2. Generating PoS Detecting CNN

Each of the parts of speech based CNN models are trained separately. We first assign each of the eight CNN models with a parts of speech group. For example, we assign one model as the noun group, another model, the verb group, the third model the pronoun group, and so on. Afterwards, we take the dataset that belongs to the models parts of speech group and run through a supervised training process.

Using the noun model as the first example, we train the model by inputting an image that has a noun in its caption and making the target the noun word. We repeat this step for all the images with noun words in its caption for the training dataset. Figuring out the images with nouns in their caption and the target noun word is done during the preparing dataset process as mention in section 3.1.1.

After we have completely trained the noun model, we select one of the remaining seven CNN models that are waiting to be trained. If we use the verb model as our second example, we train the verb model by inputting an image that has a caption with verb word in it. We set the target as the verb word and repeat this step for all the verb words in our training dataset—which is the same process for the noun model.

We repeat this same process a total of eight times—once for each of the different parts of speech models. After this process

is complete, each of our PoS CNN models will have the ability to detect features within an image that correlates to the models part of speech group.

Parts of Speech	Flickr8k	Flickr30k	MS−COCO
Noun	**116,932**	**478,411**	**2,168,844**
Pronoun	12,046	52,117	154,574
Verb	**44,197**	**192,518**	**685,503**
Adjective	**101,542**	**398,137**	**1,784,600**
Adverb	9,077	37,472	114,437
Conjunction	**54,795**	**232,697**	**1,087,985**
Preposition	**48,030**	**201,438**	**951,463**
Interjection	0	5	12

Table 1. The number of words for each of the parts of speech for Flickr8k, Flickr30k, and MS−COCO. The rows in bold represent the parts of speech CNN models that were selected in our final model to generate the evaluation caption as explained in section 4.6.

One method we use to confirm that this method of training works is by generating a heat map using Grad-CAM [31]. Grad-CAM highlights the region of the image that the model considers important during the predicting process. Example output of using Grad-CAM are available in Figure 2 and Figure 3. Figure 2 is an example of the adjective CNN model correlating the bear and adjective work 'black' for the given image. Figure 3 is an example for the verb CNN model correlating the woman with the keyword 'brushing'. This process of detecting features using PoS CNN models are represented in (1) where we denote the input image as I.

$$X_x = \sum_j^{\#\ of\ PoS} CNN_j(I) \quad (1)$$

Combining the output of the different PoS CNN models into a single output was another challenge that appears because of our approach. Several different approaches such as concatenating all outputs into a single tensor, add or multiply the output matrix of each PoS CNN models, take the average value of the output matrix by element-wise, and use a linear function to transform the output. Among the different approaches, taking the average value of the output matrix by element-wise showed the best results. This process is represented from (1) to (2).

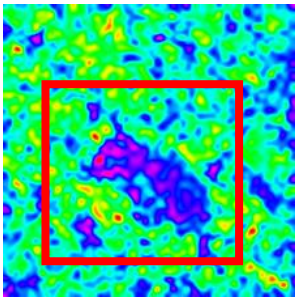
$$x_{-1} = AVG(X_x) \quad (2)$$

Element−wise matrix calculations are simple to implement because the output of each of the PoS CNN models are equal size square matrices. Taking the average value can result in loss of data but it is shown that through back propagation the loss was negated.

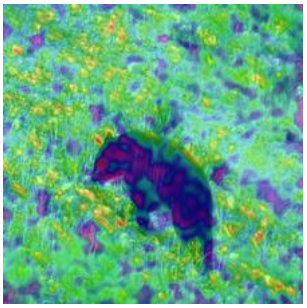
Equation (1) and (2) takes place in the encoding part of the system and the output tensor of the encoder is denoted as x_{-1}. It is denoted this way because x_{-1} is the initial input to the LSTM where the LSTM will begin generating the word w_0 at time zero.

Input Image:

Adjective CNN Model Grad-CAM Heat Map



Overlay Input Image and Grad-CAM Heat Map

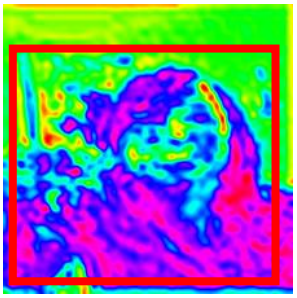


Generated Caption: a *black* bear walking through a forest with trees in the background

Figure 2. The Adjective CNN model detected the word black which is present in our caption. The red box represents the detecting zone.

Input Image:

Adjective CNN Model Grad-CAM Heat Map



Overlay Input Image and Grad-CAM Heat Map

Generated Caption: a woman is *brushing* her teeth with a toothbrush

Figure 3. The Verb CNN model detected the word brushing which is present in our caption. The red box represents the detecting zone.

3.2. Decoder

The decoder models job is to translate the features from the output of the encoder, represent by (2) and utilize them to generate our caption as represented in (3).

$$w_t = LSTM(x_t) \tag{3}$$

A key requirement for the decoder is the capability to generate sequential data. Several forms of recurrent nets exist such as vanilla recurrent neural network (RNN) [18], long short term memory (LSTM) [15] and gated recurrent unit (GRU) [7] are available and have been used to solve similar sequential generation problems.

We have selected the LSTM form because of the advantages that comes with the built-in memory cell gates. LSTM uses three gates called input (4), forget (5), and output (6). The input gate manages how much of the new cell state to keep. The forget gate, as its name implies, decides how much of the previously existing memory state to forget. The output gate controls the exposure of the cell to the next state. The formula for each gates are as follows:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{4}$$

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{5}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{6}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \tag{7}$$

$$m_t = o_t \otimes c_t \quad (8)$$

The cell memory, represented by c_t, takes the previous state and combines it with the new state as represented in (7). The result of the memory cell is represented by m_t, which is the result of the product of the cell state and the output gate as in (8). The complete structure of the memory cell is represented in Figure 4.

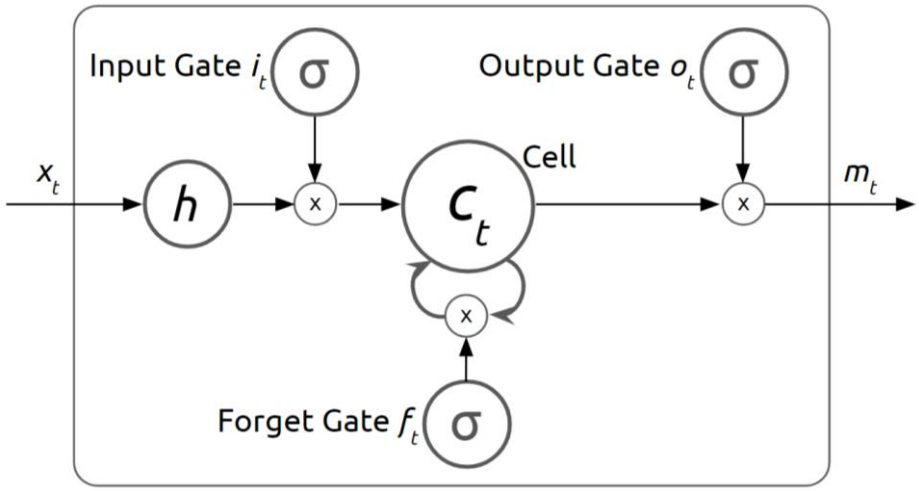


Figure 4. A LSTM cell structure. The LSTM cell uses three gates called input, forget, and output to control the flow of data.

