
Markov vs Neural Network: A Comparative Study of Classic and Modern Models for Image Captioning

Zhe Chen[†], Fan Shi^{*}, Sisi Gai[◊]

Computer Science Department[†], Language Technology Institute^{*}, Heinz College[◊]
Carnegie Mellon University
Pittsburgh, PA 15213
{zhechen2, fans, sgai}@andrew.cmu.edu

Abstract

Automatic image captioning has long been studied in literature and several iterations of algorithms, ranging from traditional template-based to the state-of-the-art deep learning with attention mechanism, has been proposed. In this paper, we conducted a comparative study of a classic model and a modern model. We enhanced a Markov chain for image caption as the classic model and chose an encoder-decoder model using ResNet-50 CNN and LSTM respectively as the modern model. Our experiments showed that our modern model outperformed our classic model by a large margin for image captioning, although both models are able to generate fluent and relevant captions to the scene. Our modern model also has a similar performance to literature using a similar architecture.

1 Introduction

Automatic generation of natural language descriptions based on salient objects in an image has long been studied in artificial intelligence. This task requires a combination of computer vision and natural language processing techniques to complete, making it interesting yet challenging. It has extensive and significant applications such as automatic caption generation for images for people who suffer from visual impairment[2]. Though for humans, to prepare a concise sentence for an image is relatively easy and natural, for machines, this task is rather challenging. The machine must not only detect objects in the image, but also understand the relationship between objects, before expressing it in fluent language.

Early works [1, 6, 12, 13] mainly used the classical retrieval-and-template-based approaches. The subject, verb and object are detected separately and then joined using a sentence template[2]. However, most of these works have not been tested on complex everyday images where the large variations of objects and poses makes it nearly impossible to learn a more general model. The advent of deep learning promoted rapid development of computer vision and a major breakthrough in image captioning occurred in 2014 when Vinyals et al. [15] introduced an encoder-decoder pipeline model. After that, various adjustments[5, 17, 18, 21] on network structure has been made and achieved outstanding results. However, it is hard to compare different works due to the diverse combination of structures, parameters and datasets used for evaluation.

In this paper, we conducted a comparative study of a classic machine learning model and a modern deep learning model. The classic model is an enhanced Markov Chain and the deep learning model uses CNN-LSTM. We analyzed the advantages and shortcomings of these two methods and compare their results on the same COCO 2014 dataset[14].

Contributions. Our main contributions are: 1. the novel enhancements for a regular Markov chain such that the sentence generation is guided by the salient image features. 2. The comparison of

performance, both in terms of metric scores and human judgement, between the classic model and modern model.

This paper is organized as follows. The third and the fourth parts detail the machine learning and deep learning models. The fifth part introduces the COCO dataset and compares the results on different models using various evaluation methods. The sixth part summarizes the work and proposes the direction and expectations of future improvement work.

2 Related Work

In recent years, many machine learning researchers began to try to mine information from the image and automatically generate the description of the image. Looking back at the development of Image caption assignment, models can be divided into two main categories: a machine learning method based on a statistical probability language model to generate handcraft features and a neural network model based on an encoder-decoder language model to extract deep features[16]. Yang et al. [20] propose a language model trained from the English Gigaword corpus to obtain the estimation of motion in the image and the probability of colocated nouns, scenes, and prepositions and use these estimates as parameters of the hidden Markov model. The image description is obtained by predicting the most likely nouns, verbs, scenes, and prepositions that make up the sentence.

Bengio and Frasconi [3] introduced a new HMM-based structure called Input-Output HMM. An IOHMM is an HMM for which the output and transition probabilities are conditional on the input sequence, using the same processing style as recurrent neural networks. IOHMMs are trained using a more discriminant learning paradigm than HMMs, while potentially taking advantage of the expectation-maximization(EM) algorithm. There have been quite a few results[4, 19] demonstrating that IOHMM can generate somewhat natural and good-quality sequences given audio or video input. But IOHMM used on image is quite uncommon.

