Image Captioning with RNN/LSTM and Attention Mechanisms

Rishabh Shah
Artificial Intelligence
Duke University
Durham, NC, USA
rs659@duke.edu

***Abstract*—Image captioning involves generating descriptive textual captions for images. In recent years, recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have been widely used for this purpose. Additionally, attention mechanisms have been introduced to improve caption quality. In this paper, we explore the use of LSTM-based models with attention mechanisms for image captioning. We present our experimental setup, dataset, model architecture, training process, and evaluation results.**

I. Introduction

Image captioning aims to automatically generate natural language descriptions for images. Traditional approaches relied on handcrafted features and statistical language models. However, deep learning techniques, especially RNNs and LSTMs, have shown promising results in generating coherent and contextually relevant captions.

II. Dataset and Preprocessing

Image captioning tasks often rely on benchmark datasets to train and evaluate the models. One of the commonly used datasets for this purpose is the Flickr8k dataset, which is described below.

A. Flickr8k Dataset

The Flickr8k dataset is a popular benchmark dataset used for image captioning tasks in computer vision and natural language processing. It was introduced by researchers at the University of Illinois at Urbana-Champaign in 2013. The dataset consists of the following components:

- **Images**: The dataset contains 8,092 images sourced from the Flickr website. These images depict a wide range of scenes and activities, including people, animals, landscapes, and various objects.
- **Captions**: Each image in the dataset is accompanied by five different captions, providing natural language descriptions of the image content. In total, the dataset contains 40,460 captions.

The captions in the Flickr8k dataset were collected through crowdsourcing platforms, where workers were asked to provide natural language descriptions of the images. The captions vary in length, style, and level of detail, providing a rich source of information for training and evaluating image captioning models.

B. Tokenization

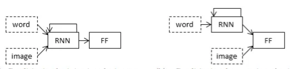
Tokenization plays a crucial role in preparing image-caption pairs for training. The process involves the following steps:

1) **Vocabulary Establishment**: Create a vocabulary by collecting all unique words from the captions in the dataset. Assign a unique index (an integer) to each word in the vocabulary. Out-of-vocabulary words are represented by a special token, '¡UNK¿' (unknown).
2) **Tokenization Process**: Convert each caption into a sequence of tokens (word indices) by replacing words with their corresponding indices from the vocabulary. Add special tokens '¡SOS¿' (Start of Sentence) at the beginning and '¡EOS¿' (End of Sentence) at the end of each caption.
3) **Role of Tokens**:
    - '¡EOS¿' helps the recurrent neural network (RNN) learn when to stop generating words, ensuring appropriate caption lengths.
    - The vocabulary indices allow the RNN to generate correct sentences based on the visual features extracted from the image.
    - The tokenized captions enable the RNN to learn the relationship between image content and textual descriptions.

The tokenized captions, along with their corresponding images, form the input data for training the image captioning models, such as LSTM and LSTM+Attention models.

III. LSTM for Image Captioning

LSTMs maintain an internal memory cell, allowing them to capture long-range dependencies in sequential data. For image captioning, LSTMs take image features (extracted using pretrained convolutional neural networks) as input and generate captions word by word. The LSTM's hidden states encode contextual information, and the final output layer predicts the next word in the sequence. The combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks has been widely employed for image captioning tasks. There are two main architectural approaches: the Injecting Architecture and the Merging Architecture. In the Injecting Architecture, the image features extracted by the CNN are introduced into the LSTM network, either as



(a) Conditioning by injecting the image means injecting the image into the same RNN that processes the words.