3.3. Training

We train the entire system in an end-to-end approach following the method in [35]. The end-to-end approach means the PoS CNN models do not predict a classification but the output is an embedding and the embedding's are directly fed to the LSTM language model. When we back propagate the loss value, both the LSTM model and the CNN model are affected.

During the end-to-end training process, our loss is calculated using the Cross Entropy function. Cross Entropy compares the target value with the predicted value and penalizes based on their distance value. The Cross Entropy function is useful because it can compare the output of our model, which is the predicted caption, and the training caption in our training dataset.

To optimize the hyper parameters during training, a separate validation set [37] was created at the start of the research. For the PoS CNN models, each PoS CNN model is trained separately with its correlating dataset. Since we are training a single CNN model at a time, to reduce training time and prevent overfitting, batch training [25] and early stop [28] techniques were implemented. Batch training [25] processes n number of data samples before updating the model. Early stop is a technique that stops training before the model's weight values have converged—however, early stop models may not be the best performing version of the model. Also, for the end-to-end training of the encoder and decoder, both batch training and early

stop techniques were used as well.

3.3.1 Combine PoS CNN Model Output

Our method generated an unforeseeable issue using Vinyals el at [35] end-to-end system structure. In Vinyals el at [35] approach, the encoder is a single CNN model and the decoder is a single LSTM model. Therefore, in Vinyals el at [35] situation, the output of the CNN model and be directly fed to the LSTM model without any issue. However, for our case, our encoder are eight separate CNN models—one model for each of the parts of speech group. To utilize an end-to-end system, we needed to solve the issue of combining the output of the multiple CNN models to in a single matrix. Because the output of the CNN model are all square matrix tensors, we are open to several arithmetic options.

We attempted five different approaches, which are concatenation, linear transformation, element-wise addition, element-wise multiplication, and element-wise matrix average. Inside each of the approaches are actual multiple sub-approaches because several different combinations are available inside each approach.

The first and basic approach was to concatenate all the output matrix of the PoS CNN models into a single tensor to feed the LSTM model as seen in Figure 5. There were two major downsides to this approach. The first downside was the fact that selected the order and shape of the output matrix had no standard

rule but was based on our choice. The lack of an objective rule reduces the effect of the approach. The second downside was the size of the resulting matrix. The concatenated matrix was the combining of eight CNN model output resulting of a large tensor that was hard to manage.

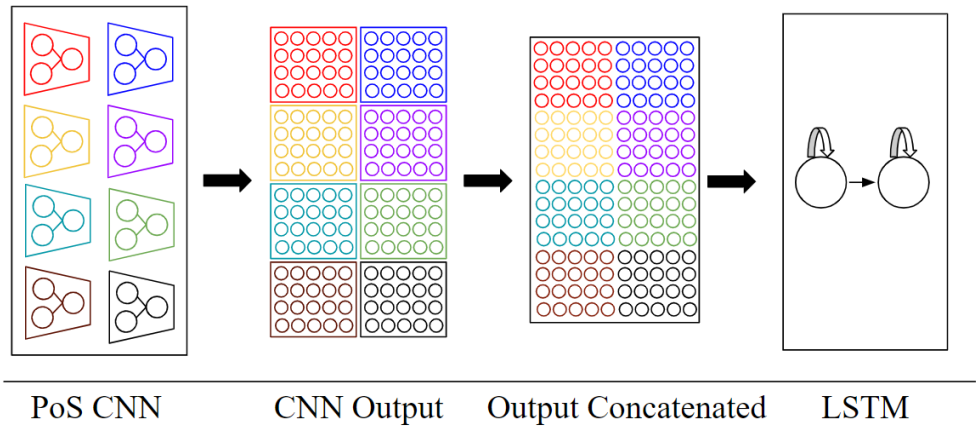


Figure 5. Showing the process of combining the output of the PoS CNN model by concatenation before feeding the RNN model.

The second approach was to transform the matrices into a single tensor using a linear function as seen in Figure 6. By transforming the output, we remove the hassle of calculations and remove the possibility of any data loss appearing due to the arithmetic operation. However, when apply this method, the average BLEU score value was lower than the concatenation method. Therefore, while this method is concise because of its poor capability we did not use it in the final model.

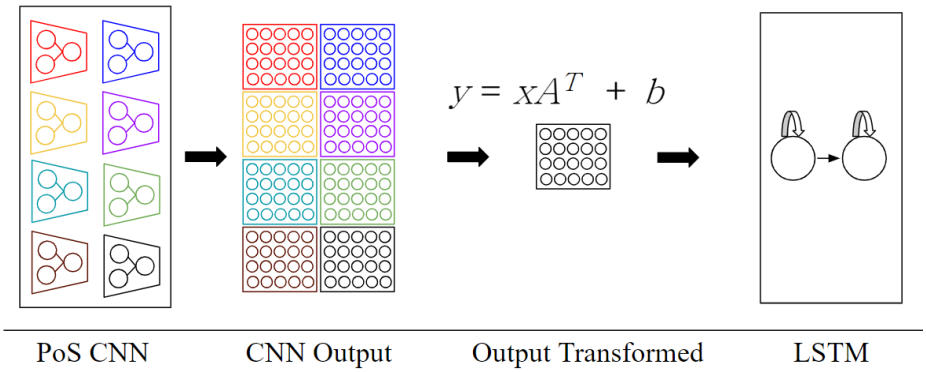


Figure 6. Showing the process of combining the output of the PoS CNN model by transformation before feeding the RNN model.

As mentioned before, there are several element−wise arithmetic operations that we have undertaken. However, among all the element−wise arithmetic operations, concatenation, and transformation, the best results came from combining the multiple CNN model outputs is the element−wise average. Therefore, in our final model is based on element−wise average method as seen in (2).

3.3.2 Serving PoS CNN Output to RNN

Yao at el [39] proposed several different combinations of serving the output of the encoder model to the decoder model. Among the different combinations, three major approaches are compatible with our model and they are as follows: 1. at start and each iteration, 2. at each iteration, and 3. only at start.