The mainstream deep learning model structure uses a combination of convolutional neural networks to obtain vectorial representation of images and recurrent neural networks to decode those representations into natural language sentences. The state-of-art techniques[15, 5, 17, 18, 21] usually made some adjustments on the network or introduce some new tricks to improve the performance. Vinyals et al. [15] first migrated the encoder-decoder structure from the machine translation task to image caption, which is considered to be the pioneer of image caption task. Later on, Xu et al. [18], Wu et al. [17] and Chen and Zitnick [5] made improvements on the basis of this paper. These models are quite good, but have their own constraints. Image captioning still have a long way to go in improving the accuracy of captioning the events in images.

3 Markov-based Model

3.1 Markov Model of Language

Generally [11], given a string of English words $\mathbf{W} = w_1, w_2, \dots, w_n$, its probability can be decomposed as $\mathbb{P}(W) = \mathbb{P}(w_1)\mathbb{P}(w_2 | w_1)\mathbb{P}(w_3 | w_1, w_2) \dots \mathbb{P}(w_n | w_1, w_2, \dots, w_{n-1})$. By applying the Markov assumption, this expression can be simplified as $\mathbb{P}(w_1)\mathbb{P}(w_2 | w_1) \dots \mathbb{P}(w_n | w_{n-1})$. To greedily generate the next word, we can iterative find the word w_i through $\arg \max_{w_i} \mathbb{P}(w_i | w_{i-1})$. To find the best n words, we should find the words that lead to $\arg \max_{w_1 \dots w_n} \mathbb{P}(w_1)\mathbb{P}(w_2 | w_1) \dots \mathbb{P}(w_n | w_{n-1})$. As such, this language model is equivalent to a Markov Chain model with each node being a unique word observed in training.

This language model can be generalized with n-gram[13] such that each new word depends on the preceding n-1 words. It has been suggested [11] that trigram (3-gram) is most commonly used. However, capturing n-gram frequencies using a dense n-dimensional matrix can use up a lot of computer memory. Capturing trigrams of 2000 unique words, for example, would require a matrix of size 2000^3 . Assuming each entry takes up 8 bytes of space, this matrix would end up taking 64 gigabytes of memory. This requirement is not scalable. Intuitively, a sparse matrix can alleviate this limitation, as most of word combinations would not be used in the English language at all. Our preliminary testing confirmed this intuition, as our trigram matrix with 9994 unique words only took up 36.6 megabytes of memory with a density of 1.2×10^{-6} .

3.2 Enhanced Markov Model for Captioning

3.2.1 Special Tokens

We introduced the modern concept of special tokens into training an n-gram Markov model, and the following tokens are added into the tokenized training captions:

1. <start>: n-1 such tokens are prepended to the tokenized captions to allow the Markov model to learn the starting word of a sentence $\mathbb{P}(w_1 \mid \text{<start>} \times (n-1))$ from training data instead of relying on the most common word $\mathbb{P}(w_1)$ that may not be appropriate as the starting word of a sentence.
2. <end>: 1 such token is appended to the tokenized captions. This allows the Markov model of language to learn how to end a sentence. Similar to modern models, sentence generation stops after this token is generated.
3. <unk>: This token replaces words that appear less than a threshold in all training captions. This would reduce the number of unique words to be learnt by the model.

3.2.2 Feature-guided Sentence Generation

Given m features (denoted by x) extracted from the image, we can condition the new word to be generated on not only the preceding word(s) but also on all features, such that the selected word will not only be fluent but also relevant to the current features. For example, probability of word i in a sentence using trigram (3-gram) Markov model can be written as $\mathbb{P}_f(w_i) = \mathbb{P}(w_i \mid w_{i-1}, w_{i-2}, x_1, x_2, \dots, x_n)$. Equation 1 breaks down $\mathbb{P}_f(w_i)$ using Naïve Bayes assumption. This equation also allows us to numerically estimate $\mathbb{P}_f(w_i)$ instead of ignoring its denominator when maximizing the posterior probability, as is the case of a regular Naïve Bayes classifier. As such, we can use beam search instead of greedy search by heuristically finding a sequence of tokens w_1, \dots, w_n that satisfies $\arg \max_{w_1, \dots, w_n} \prod_{i=1}^n \mathbb{P}_f(w_i)$.