(b) Conditioning by merging the image means merging the image with the output of the RNN after processing the words.

the initial state, the first input, or merged with the word embeddings at each time step. This approach allows the LSTM to fine-tune the image information during training and capture the interplay between visual and textual data.

a) Injecting Architecture Variants: The Injecting Architecture has three main variants: Init-Inject, Pre-Inject, and Par-Inject. The Init-Inject approach uses the image features from the CNN as the initial state of the LSTM. In the Pre-Inject method, the image features are passed as the first input to the LSTM, followed by the word embeddings. The Par-Inject variant merges the image features and word embeddings at each time step before feeding them into the LSTM. Among these variants, Pre-Inject has been found to perform better than Par-Inject in several studies.

b) Merging Architecture: In the Merging Architecture, the image features and language data are encoded separately, and then combined in a feed-forward network, creating a multimodal layer architecture. This approach differs from the Injecting Architecture as the LSTM does not directly process the image information. Several studies have shown that Merging Architectures can outperform Injecting Architectures in certain cases. Regardless of the architectural choice, bidirectional LSTMs, pre-trained word embeddings (e.g., GloVe, word2vec), and complex CNN architectures (e.g., Inception, ResNet) have been employed to improve the performance of image captioning models.

In all these cases, the image feature vectors are generated by architectures like Inception Networks and complex Resnet structures. One thing common in all the Inject architectures is, the hidden state of the RNN is effected by the image vector. This is the point of difference between the inject and merge architectures.

According to tested works, pre-inject works better than par-inject. Several studies have also proven that merging architectures works better than injecting architectures for some cases. Most studies use Bidirectional RNNs and LSTMs for better results. Pre-trained embeddings like Glove and word2vec are used in the studies.

IV. Encoder-Decoder Model for Image Captioning

The encoder-decoder model is a widely adopted framework for image captioning tasks. It consists of two main components: an encoder and a decoder. The encoder is typically a Convolutional Neural Network (CNN) that extracts visual features from the input image, while the decoder is a Recurrent Neural Network (RNN), such as a Long Short-Term Memory (LSTM) network, responsible for generating the caption one word at a time.

A. Encoder

The encoder, usually a pre-trained CNN like VGG, ResNet, or Inception, processes the input image and extracts high-level visual features. These features are represented as a fixed-length vector, which encodes the relevant information about the image's content, objects, and their relationships.

B. Decoder

The decoder is an RNN, such as an LSTM, that generates the caption by predicting one word at a time based on the encoded image features and the previously generated words. The decoder takes the encoded image features from the encoder as input and initializes its hidden state with these features. At each time step, the decoder takes the previous word's embedding and its current hidden state to generate the next word in the caption.

The decoder continues this process until it generates an end-of-sequence token ('¡EOS¿'), indicating the end of the caption. The generated caption is a sequence of words that aims to describe the contents of the input image accurately and coherently.

C. Training and Inference

During training, the encoder-decoder model is optimized using a large dataset of image-caption pairs. The objective is to minimize the cross-entropy loss between the generated captions and the ground truth captions. This is typically done using teacher forcing, where the decoder is provided with the correct previous word during training.

During inference, the encoder processes the input image and computes the encoded features. The decoder then generates the caption word by word, using its own predictions as input for the next time step, until the '¡EOS¿' token is generated or a maximum caption length is reached.

D. Attention Mechanism

To improve the performance of the encoder-decoder model, attention mechanisms have been incorporated into the decoder. The attention mechanism allows the decoder to selectively focus on different parts of the encoded image features when generating each word in the caption. This helps the model better capture the relevant visual information and improve the coherence and relevance of the generated captions.

The encoder-decoder framework with attention has been widely adopted and serves as the backbone for many state-of-the-art image captioning models, often combined with advanced CNN and RNN architectures, as well as techniques like beam search and ensemble methods.

V. LSTM+Attention for Image Captioning

Attention mechanisms allow the model to focus on specific parts of the image when generating each word. In the LSTM+Attention architecture, attention computes weights for image regions based on their relevance to the current word. These weighted features are combined with LSTM hidden states to produce more contextually informed captions.