Theoretically, the first approach of feeding the decoder at the beginning and at each iteration should have the best result

because this method offers the decoder with the most amount of data as seen in Figure 7. Even in Yao at el [39] paper, this approach showed the best results. However, in our case, due to the fact that we use multiple models in the encoder, the decoder becomes overwhelmed with data and cannot properly work.

The second approach, represent in Figure 8, feeds the output of the encoder to the decoder only at each iteration. However, sometimes, this approach causes the decoder to begin at the wrong point, resulting in a completely wrong prediction.

In our final model, we utilized the third approach, which was originally proposed by Vinyals el at [35]. The output of the encoder is fed to the decoder only at the beginning as seen in Figure 9.

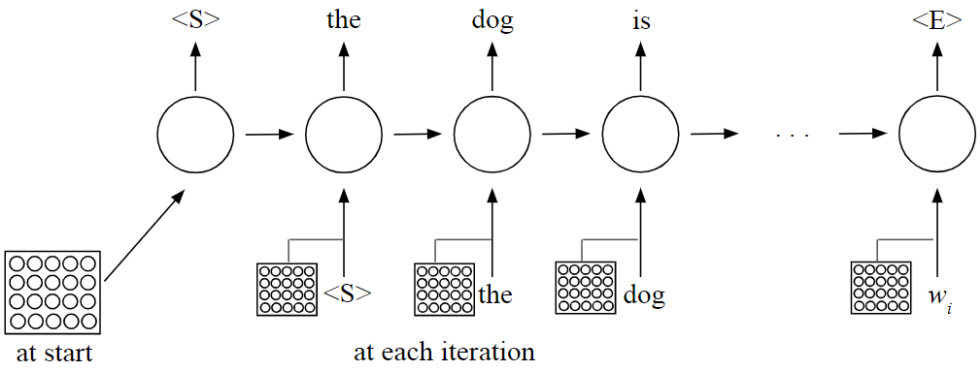


Figure 7. PoS CNN output matrix being fed to the LSTM at the beginning and at each recurrent iteration.

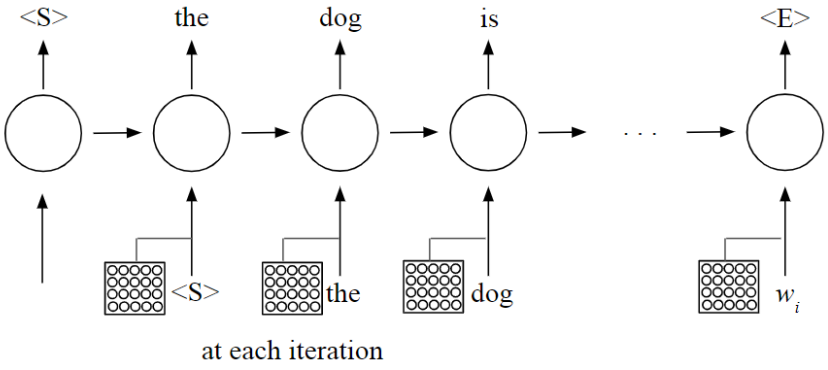


Figure 8. PoS CNN output matrix being fed to the LSTM only at each recurrent iteration.

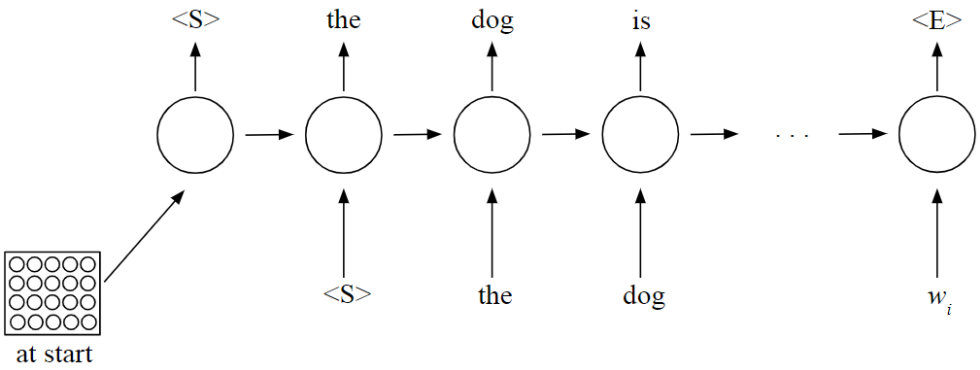


Figure 9. PoS CNN output matrix being fed to the LSTM at the beginning only.

3.3.3 End-To-End Loss Optimization

During the "end to end" training process, our loss is calculated using the Cross Entropy function. Cross Entropy compares the target value with the predicted value and penalizes based on their distance value. However, the Cross Entropy function operated different in PyTorch than the original function. The Cross Entropy used in PyTorch is defined as follows in (9) and (10).

$$\text{loss}(x, class) = -\log\left(\frac{\exp(x[class])}{\Sigma_j \exp(x[j])}\right) \quad (9)$$

$$= -x[class] + \log\left(\sum_j \exp(x[j])\right) \quad (10)$$

To use Cross Entropy in PyTorch with any of our dataset, we first need to label encode each word and then apply one hot encoding. Label encoding converts a corpus of strings into values

between 0 and *corpus_size* − 1. One hot encoding encodes integer values as a one hot numeric value this is to prevent the model from mixing up an identification value from a numeric value.

3.4. Inference

We used two popular inference method called sampling and beam search to generate our caption with LSTM. Sampling method selects the highest predicted word at each time t and feed that word back into the model to predict the next word for time t + 1. Beam search considers the top k predicted words at each time t and feeds them all back in to the model to open up the probability of a better sentence to appear. Details about their usage are explored in section 4.3.

Chapter 4. Experiment

4.1. Environment

The specifications for our server which we developed, trained, and tested our model are as follows: OS is Ubuntu 16.04 LTS, CPU is Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz, RAM is Samsung DDR4 128G, SSD is 480 GB Samsung MZ7LM480, and GPU is four NVIDIA GP102 TITAN Xp.

4.2. Data

We evaluated our model using three different datasets: Flickr8k [16], Flickr30k [41], and MS-COCO [24].

Flickr8k [16] is a dataset created by crowdsourcing on the Flickr website. The majority of the content of the image is based on human daily activities. The dataset comes with train, validation, and test set already separated. The dataset has a total of 80,000 images with captions. The training set is 30,000, validation set is 25,000, and test set is also 25,000.

Flickr30k [41] is a dataset that extends Flickr8k. Flickr30k is also based on crowdsourcing on the Flickr website. Flickr30k has a total of 158,915 captions with images. Unlike Flickr8k, Flickr30k has no standard split for train, validation, and test. Therefore, following [39] and [1] work, we split the testing and validation dataset by 5,000 each and the remaining dataset will be used as our training set.