We first observed that to calculate equation 1, the parameters to learn from the training captions are $\mathbb{P}(w_i)$, $\mathbb{P}(x_j \mid w_i)$, $\mathbb{P}(w_{i-2}, w_{i-1} \mid w_i)$ and $\mathbb{P}(w_{i-1} \mid w_i)$ for all unique words w_i and features x_j . Assuming that all features are binomial, multinomial or Gaussian distributions, these parameters can be learned through MLE with a close-formed solution. We also observed an interesting property of Equation 1 in that when trying to find the best sequence of words, the conditional probabilities of features given intermediate words (only w_i in the case of Equation 2) cancel out entirely. Intuitively, this limits the weight that all given features can place on sentence generation, such that the probability of a new word is also jointly controlled by the underlying Markov model of language.

$$\begin{aligned}
 \mathbb{P}_f(w_i) &= \mathbb{P}(w_i \mid w_{i-1}, w_{i-2}, x_1, x_2, \dots, x_n) \\
 &\stackrel{\text{Bayes' Rule}}{=} \frac{\mathbb{P}(w_{i-1}, w_{i-2}, x_1, x_2, \dots, x_n \mid w_i) \mathbb{P}(w_i)}{\mathbb{P}(w_{i-1}, w_{i-2}, x_1, x_2, \dots, x_n)} \\
 &= \frac{\mathbb{P}(w_{i-1}, w_{i-2}, x_1, x_2, \dots, x_n \mid w_i) \mathbb{P}(w_i)}{\mathbb{P}(w_{i-2}, x_1, x_2, \dots, x_n \mid w_{i-1}) \mathbb{P}(w_{i-1})} \\
 &\stackrel{\text{Naïve Bayes}}{=} \frac{\mathbb{P}(w_{i-1}, w_{i-2} \mid w_i) \mathbb{P}(w_i) \prod_{j=1}^m \mathbb{P}(x_j \mid w_i)}{\mathbb{P}(w_{i-2} \mid w_{i-1}) \mathbb{P}(w_{i-1}) \prod_{j=1}^m \mathbb{P}(x_j \mid w_{i-1})}
 \end{aligned} \tag{1}$$

$$\begin{aligned}
 &\arg \max_{w_i, w_{i+1}} \mathbb{P}_f(w_i) \mathbb{P}_f(w_{i+1}) \\
 &= \arg \max_{w_i, w_{i+1}} \frac{\mathbb{P}(w_{i-1}, w_{i-2} \mid w_i) \mathbb{P}(w_{i-1}, w_i \mid w_{i+1}) \mathbb{P}(w_{i+1}) \prod_{j=1}^m \mathbb{P}(x_j \mid w_{i+1})}{\mathbb{P}(w_{i-2} \mid w_{i-1}) \mathbb{P}(w_{i-1}) \mathbb{P}(w_{i-1} \mid w_i) \prod_{j=1}^m \mathbb{P}(x_j \mid w_{i-1})}
 \end{aligned} \tag{2}$$

For the image captioning task, we encode the object categories and locations in the image as binary features. Figure 2 shows the workflow of our enhanced model. This would make the resulting sentence relevant to the image and hopefully provide a good description of the image content. To encode the object categories and locations, we first divide each image into n-by-n grids, grid size n being a tunable parameter. Then we iterate through the bounding boxes of each type of object



Ref: a hot dog and an ear of corn are on a plate
No decay: a hot dog with mustard and ketchup on a bun covered in ketchup and mustard hotdog
Decay = 10^{-2} : a hot dog with ketchup and mustard on top of it