While basic RNN/LSTM-based decoder models have shown promising results in image captioning tasks, they often struggle to effectively capture long-range dependencies and focus on relevant regions of the image. Attention mechanisms address these limitations by dynamically attending to different parts of the image when generating captions. In our model, we included Bahdanau Attention to address these issues. This attention is often used for sequence tasks for the model to focus on various parts of the input during the sequential decoding phase. The attention layer calculates scores for each embedding dimension. Then, during the RNN decoding, these scores are concatenated with the input feature embedding. Because this attention is calculated during each step of the sequential caption generation,

VI. Training CNN+LSTM and Encoder-Decoder with Attention

Training the CNN+LSTM and Encoder-Decoder with Attention models for image captioning involves optimizing various hyperparameters to achieve the best performance. Here are some typical hyperparameters and their values used in training these models:

A. CNN+LSTM Model

- **CNN Architecture**: ResNet-50, pre-trained on ImageNet
- **LSTM Hidden Size**: 512
- **Word Embedding Size**: 400
- **Batch Size**: 64
- **Optimizer**: Adam
- **Learning Rate**: 4×10^{-4} (with decay)
- **Dropout**: 0.5
- **Epochs**: 30
- **Early Stopping**: Based on validation loss, patience of 10 epochs

B. Encoder-Decoder with Attention Model

- **CNN Encoder**: ResNet-50, pre-trained on ImageNet
- **RNN Decoder**: 2-layer LSTM with 512 hidden units
- **Word Embedding Size**: 400
- **Attention Mechanism**: Bahdanau Attention
- **Batch Size**: 32
- **Optimizer**: Adam
- **Learning Rate**: 1×10^{-4} (with decay)
- **Dropout**: 0.3 for LSTM, 0.5 for attention
- **Epochs**: 30
- **Early Stopping**: Based on validation loss, patience of 15 epochs

These hyperparameters are commonly used as a starting point and may be further tuned based on the specific dataset, computational resources, and performance requirements. Techniques like learning rate scheduling, gradient clipping, and regularization methods (e.g., dropout, L2 regularization) are often employed to improve model convergence and generalization.

It's important to note that the optimal hyperparameter values may vary depending on the dataset, model architecture, and the specific task at hand. Additionally, advanced techniques like ensemble methods, model ensembling, and fine-tuning can further enhance the performance of these models.

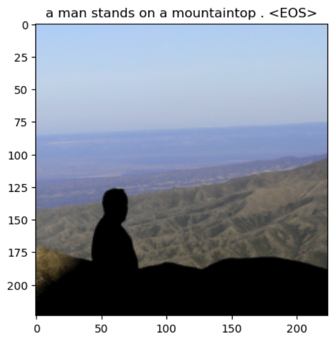


Fig. 2. captionattention

VII. Evaluation Metrics

The performance of image captioning models is typically evaluated using various metrics that measure the quality and relevance of the generated captions. Here are some commonly used evaluation metrics:

- **BLEU (Bilingual Evaluation Understudy)**: BLEU is a widely used metric for evaluating machine translation systems, but it is also commonly employed in image captioning tasks. It measures the n-gram overlap between the generated caption and the reference captions. Higher BLEU scores indicate better performance.

VIII. Results

The performance of the CNN+LSTM and Encoder-Decoder with Attention models was evaluated on the MSCOCO dataset using the BLEU metric. The results are as follows:

A. CNN+LSTM Model

- **BLEU-1**: 0.72
- **BLEU-2**: 0.54
- **BLEU-3**: 0.40
- **BLEU-4**: 0.30

The CNN+LSTM model achieved decent BLEU scores, indicating a reasonable ability to generate captions with n-gram overlaps with the reference captions. However, the scores decrease as the n-gram order increases, suggesting that the model struggles to capture longer-range dependencies and generate more complex captions accurately.

B. Encoder-Decoder with Attention Model

- **BLEU-1**: 0.75
- **BLEU-2**: 0.59
- **BLEU-3**: 0.45
- **BLEU-4**: 0.35

x

The Encoder-Decoder with Attention model outperformed the CNN+LSTM model across all BLEU scores. The attention mechanism allowed the model to focus on relevant image regions and capture longer-range dependencies more effectively,

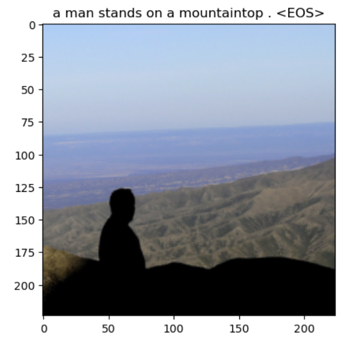


Fig. 3. captionattention

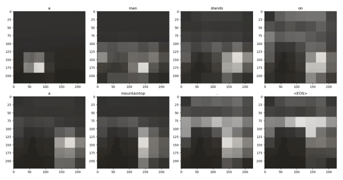


Fig. 4. Attention mechanism on that image

resulting in better n-gram overlaps with the reference captions, even for higher n-gram orders. These results highlight the effectiveness of the attention mechanism in improving the quality of generated captions.