MS-COCO [24] dataset has a 2014 and a 2017 version. We have selected MS-COCO 2017 because it has a larger dataset of 596,755 images and 2,983,775 human captions. We separated 5,000 for validation following [39] and [1] work. We also separated 5,000 for testing just like [39] and [1]. The remaining 586,755 images are used for training.

4.3. Implementation

Our model uses a three-layer LSTM with an input size of 1024 and a hidden node of 512. For the training algorithm, we have selected ADAptive Moment estimation (ADAM) [19] because of its speed and reliability—it has not failed to converge in our experiment so far. We use a learning rate of 10^{-3} for the entire end-to-end training. Several different batch sizes were test but 128 showed the best result. We placed no limit on the number of epoch and used a patience value of 7 for early stop. As previously mentioned, when generating our caption, we used sampling and beam search for inference. For beam search, a beam size of 3, 5, 10, and 20 were all tested. However, on average, beam search resulted in a BLEU-4 score 2 lower than sampling. Therefore, in our final result we presented using sampling inference method.

To reduce word noise, we placed a threshold of 5 following [35]. Any word in the corpus that does not appear more than 5 times was dropped. The time taken to train, validate, and test took on average of 8 hours.

As previously mention in section 3.2, batch normalization and early stop methods were utilized to prevent overfitting and optimize training. However, to further prevent overfitting and generalize our model usage, we replaced our noun model with an object detection model that was pretrained on ImageNet [29]. This replacement step was taken to further deter overfitting because the model would be based on another dataset. Detail comparison of using this replace model verses a noun model is explained in section 4.6.

4.4. Evaluation Metrics

Determining what considers as a good caption is a hard task. Currently, there are no standard method to determine if a given sentence is a good caption. Therefore, we evaluated our caption generator model through human survey and automatic reference text comparison metric calculations.

Human survey can be considered as one of the best measuring tool. However, there still are variables that limit the quality and frequency of human survey. For example, the same human testers cannot be used for every research. Human factors like personality, experience, body condition, and state of mind can also affect their decision making. Most importantly, during the survey, humans can also make some mistakes increasing error rate which is unpredictable. To reduce the effects of these variables, we conducted two different types of surveys.

The first survey follows [35] method. A total of 40 questions

were given in the survey. The format for the survey was an image and a caption was given to the tester. Then, from a set of multiple choices, the tester would rate our caption. We replicated the multiple choices present in [35] which are as follows: 1) describes without errors, 2) describes with minor errors, 3) some-what related to the image, 4) unrelated to the image. The questions to the survey were taken from the test dataset explained in section 4.3. To prevent skewing when selecting random questions from the test set, we first grouped the test set by BLEU-4 scores. To eliminate outliers, we removed images with scores from 0~10 and 90~100. We selected 5 random images from every 10 point intervals (ex. 10~20, 20~30, 30~40, etc.) resulting in a total of 40 questions. The survey questions were based on our MS-COCO data test set—neither Flickr8k nor Flickr30k test set were used. The purpose of this survey is to evaluate the quality of our caption generator. By taking the element size average value of all the selected answer by the test takers we can know how the test takers felt about the quality of our model. The smaller the value means more people selected the first choice meaning the caption generated has no issues.

The second survey consists of 25 questions. For each image, three multiple choices were given. The first choice is the ground truth, second is the caption generated by our model, and the third choice is an option saying that the two choices are similar. When selecting the questions from the test set, we followed the same method used for the first survey and just like the first survey,

the questions were only taken from the MS-COCO test set and not Flickr 8k or Flickr30k. Also, we excluded the images used in the first survey when generating the questions for the second survey—meaning no duplicate images were used during both surveys. The purpose for this survey is to see how well our model compares to ground truth.

Comparing our model to other prior research models through human survey is a difficult task. Therefore, we utilized automatic reference comparison metrics to juxtapose against previous approaches. We using four popular text comparison metrics called BLEU [27], CIDEr [30], ROUGE [23], and SPICE [2]. Each of these metrics take different approaches when comparing the ground truth with the proposed caption. In all these metrics, having a higher score means that the generated caption is similar to the human generated caption. Therefore, achieving a higher score in these metrics is our goal.

BLEU [27] metrics, which stands for BiLingual Evaluation Understudy, compares how many machine generated works appear in the human generated caption through n-grams.

CIDEr [30] metrics, which stands for Consensus-based Image Description Evaluation, compares the words generated by the machine with a consensus caption by humans.

ROUGE [23] metrics, which stands for Recall-Oriented Understudy for Gisting Evaluation, compares how many human generated words appear in the machine generated caption. ROUGE [23] can be seen as the opposite of BLEU [27].

SPICE [2] metrics, which stands for Semantic Propositional Image Caption Evaluation, focuses on detecting features related to colors and correct count of objects.

4.5. Results

We have used several different evaluation methods to ensure that we have a full understanding of how well our approach works compared to prior research and to human caption generation skills. In section 4.5.1, we will talk about the survey that evaluated the quality of our caption. In section 4.5.2, we will talk about the survey for the one-on-one comparison against the ground truth and our caption. Lastly, in section 4.5.3, we will talk about the automatic evaluation metrics.

4.5.1. Evaluate Caption Quality

The first survey is related to evaluating the quality of our caption. The results for our first survey is organized in Table 2. A total of 21 people took our survey. Our test taker comes from a wide variety of different backgrounds and age group. Their age range from 24~35 years old. Half of the test takers are university students and the other half are working professionals with jobs. When looking at the results of the survey, the distribution of the answers seems skewed towards the "No Errors" answer by a ratio of 46.42%. By looking at these results, one can assume that the questions were selected with bias. However, in section 4.4, we have explained in details about

how the questions were selected and the steps taken to prevent any bias or manipulation. We can conclude from the ratio of the first answer that almost half of all the captions generated by our model have no errors by human standards.

The second highest selected answer is "Minor Errors". In the official survey, the "Minor Errors" was defined as at most one or two errors in the caption. Even though there were one or two errors in the caption, human testers had no problem understanding the content. Therefore, we can conclude that any caption generated by our model have a 73.45% chance of generating a caption that was suitable for human interaction. We can see that this survey provides substantial evidence that our model generates high quality captions.

However, there still remains 11.54% of time when our model generates completely wrong and unrelated caption. In Figure 13 are four examples of when our model generates completely unrelated captions. We can see from these four examples that if the image is too dark to distinguish features then our model fails to work properly. However, this issue is a well-known problem in the field of computer vision where the lighting and angle of the image of the same location and object can affect the detection quality. Issues like these are not specific to our model but are issues related to the entire field of computer vision. Nonetheless, considering the diverse test set from MSCOCO, 11.54% is not a high value. Figure 10 are examples where our model generated perfectly acceptable captions for images in the

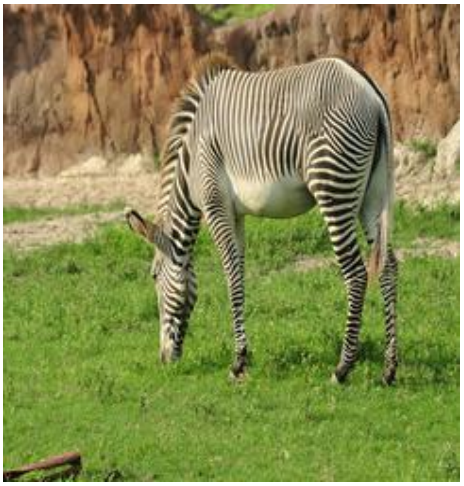
first survey Figure 11 and Figure 12 are examples with minor to some errors in the caption.