Figure 1: Word Probability Decay Example

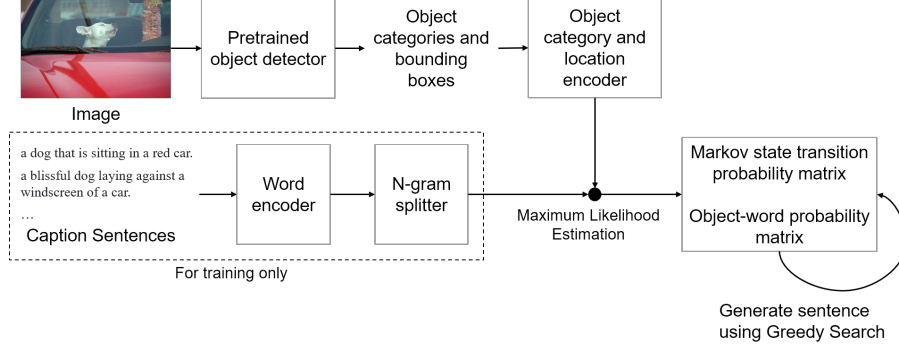


Figure 2: Workflow of Markov-based Model

and intersecte them with each grid. If any of the object of category o^i intersected a grid g^j , the corresponding feature $x_{o^i, g^j} = 1$.

As will be explained in section 5.2, the added `<end>` token appears more frequently than most regular words, causing the maximization to often end the sentence early with this token. To resolve this, we impute the frequency of `<end>` to be the mean frequency of regular words when calculating $\mathbb{P}(w)$.

3.2.3 Word Probability Decay

Our earlier experiments showed that the Markov model has a tendency to generate repeated phrases for a salient object, and this tendency didn't change after implementing beam search. Figure 1 shows such an example. After thorough analysis, we concluded that this behavior is caused by the relatively large number of relevant words for a given object category. As shown in Figure 1, there are many descriptive words related to a hot dog, such as "hot", "dog", "bun", "hotdog" (common misspelling), "mustard" and "ketchup". Varied usages of these words caused the Markov model to learn strongly linked loops among these words such that these words get repeated during sentence generation.

To counter this issue, we added a tunable penalty for repeating the same word, named Word Probability Decay D_w . This penalty term $n \log D_w$, where n is the number of times that the particular word appeared, is added to the probability of each candidate sentence. As is shown in the same figure, adding some penalty cleaned up the repeated words, resulting in a more fluent sentence.

4 Neural Network Model

4.1 Encoder: CNN-based Feature Extractor

Inspired by the usage paradigm shown in [10, 15, 18], we use pre-trained CNN image classifier as image encoder and remove the last fully-connected layer for classification. The encoder takes pre-processed image as input and extracts a fixed-length feature vector which corresponding to information about each part of image. We choose ResNet-50 [7], the official benchmark of MLPerf, as our CNN encoder. Its residual learning framework is easier to optimize, and can gain accuracy from considerably increased depth.

4.2 Word Embedding

Thousands of words are used to generate image captions so that we use word embeddings instead of one-hot encoding to represent captions which can reduce data dimensions input to decoder. The

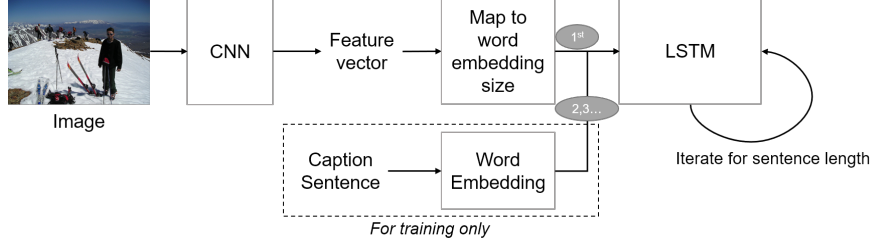


Figure 3: Workflow of the CNN-LSTM Model

word-embedding will learn language features from scratch where similar words have shorter distance in embedding space. Also, we map the feature vectors of images to the embed space and serve as the first input to decoder. We choose 512 as embed size which has been stated in [15].