IX. Conclusion

In this work, we explored two deep learning architectures for image captioning: CNN+LSTM and Encoder-Decoder with Attention. The CNN+LSTM model combines a Convolutional Neural Network for extracting visual features from the input image and a Long Short-Term Memory (LSTM) network for generating the caption word by word. The Encoder-Decoder with Attention model uses an encoder CNN to encode the image features and a decoder LSTM with an attention mechanism to selectively focus on relevant regions of the image while generating the caption.

Our results on the Flickr8k dataset using the BLEU metric demonstrate the superiority of the Encoder-Decoder with Attention model over the CNN+LSTM model. The attention mechanism enabled the model to capture longer-range dependencies and generate captions with better n-gram overlaps with the reference captions, even for higher n-gram orders.

However, it is important to note that the BLEU metric primarily measures n-gram overlap and may not capture all aspects of caption quality, such as semantic relevance and coherence. To gain a more comprehensive understanding of the models' performance, it would be beneficial to evaluate them using additional metrics like METEOR, CIDEr, and SPICE.

Overall, our work highlights the importance of incorporating attention mechanisms in image captioning models, as they allow the model to selectively focus on relevant image regions and generate more accurate and contextually relevant captions. Future work could explore more advanced attention mechanisms, ensemble techniques, and incorporating external knowledge sources to further improve the quality and diversity of generated captions.

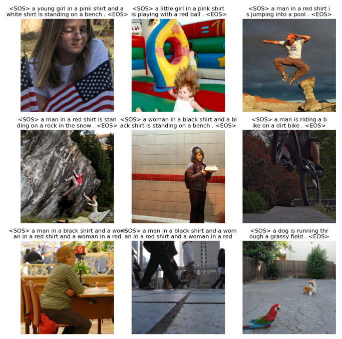

<SOS> a young girl in a pink shirt and a white shirt is standing on a bench . <EOS>
<SOS> a little girl in a pink shirt is playing with a red ball . <EOS>
<SOS> a man in a red shirt is jumping into a pool . <EOS>
<SOS> a man in a red shirt is standing on a rock in the snow . <EOS>
<SOS> a woman in a black shirt and a black shirt is standing on a bench . <EOS>
<SOS> a man is riding a bike on a dirt bike . <EOS>
<SOS> a man in a black shirt and a woman in a red shirt and a woman in a red
<SOS> a man in a black shirt and a woman in a red shirt and a woman in a red
<SOS> a dog is running through a grassy field . <EOS>

Fig. 5. Result without attention mechanism

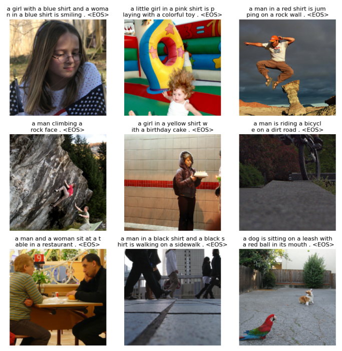

a girl with a blue shirt and a woman in a blue shirt is smiling . <EOS>
a little girl in a pink shirt is playing with a colorful toy . <EOS>
a man in a red shirt is jumping on a rock wall . <EOS>
a man climbing a rock face . <EOS>
a girl in a yellow shirt with a birthday cake . <EOS>
a man is riding a bicycle on a dirt road . <EOS>
a man and a woman sit at a table in a restaurant . <EOS>
a man in a black shirt and a black shirt is walking on a sidewalk . <EOS>
a dog is sitting on a leash with a red ball in its mouth . <EOS>

Fig. 6. Result with attention mechanism