Answer	Count	Ratio
No Errors	390	46.42
Minor Errors	227	27.02
Little Related	126	15.00
Unrelated	97	11.54

Table 2. Results for the first survey where human testers rated our caption that describe the content of an image from the MS-COCO dataset. Human testers rated our caption from a set of four multiple choices as seen in the table. The average score our model received from this survey was 1.92.

A fire hydrant is sitting on the side of a road

A group of people sitting at a table with food



A zebra standing in a field of grass

A baseball player holding a bat on the field

Figure 10. Several images and captions taken from the caption quality survey that received the height scores for "Describes without errors".

A baby laying on a bed with a blanket

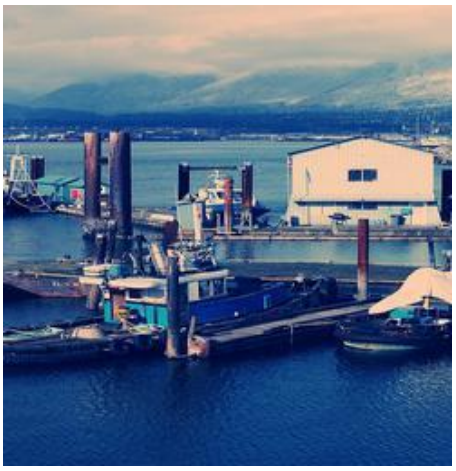
A man walking down a street holding an umbrella

A group of people riding motorcycles on a street

A group of people standing on a beach with a frisbee

Figure 11. Several images and captions taken from the caption quality survey that received the height scores for "Describes with minor errors"

A man flying a kite on a beach



A large white boat sitting on a river

A man is skiing down a snowy hill

A man riding a skateboard down the side of a ramp

Figure 12. Several images and captions taken from the caption quality survey that received the height scores for “Somewhat related to the image”.

A man is walking a horse on the beach

A woman sitting on a bench with a dog

A night time scene of a city street

A glass of orange juice sitting next to a cup of coffee

Figure 13. Several images and captions taken from the caption quality survey that received the height scores for “Unrelated to the image”.

4.5.2. Ground Truth Comparison

The first survey summarized in Table 2 was to determine whether or not our model was generating good quality captions. However, rating our caption alone does offer enough of an understanding to decide whether it is usable in the real world settings. Therefore, a second survey was given to the same test takers of the first survey. In the second survey, our captions were compared one-on-one against the ground truth captions. The results are available in Table 3.

On average, about 44.6% of the time, our model generated captions that are equal-to or better than human generated captions. Figure 14 are three actual problems taken from the survey. For the first image, in Figure 14, our caption received a total of 19 votes out of 20 which is one of the questions that receive the most amount of votes for our caption. The second image is an example where test takers could not decide whether the ground truth was better or our caption was better—the voting resulted in both captions receiving equal amount of votes. Lastly, the third image is an example where our model failed to generate any meaningful caption for the image.

Even though the results for the second survey is below 50%, if we take into account the results from the first survey and human error, we could see more improvements when comparing the two captions one-on-one. While there is room for improvement, we could consider that our model is almost equal to that of human caption generation skills.

Type	Count	Percentage
Ground Truth	277	55.40
Our Model (PCR)	223	44.60

Table 3. Results for the second survey that compared a ground truth caption from the MS-COCO dataset against a caption generated by our model.

	Verb	Adj.	Conj.	Prep.
Top-1 (%)	54.70	99.79	84.07	84.91

Table 4. The top-1 score for the parts of speech verb, adjective, conjunction, and preposition CNN model using the MS-COCO dataset.

Image	Captions (Votes by percentage)
	PCR(95%): a train is traveling down the tracks in the countryside GT(5%): a train rids in front of a city skyline
	PCR(65%): a man holding a tennis racquet on a tennis court GT(35%): a man holding a tennis racquet on top of a tennis court
	PCR(0%): a plate of food with a sandwich and a cup of coffee GT(100%): a sandwich and sauce is placed on a plate at the dinner table

Figure 14. Three questions taken from the second survey that was conducted to compare our model caption to human generated caption. The first image is a problem where our caption received the most vote. The second image is one of the questions where the test takers all voted that the quality of the captions were equal. The last image is one of the questions where our model

failed to generate a caption with any relevance to the image.

4.5.3. Automatic Evaluation

We compared our model against human generated captions (as the baseline), neighearst-neigbhor (NN) [9], neural image captioner (NIC) [35], visual concepts and back (VCB) [11], and parts of speech guidance (PSG) [14], using Flickr8k, Flickr30k, and MS-COCO datasets. The results of these models are shown in Table 5, Table 6 and Table 7.

Both Flickr8k and Flickr30k are small dataset used and are less than a ten in size compared to MS-COCO. All the machine learning based models that require a large dataset for deep learning training like NN [9], NIC [35], and our model, all fall short in score compared to the human generated caption scores in Flickr8k and Flickr30. However, NN was successful in receiving a higher score for BLEU-4 metrics than human generated caption and kept its success until our model received a higher score in the larger MS-COCO dataset.

The PoS CNN models require a lot of pretraining to insure that they can properly correlate PoS with features within an image. As seen in all metrics in Table 5-7, our model has the worst results for smaller datasets like Flickr8k. However, as the dataset increase, as seen in Flickr30k, our model only falls behind other model by a few points. In Flickr30K, our BLEU-4 score is relatively the same as NIC score. When we us the largest

dataset, MS-COCO, we can see that our model our scores all the other models from a few points like ROUGE but by two points in BLEU. For the metrics of CIDEr and SPICE, the human generated captions continually received the highest scores. However, if we take the human generated caption scores out of the picture, we come in second place. If we considering how the human generated caption scores for CIDEr and SPICE was always first, we could assume at the CIDEr and SPICE metrics are the most important. In these important metrics, scoring first in non-human generated models, our model proves its capability.

Model	CD	B4	RG	SP
HUMAN	**78.65**	29.91	**51.19**	**25.65**
NN [9]	33.30	**30.63**	42.11	–
NIC [35]	35.00	27.37	46.59	14.59
PCR	12.74	28.36	43.14	16.23

Table 5. Comparing our model against other popular methods on Flickr8k dataset using CIDEr, BLEU, ROUGE, and SPICE as the evaluation metrics. "–" indicates unknown metrics.