4.3 Decoder: LSTM-based Sentence Generator

We use long short-term memory (LSTM) network as decoder to process the sequential caption data. Caption input can be formulated as $S = (S_0, \dots, S_N)$ where S_0 and S_N are special tokens designating the start and end of the sentence. We use the insights of MLE where the log-likelihood function of a correct image description can be represented as:

$$\log p(S|I) = \sum_{t=0}^N \log p(S_t|I; S_0, \dots, S_{t-1}) \quad (3)$$

Notice the probability of a caption word only depends on the image feature vectors and preceding words. This probability will be computed by an LSTM cell. The definition of the three gates (forget gate f , input gate i and output gate o) and cell updates are as follow:

$$\begin{aligned} i_t &= \sigma(W_{ix}x_t + W_{im}m_{t-1}) & f_t &= \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \\ o_t &= \sigma(W_{ox}x_t + W_{om}m_{t-1}) & c_t &= f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \\ m_t &= o_t \odot c_t & p_{t+1} &= \text{Softmax}(m_t) \end{aligned}$$

where \odot is the product with a gate value, W s matrices are trained parameters, sigmoid $\sigma(\cdot)$ and hyperbolic tangent $h(\cdot)$ are non-linearities and output p_t is a probability distribution over all words.

4.4 Model structure and Inference Step

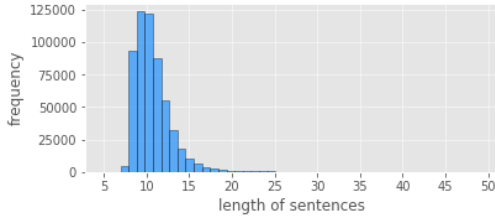
The model will take a transformed image as input, compute feature vectors and then map embedded space. We only feed this image feature vector once to LSTM at step $t = -1$. Then the following step will take captions embedding as inputs and produce the likelihood of each word at each time step. The model workflow visualized as in Figure 3.

For inference, We firstly set LSTM initial to all zeros and feed in the image feature vector and infer the next word based on likelihood. We iteratively use picked features and state from last timeline $t - 1$ to produce the candidates in current timeline t . Greedy search and beam search are used to pick top targets from word candidates.

5 Experiments

5.1 Data and Preprocessing

We conducted experiments on the widely-used large-scale object detection, segmentation, and captioning dataset, COCO 2014 [14]. This dataset consists of 82,783 training, 40,504 validation, and 40,775 testing images. There are 80 common object categories, more than 1.5 million labeled instances and 5 human-submitted captions to each image. We used the offline ‘‘Karpathy’’ split [9] for performance comparison. This split uses 10,000 images from the COCO 2014 validation set and split them into 5,000 test images and 5,000 validation images. The rest of the validation and the training set images are combined for training both models.



From the left histogram, we can see that long-length sentences are not common. Too long sentences would not carry much more information but are wordy, meaningless repetition most of the time. Also, they would affect the performance of the model. It is sensible to limit the maximum length of sentences to 16.

Figure 4: Histogram of Length of Sentences

Table 1: Performance of selected modern models on COCO “Karpathy” test split, where B@N, M, R, C and S are short for BLEU@N, METEOR, ROUGE-L, CIDEr-D and SPICE scores. All values are reported as percentages (%)

Model	B@1	B@4	M	R	C	S
LSTM [15]	-	29.6	25.2	52.6	94.0	-
AoANet [8] (Cross-Entropy Loss)	77.4	37.2	28.4	57.5	119.8	21.3

5.2 Markov-based

Table 2 summarizes the scores of a fully-trained Markov-based image captioner. To simplify the pipeline, we directly used the official annotation data that contains hand-labeled object categories and locations within each image. By observing this table, we can see that 4-gram models perform better on BLEU-4 metric as compared to trigram models. This is expected as BLEU-4 checks specifically for matching 4-grams. Also, increment of grid size actually drops the performance of the model. This is likely due to the fact that the likelihood of an object appearing in each grid is not fully independent of that of other grids. This violates the Naïve Bayes assumption used to derive the Markov model. In terms of beam width, increasing it definitely helps to improve model performance but slows down the caption generation speed. We also observed that these scores are lower than our experiments with a previous version of the code that didn’t calculate the $\langle \text{end} \rangle$ token probability properly and always generate sentences until the defined word limit of 16 (See analysis in Figure 4). So we compared the resulting captions from these 2 sets of experiments. This comparison showed that the caption generated by the corrected newer code is much shorter and a lot of sentences are terminated before fully describing the scene. Thus, we suspected that this is due to the high word appearance probability $\mathbb{P}(w)$ of the $\langle \text{end} \rangle$ token since it is artificially appended to every sentence, which caused the model to have a higher chance of generating the $\langle \text{end} \rangle$ token and subsequently terminating the sentence.

Having found the suspected root cause, we imputed the frequency of the $\langle \text{end} \rangle$ token to be the average of all regular words and ran another series of experiments using the most promising settings from Table 2. Table 3 tabulates the scores of the improved model, and we see an immediate improvement over the previous experiments. This also showed that while setting a smaller word probability decay would improve some metrics, it could also hurt other metrics, especially metrics that looks for longer n-grams, such as BLEU-4 and ROUGE-L.

Comparing to models listed in Table 1, the Markov-based model performed much worse. This is not unexpected, as the Markov-based model received insufficient features (e.g. there are no color information) and made relatively strong assumptions on those features. Also, the absolute location of an object encoded for the Markov-based model may not be directly related to the captions, since it’s the relative position of the objects and their interactions that matters more.

5.3 CNN-LSTM

We use two methods to sample the data. Method 1 pads captions to the fix length of 16 then iterate random batch from the whole training set. Method 2 does not pad shorter sentences. It first chooses a random caption length for each batch and then sample a random batch of training captions with exactly that length. Experiments were conducted on single Tesla-V100 GPU using batch size of 512.

Table 4 summarizes the performance of CNN-LSTM model after different training epoch with several beam-width sampling methods. During training process, we use fixed learning rate without decay

Table 2: Performance of the Markov model on COCO “Karpathy” validation split under various combination of parameters. Notations are the same as Table 1

N-gram	Grid Size	Beam Width	Decay	B@1	B@4	M	R	C	S
3	2	20	10^{-4}	35.6	4.0	13.8	25.5	27.9	8.9
4	4	20	10^{-2}	35.0	5.4	13.5	25.2	27.3	9.4
4	4	20	10^{-4}	34.7	5.2	13.5	25.3	28.1	9.3
3	2	20	10^{-4}	34.5	5.5	13.6	27.0	29.4	9.0
3	2	5	10^{-1}	34.0	4.0	13.6	25.2	25.2	9.1
3	4	20	10^{-4}	31.6	2.2	14.9	23.6	22.9	10.1
4	2	20	10^{-2}	31.5	6.8	12.9	28.1	30.4	8.6
3	4	5	10^{-4}	31.2	1.8	14.5	22.7	21.6	9.6
4	2	20	10^{-4}	31.0	6.7	12.7	27.9	30.0	8.5
3	6	5	10^{-4}	29.5	1.6	14.6	22.3	18.8	10.0
3	4	5	10^{-1}	28.2	1.9	13.9	22.3	18.5	9.8
3	6	5	10^{-2}	28.0	1.8	14.2	22.0	17.6	10.0
3	6	5	10^{-1}	26.6	1.8	13.9	21.8	16.2	9.9

Table 3: Performance of the improved Markov model on COCO “Karpathy” validation split. Notations are the same as Table 1

N-gram	Grid Size	Beam Width	Decay	B@1	B@4	M	R	C	S
4	2	20	10^{-2}	38.7	6.4	16.4	29.8	27.1	10.7
4	2	20	1	37.9	8.2	17.2	32.2	25.6	11.9
4	4	5	10^{-2}	36.5	4.0	15.4	26.5	24.8	10.2
4	2	5	1	36.3	6.3	15.9	29.2	24.1	10.8
3	2	20	10^{-2}	35.2	4.6	16.6	27.5	23.5	11.2
4	4	20	1	34.4	5.9	15.9	28.3	23.4	11.5
3	4	20	10^{-4}	31.6	2.3	15.6	23.9	22.4	10.7

or momentum. From the table, we can observe that the evaluation score increase at first, but then start to decline after 10 training epoch which indicates that the model start to over-fit on training data. Width 5 Beam search can slightly improve the overall performance but the difference is not significant. A large beam width of 20 perform worse than greedy search in evaluation, which is out of our expectation. The degeneration of performance using large beam width is potentially caused by over-fitting of the model on training set. Table 5 evaluates on LSTM-model trained with different data pre-processing and sampling techniques. Compared with the result in Table 4, we noticed that this data processing and sampling technique doesn’t make significant difference compared to fixed-length