Model	CD	B4	RG	SP
HUMAN	**64.83**	24.3	**49.25**	**21.52**
NN [9]	24.13	**29.7**	38.80	–
NIC [35]	30.07	27.6	46.13	12.75
PSG [14]	–	21.1	–	–
PCR	27.73	26.9	45.72	12.62

Table 6. Comparing our model against other popular methods on Flickr30k dataset using CIDEr, BLEU, ROUGE, and SPICE as the evaluation metrics. "–" indicates unknown metrics.

Model	CD	B4	RG	SP
HUMAN	87.2	29.2	48.7	**23.61**
NN [9]	62.2	30.2	46.6	–
NIC [35]	79.6	33.4	52.3	20.14
VCB [11]	85.4	25.7	48.4	–
PSG [14]	**88.2**	27.9	–	–
PCR	81.6	**34.2**	**53.0**	20.64

Table 7. Comparing our model against other popular methods on MS-COCO dataset using CIDEr, BLEU, ROUGE, and SPICE as the evaluation metrics. "–" indicates unknown metrics.

4.6. Discussion

Our main contribution is the utilization of multiple CNN models to extract PoS related features from an image and feed them to a language model to improve image caption generation. While the English parts of speech has a total of eight main groups, the frequency of words used for each group differs between sentence and between the datasets as seen in Table 1. In all three datasets, nouns are the most widely used group and interjections are used so infrequently that it can be completely ignored.

The results of our experiment showed that datasets, like MS-COCO, that have a lot of training samples are more effective in training a CNN model to extract parts of speech related features compared to small datasets like Flickr8k as seen in Table 5-7. Based on these findings, we experimented with using different combinations of the PoS model such as using noun-verb, noun-verb-adjective, noun-verb-conjunction, noun-verb-pronoun and so on. From our test, we concluded that the best combination was using only five of the eight different PoS: noun-verb-adjective-conjunction-preposition (NVACP). Ironically, these five parts of speech groups had the highest number of words used in all three datasets as seen bolded in Table 1. The top-1 score for each of the model are available in Table 4.

Furthering this train of thought, as mentioned in section 4.2, we wanted to reduce any factors to overfitting. An additional

approach we decided to use was to replace the noun CNN model in NVACP combination with a simple ResNet-152 [13] pretrained [AP] object detecting model. We saw that the majority of the features detect by the noun CNN model were closely resembling features detect by an object detecting CNN model. Amazingly, by replacing the noun model with an object model we were able to see that the combination, object-verb-adjective-conjunction-preposition (OVACP), showed minor to no difference at all compared to NVACP combination in our test. Two important facts are learned from these results. The first fact is that a noun based CNN model will in fact extract features similar to object detecting CNN models. The second fact, which supports our approach, is that PoS CNN models are in fact extracting features from an image correctly.

Chapter 5. Conclusion

In this paper, we have presented a novel approach in creating an image caption that are as good as human generated caption. Our approach uses multiple CNN models to detect parts of speech related features and use a LSTM based language model to generate our caption. Through the different evaluation methods discussed in section 4, we believe that we have been able to defend our statement of generate good image captions.

In our future work, we aim to introduce an *attention* mechanism [36] to be able to focus exactly where the parts of speech features are within the image which will result in higher accuracy and less noise.

Secondly, we would like to improve the generation of the parts of speech dataset. Currently, the dataset is generated using the definition of the words part of speech. However, we would like to detect the parts of speech of the word in context to the sentence.

Last, we are only detecting PoS features and use them as cue’s to the language model but we would like to incorporate the into a more natural language processing manager because of its inherit nature.

In conclusion, our proposal has not solved all the issues related to image captioning and still has room to improve. However, we believe that our work is core stepping stone for future works.

Bibliography

[1] J. Aneja, A. Deshpande, A. Schwing. Convolutional image captioning. In CVPR, 2018.

[2] P. Anderson, B. Fernando, M. Johnson, and S. Gould. SPICE: Semantic propositional image caption evaluation. In ECCV, 2016.

[3] S. Bai and S. An. A survey on automatic image caption generation. In Neurocomputing, 2018.

[4] S. Bird and E. Loper. NLTK: the natural language toolkit. In ACL, 2002.

[5] X. Chen and C. L. Zitnick. Mind' s eye: a recurrent visual representation for image caption generation. CVPR, 2015.

[6] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In EMNLP, 2014.

[7] J. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In NIPS 2014.

[8] B. Dai, S. Fidler, and D. Lin. A neural compositional paradigm for image captioning. In NIPS, 2018.

[9] J. Devlin, S. Gupta, R. Girshick, M. Mitchell, and C. L. Zitnick. Exploring nearest neighbor approaches for image captioning. arXiv preprint arXiv: 1505.04467, 2015.

[10] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan,

S. Guadarrama, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In CVPR, 2015.

[11] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollar, j.Gao, X. He, M. Mitchell, J. C. Platt, et all. From captions to visual concepts and back. In CVPR, 2015.

[12] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. In ECCV, 2010.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2015.

[14] X. He, B. Shi, X. Bai, G. S. Xia, Z. Zhang, and W. Dong. Image caption generation with part of speech guidance. In PRL, 2019.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. In Neural Comput., 1997.

[16] M. Hodosh, P. Young and J. Hockenmaier. Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. In JAIR, 2013.

[17] X. Jia, E. Gavves, B. Fernando, and T. Tuytelaars. Guiding the Long-Short Term Memory model for Image Caption Generation. In ICCV, 2015.

[18] M. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In Cognitive Science Conference, pp 531-546, 1986.

[19] D. Kingma, and J. Ba. Adam: a method for stochastic optimization. In ICLR, 2015.

[20] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. In CoRR vol. abs/1404.5997, 2014.

[21] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Chaoi, A. C. Berg, and T.L. Berg. Baby talk: Understanding and generating simple images descriptions. In CVPR, 2011.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In Proceedings of the IEEE (Volume: 86 , Issue: 11), 1998.

[23] C. Lin. ROUGE: a package for automatic evaluation of summaries. In ACL, 2004.

[24] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: common objects in context. In CVPR, 2014.

[25] S. Loffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[26] J. Mao, W. Xu. Y. Yang, J. Wang, and A. L. Yuille. Explain images with multimodal recurrent neural networks. arXiv preprint arXiv:1410.1090, 2014.

[27] K. Papineni, S. Roukos, T. Ward, W. J. Zhu. BLUE: a method for automatic evaluation of machine translation. ACL, 2002.

[28] L. Prechelt. Early stopping – but when?. In Neural Networks, 1998.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. F. Fei. ImageNet large scale visual recognition challenge. In IJCV, 2015.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ACPR, 2015.