Table 4: Performance of CNN-LSTM model COCO “Karpathy” validation split with fixed-length caption batches. Notations are the same as Table 1

Training Epoch	Beam Width	B@1	B@4	M	R	C	S
5	1	67.2	24.4	22.7	49.6	78.9	15.3
5	5	67.9	27.2	23.1	50.2	82.6	15.7
7	1	67.7	25.2	22.8	49.7	80.6	15.5
7	5	68.2	28.0	23.3	50.2	84.3	16.0
7	20	66.8	27	22.9	49.2	81.5	15.7
8	1	68.1	25.3	23.0	50.3	35.8	25.3
8	5	67.6	27.4	23.3	50.1	83.7	16.0
9	1	67.9	25.3	23.3	50.0	81.2	16.0
9	5	68.2	27.7	23.6	50.5	84.1	16.5
11	1	67.4	24.8	22.9	49.6	80.3	15.8
11	5	67.9	27.2	23.3	50.1	83.7	16.1
11	20	66.4	26.4	22.9	49	80.8	15.7

Table 5: Performance of CNN-LSTM model COCO “Karpathy” validation split with variable-length caption batches. Notations are the same as Table 1

Training Epoch	Beam Width	B@1	B@4	M	R	C	S
5	5	64.6	24.8	23.7	49.5	79.8	16.3
7	5	66.0	25.2	23.1	49.2	79.3	15.8
9	5	65.7	25.4	23.4	49.4	80.7	16.0

captions. We propose this method to evaluate the influence of <pad> tokens on the performance model. With two results, we can make sure the existence of <pad> token won’t interfere the training process and performance of the model.

For the whole CNN-LSTM experiment, we reached reasonably good performance on all the evaluation benchmark. Its performance can be further tuned using different training hyper-parameters by apply learning rate decay and momentum.

5.4 Model Comparison

Figure 5 compares the generated captions from the best-tuned models to one of the reference captions from the dataset. The Markov-based model used is trained with 4-grams and a 2×2 grid. Sentence generation uses a beam width of 20 and decay of 10^{-2} . The CNN-LSTM model used is trained for 9 epochs with method 1. Sentence generation uses a beam width of 5. For the first image, the Markov-based model missed the “cow” while the CNN-LSTM model misunderstood it as elephant. For the second image, both models described it accurately, although the Markov-based model started to ramble. For the third image, the CNN-LSTM model correctly captioned it while the Markov-based model missed the “girl” in the image and instead hallucinated a “steering wheel”. Overall speaking, the Markov-based model frequently missed or hallucinated objects while the CNN-LSTM model more accurately depicted each image.

6 Conclusion

In this paper, we tackled the challenge of image captioning with 2 approaches: a classic Markov-based model and a modern CNN-LSTM model, and compared their performances on COCO 2014 dataset side-by-side. The Markov-based model, being a probabilistic method, made strong assumptions on the input sequences and features such that it’s trainable. Some feature engineering are also needed to enrich its understanding of the images. The CNN-LSTM model does not make any strong assumptions on the input and merely approximates a highly-complex function that minimizes our target loss given any inputs. No feature engineering is required as the model can automatically extract features from images using CNN and context from words using embedding. Our experiments showed a clear advantage of the CNN-LSTM model in image captioning, though the Markov-based model is still able to generate a somewhat relevant sentence. In the future, we plan to study the attention mechanism, apply it on the CNN-LSTM model and observe the hidden states to understand how it improves the model performance. We also plan to study the better way to extract and encode features, for example using multivariate Gaussian, for the Markov-based model such that it can perform better.




	Ref: a young boy barefoot holding an umbrella touching the horn of a cow Markov: a group of people are standing in the grass near trees CNN-LSTM: a couple of elephants standing next to each other
	Ref: doorway view of a kitchen with a sink, stove, refrigerator and pantry Markov: modern kitchen with stainless steel appliances and granite counter tops and stainless steel refrigerator microwave toaster CNN-LSTM: a kitchen with a sink , stove , and cabinets
	Ref: a young girl is holding a small cat Markov: a cat laying on top of the steering wheel CNN-LSTM: a woman holding a cat in her arms

Figure 5: Model Comparison

References

- [1] Ahmet Aker and Rob Gaizauskas. 2009. Summary Generation for Toponym-referenced Images using Object Type Language Models. *Proceedings of the International Conference RANLP-2009* (01 2009), 6–11.
- [2] S. Amirian, Khaled Rasheed, T. Taha, and H. Arabnia. 2019. A Short Review on Image Caption Generation with Deep Learning.
- [3] Yoshua Bengio and Paolo Frasconi. 1994. An Input Output HMM Architecture. In *Proceedings of the 7th International Conference on Neural Information Processing Systems (NIPS'94)*. MIT Press, Cambridge, MA, USA, 427–434.
- [4] Y. Bengio and P. Frasconi. 1996. Input-output HMMs for sequence processing. *IEEE Transactions on Neural Networks* 7, 5 (1996), 1231–1249. <https://doi.org/10.1109/72.536317>
- [5] Xinlei Chen and C. Zitnick. 2015. Mind’s eye: A recurrent visual representation for image caption generation. 2422–2431. <https://doi.org/10.1109/CVPR.2015.7298856>
- [6] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: generating sentences from images. *Lecture Notes in Computer Science* 21, 10 (2010), 15–29.
- [7] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- [8] Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. 2019. Attention on Attention for Image Captioning. In *International Conference on Computer Vision*.
- [9] A. Karpathy and L. Fei-Fei. 2017. Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 4 (2017), 664–676. <https://doi.org/10.1109/TPAMI.2016.2598339>
- [10] Ryan Kiros, Ruslan Salakhutdinov, and Richard Zemel. 2014. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. *31st International Conference on Machine Learning, ICML 2014* 3 (11 2014).
- [11] Philipp Koehn. 2009. *Statistical Machine Translation*. <https://doi.org/10.1017/CB09780511815829>
- [12] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *CVPR 2011*. 1601–1608. <https://doi.org/10.1109/CVPR.2011.5995466>
- [13] Siming Li, Girish Kulkarni, Tamara Berg, Alexander Berg, and Choi Yejin. 2011. Composing Simple Image Descriptions using Web-scale N-grams. *CoNLL 2011 - Fifteenth Conference on Computational Natural Language Learning, Proceedings of the Conference* (01 2011).
- [14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Zitnick. 2014. Microsoft COCO: Common Objects in Context. (05 2014).
- [15] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2015. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3156–3164. <https://doi.org/10.1109/CVPR.2015.7298935>
- [16] Haoran Wang, Yue Zhang, and Xiaosheng Yu. 2020. An Overview of Image Caption Generation Methods. *Computational Intelligence and Neuroscience* 2020 (01 2020), 1–13. <https://doi.org/10.1155/2020/3062706>
- [17] Qi Wu, Chunhua Shen, Lingqiao Liu, Anthony Dick, and Anton Hengel. 2016. What Value Do Explicit High Level Concepts Have in Vision to Language Problems? 203–212. <https://doi.org/10.1109/CVPR.2016.29>
- [18] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Y. Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. (02 2015).
- [19] Yan Li and Heung-Yeung Shum. 2006. Learning dynamic audio-visual mapping with input-output Hidden Markov models. *IEEE Transactions on Multimedia* 8, 3 (June 2006), 542–549. <https://doi.org/10.1109/TMM.2006.870732>

- [20] Yezhou Yang, Ching Teo, Hal III, and Yiannis Aloimonos. 2011. Corpus-Guided Sentence Generation of Natural Images. *EMNLP 2011 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 444–454.
- [21] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. 2016. Image Captioning with Semantic Attention. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4651–4659. <https://doi.org/10.1109/CVPR.2016.503>