[31] R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: visual explanations from deep networks via gradient-based localization. In ICCV, 2017.

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In CVPR, 2016.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015

[34] R. Vedantam, C. L. Zitnick, and D. Parikh. CIDEr: Consensus-based Image Description Evaluation. In CVPR, 2015.

[35] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell : A neural image caption generator. CVPR, 2015.

[36] K. Xu, J. Ba, R. Kiro, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In ICML, 2015.

[37] Y. Xu, and R. Goodacre. On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization

performance of supervised learning. In JAT, 2018.

[38] Y. Yang, C. L. Teo, H. D. III, and Y. Aloimonos. Corpus-guided sentence generation of natural images. In EMNLP, 2011.

[39] T. Yao, Y. Pan. Y. Li, Z. Qiu, and T. Mei. Boosting image captioning with attributes. In CVPR, 2016.

[40] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. In CVPR, 2016.

[41] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. In ACL, 2014.

Appendix

A. Preprocessing

NLTK subgroups organized into eight parts of speech

```
nltk_to_pos = {
    "noun"          : ["CD", "NN", "NNS", "NNP", "NNPS", "PDT"],
    "pronoun"       : ["CD", "PRP", "PRP$", "WP", "WP$"],
    "verb"          : ["VB", "VBD", "VBG", "VBN", "VBP", "VBZ"],
    "adjective"     : ["CD", "DT", "JJ", "JJR", "JJS"],
    "adverb"        : ["CD", "RB", "RBR", "RBS", "WRB"],
    "conjunction"   : ["CC", "IN"],
    "preposition"   : ["IN"],
    "interjection"  : ["UH"]
}
```

Preparing Parts of Speech Dataset Using NLTK

```
for path in [COCO_TRAIN_ANNOTATION, COCO_VALIDATION_ANNOTATION]:

    with open(path) as f:
        coco_annotation = json.load(f)

    coco_image = { }
    for item in coco_annotation["images"]:
        coco_image[item["id"]] = item

    for anno_item in tqdm(coco_annotation["annotations"]):

        # retrieve item information
        caption = nltk.tokenize.word_tokenize(anno_item["caption"].lower())
        filename = coco_image[anno_item["image_id"]]["file_name"]

        # check if already added
        if filename not in complete_dataset:
            complete_dataset[filename] = [ ]

        complete_dataset[filename].append(caption)
```

Resize image for RESNET−152

```
def resize_images(image_dir, output_dir, size):
    """Resize the images in 'image_dir' and save into 'output_dir'."""
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    images = os.listdir(image_dir)
    num_images = len(images)
    for i, image in enumerate(images):
        if image == "README.md": continue
        with open(os.path.join(image_dir, image), 'r+b') as f:
            with Image.open(f) as img:
                img = resize_image(img, size)
                img.save(os.path.join(output_dir, image), img.format)
        if (i+1) % 100 == 0:
            print ("[{}/{}] Resized the images and saved into '{}'."
                   .format(i+1, num_images, output_dir))

```

B. Configuration

Data Loader

```
# ##################################
# DATA LOADER
# ##################################

# number of process worker (load data from disk to gpu)
number_of_workers =  32
# number_of_workers =  0 # for bidirectional

# data loader shuffle dataset
is_shuffle = False

# size of each training cycle
pretrain_batch_size = 128

# size of each training cycle
train_batch_size = 192
# train_batch_size = 16 # for bidirectional
```

Pretraining & Training

```
# number of epochs
cnn_rnn_number_epochs = 1000

# image random crop size
image_crop_size = 224

# number of epochs
pretrain_number_epochs = 9

# log step size
log_step = 5
```

CNN & RNN Model

```
# ##################################
# Convolutional Neural Network
# ##################################

# fully connected layer output size
cnn_output_size = 1024

# ##################################
# Recurrent Neural Network
# ##################################

# Using bi-direction RNN
use_bi_direction_lstm = False

# LSTM input size
rnn_embed_size = 1024 # 5120

# dimension of lstm hidden states
rnn_lstm_hidden_size = 512

# number of layers in lstm
rnn_lstm_number_of_layers = 3

# generation sentence method
rnn_inference = "sample" # "sample" "beam_search"

# number of beam search K
rnn_beam_search_width = 3

# dropout
rnn_lstm_dropout = 0
```

C. Different Config. Used for Our Model and the Results

#	EVALUATION					Pretrain		Train			CNN					RNN (LSTM)						
	CD	B1	B2	B3	B4	RG	Epoch	Zero_grad	PoS	Learning Rate	Batch Size	Model	Combine Method	Combine Method ID	Back Propagation	Output Size	Layer Number	Hidden Node	Inference	Dropout	K	KA	Bi direct
1	0.796272544	0.21347316	0.332151020	0.252366286	0.331470668	0.5234259011	5			0.01	128	resnet-152	None		All	256	1	512	Sample	0			
2	0.791670608	0.21738229	0.337020760	0.256719113	0.336048315	0.5260244059	8	Y	VACP	0.004	128	resnet-152	Avg	2	All	1024	3	512	Sample	0			N
3	0.473443969	0.12328962	0.249946234	0.166284369	0.270353647	0.3967360525	8	Y	VACP	0.004	128	resnet-152	Avg	2	All	1024	3	512	Beam	0	3	Mult	N
4	0.716726970	0.19705466	0.318611238	0.237196121	0.322309429	0.5081997709	8	Y	VACP	0.004	128	resnet-152	Concate	1	All	5120	3	512	Sample	0			N
5	0.776275920	0.20911834	0.329574983	0.248779200	0.329568691	0.5212391412	8	Y	VACP	0.004	128	resnet-152	LSTM-A	3	RN	1024	3	512	Sample	0			N
6	0.749723089	0.20606381	0.328011976	0.246480993	0.329260936	0.5211297937	8	Y	VACP	0.004	128	resnet-152	LSTM-A	3	RN	1024	3	512	Sample	0.3			N
7	0.706197725	0.19921696	0.320370954	0.238648267	0.324577857	0.5076235350	8	Y	VACP	0.004	128	resnet-152	Avg	2	RN	1024	3	512	Sample	0.5			N
8	0.831983509	0.22557089	0.343677721	0.264037246	0.339866190	0.5287841047	8	Y	VACP	0.0004	128	resnet-152	Avg	2	RN	1024	3	512	Sample	0			N
9	0.752340709	0.20914603	0.331476014	0.249556903	0.330087468	0.5196283929	8	Y	VACP	0.00004	128	resnet-152	Avg	2	RN	1024	3	512	Sample	0			N
10	**0.816869175**	**0.22653704**	**0.345793694**	**0.265499199**	**0.342743936**	**0.5300964000**	**8**	**Y**	**VACP**	**0.0001**	128	**resnet-152**	**Avg**	**2**	**RN**	**1024**	**3**	**512**	**Sample**	**0**			**N**
11	0.825459011	0.22570801	0.345076493	0.264757986	0.341091453	0.5307003872	8	Y	VACP	0.00009	128	resnet-152	Avg	2	RN	1024	3	512	Sample	0			N
12	0.810483968	0.22100235	0.341190090	0.260650389	0.338886269	0.5308324325	8	Y	VACP	0.000095	128	resnet-152	Avg	2	RN	1024	3	512	Sample	0			N
13	0.536240437	0.16532307	0.292019960	0.207803599	0.305737354	0.4888666999	8	Y	VACP	0.00001	128	resnet-152	Avg	2	RN	1024	3	512	Sample	0			N
14	**0.817863004**	**0.22464196**	**0.343094490**	**0.263644709**	**0.340167310**	**0.5381168306**	**8**	**Y**	**VACP**	**0.0001**	128	**resnet-152**	**LSTM-A**	**3**	**RN**	**1024**	**3**	**512**	**Sample**	**0**			**N**
15	0.824549403	0.22333443	0.342923996	0.262714046	0.340032580	0.5301207205	8	Y	VACP	0.0001	128	resnet-152	Avg	2	All	1024	3	512	Sample	0			N
16	0.602781297	0.18126385	0.306776550	0.223341602	0.314096857	0.5011603139	8	Y	VACP	0.00001	128	resnet-152	Avg	2	All	1024	3	512	Sample	0			N
17	0.818334617	0.22421481	0.345028132	0.264076760	0.341258732	0.5279562066	8	Y	VACP	0.0001	128	resnet-152	Avg	2	RNV	1024	3	512	Sample	0			N
18							8	Y	VACP	0.0001	0	resnet-152	Avg	2	All	1024	3	512	Sample	0			Y
19	0.808769540	0.22120782	0.341567172	0.260744443	0.339368797	0.5275175395	8	Y	VACP	0.0001	128	resnet-152	Linear	4	All*	1024	3	512	Sample	0			N
20	0.813799931	0.22068755	0.340659821	0.260263478	0.338511140	0.5267877166	8	Y	VACP	0.0001	128	resnet-152	Linear	4	RN	1024	3	512	Sample	0			N
21	0.793654734	0.22329305	0.342502300	0.262006999	0.338935686	0.5287329755	8	Y	VACP	0.0001	128	resnet-152	Avg	2	RN	1024	3	512	Sample	0.2			N
22	0.818703693	0.22420620	0.344428629	0.263895630	0.340975942	0.5316284396	8	Y	VACP	0.0001	128	resnet-152	Avg	2	RNVC	1024	3	512	Sample	0			N
23	0.815232460	0.22420718	0.343237669	0.263133215	0.340666851	0.5312523784	8	Y	VACP	0.0001	128	resnet-152	Avg	2	RN	1024	3	512	Sample	0			N
24	0.666105275	0.18971272	0.313797578	0.230622200	0.316752485	0.5006146282	8	Y	VACP	0.0001	128	resnet-152	Linear2	5	RN	1024	3	512	Sample	0			N
25	0.768056399	0.21311100	0.331943703	0.251945232	0.331674475	0.5216087756	8	Y	VACP	0.0001	256	resnet-152	Avg	2	RN	1024	3	512	Sample	0			N
26	0.820042866	0.22417628	0.344533963	0.263949731	0.340782101	0.5244452006	8	Y	VACP	0.0001	64	resnet-152	Avg	2	RN	1024	3	512	Sample	0			N
27	0.872219701	0.17054994	0.292287313	0.211712499	0.300983185	0.4874965435																	
28	0.668194118	0.19188388	0.315142276	0.232468137	0.321966785	0.5014997369	8	Y	VACP	0.0001	64	resnet-152	Linear2*	5	RN	1024	3	512	Sample	0			N
56	0.812233459	0.22258373	0.341674110	0.261640837	0.338826976	0.5290360573	8	Y	VACP	0.0001	192	resnet-152	Avg	2	RN	1024	3	512	Sample	0			N

D. List of Example Captions Generated by Our Model & Ground Truth (GT)

GT: Two giraffes standing side by side at a zoo in front of a large city.
Our Model: A giraffe standing next to a tree in a zoo.

GT: A large cat sitting with its back turned by another smaller cat.
Our Model: A brown chair sitting in a living room next to a window.

GT: A blue umbrella sitting on a sandy beach
Our Model: A person is walking on the beach with a surfboard.

GT: There is a train that is stopped at the station.
Our Model: A train is parked at a train station.

GT: A woman sitting on the ground holding a purple stripped umbrella.
Our Model: a woman is holding a umbrella in the rain.

GT: Person perfecting their flip on water ski board.
Our Model: A man riding a wave on top of a surfboard.

GT: A train is pulling into the train station
Our Model: a train is traveling down the tracks in the city

GT: A traffic light sitting by the side of a road.
Our Model: A traffic light with a red light on it.

GT: A train on track passing through a small station.
Our Model: A train is traveling down the tracks in the country.

GT: A white bathroom sink sitting next to a tub.
Our Model: A bathroom with a sink, toilet, and mirror.

GT: A man is feeding a giraffe while other birds are in the background.
Our Model: a giraffe is standing next to a tree.

GT: A man posing on skiis in the snow.
Our Model: a man is skiing down a snowy hill.

초록

영어 품사 정보를 활용한 이미지 캡션 생성 모델

강필구

컴퓨터공학부

공과대학

서울대학교

스마트 기기와 일상 생활에서의 인공 지능에 대한 의존도가 증가해가면서, 이미지를 스스로 설명하는 기술의 중요성이 점점 더 증가하고 있다. 본 논문에서는 CNN 모델을 명사, 형용사, 전치사와 같은 영어 품사별로 나누어 학습하여 품사 별로 구분되는 특징을 학습하고, 학습된 결과를 활용하여 이미지를 설명할 수 있는 문장을 생성하는 방법을 제안한다. 품사 별로 학습된 CNN 모델에서는 품사별로 구별되는 시각적 특징 벡터들을 추출하고, 추출한 특징 벡터들을 합성하여 언어 모델에서 좋은 설명문을 생성하는데에 활용된다. 본 논문에서는 해당 분야에서 널리 사용되는 있는 Flickr8k, Flickr30k 그리고 MS-COCO 데이터 셋에 대한 실험을 통해 제안하는 모델의 우수성을 검증하였다. 또한, 사람들을 대상으로 한 설문 조사를 진행하여 제안한 모델에서 사람이 이해하기에 충분한 문장을 생성하는 것을 확인하였다.

주요어: 이미지 캡션, 이미지 설명문, 이미지 처리, 컴퓨터 비전
학 번: 2018-22788