Image captioning with region attention : fine-tuning with Reinforcement Learning

EL BAZ Adrian
ENS Paris-Saclay
Master MVA
January, the 21st 2020

adrian.el_baz@ens-paris-saclay.fr

Abstract

It has been shown that image-captioning tasks performs well considering an encoder-decoder paradigm in previous works. In this paper, we replicate results from one popular approach which is the Bottom-Up Top Down attention mechanism [1]. We obtain a similar result than from the original paper : 106.2 against 113.5 with CIDEr score.

I.Introduction

A lot of work has been done using attention mechanism in Computer Vision since its birth in the machine translation context. The image captioning problem is one of the most challenging task in this field, and most of the recent papers focused on a specific deep network architecture : the encoder-decoder paradigm. The underlying idea of this approach is to find a way to associate visual context with language context through attention. In other words, for an input image, we need to capture its global context via a vector representation and allow to generate a caption which is directly linked to it. The attention mechanism takes its inspiration from our own visual system : focusing on specific parts of the image should be sufficient to describe relevantly a scene as we do. Following this idea, it is intuitive that these parts could be salient objects present in the image : this is one the key points from [1] that is different from previous approaches.

The overall task is challenging because we want to create a robust caption generator that could adapt in unseen images. Hence, this model must capture and bind visual concepts to language, and not overfit to the training dataset. This is the crucial part of the process.

Different forms of visual features extractions had been considered and will be discussed in Section II.

The whole process is done with the complete end-to-end training of a deep network that is composed of several modules connected to each over. Each module has a specific task to achieve (e.g. extraction of features, language LSTM). This paradigm is described in Figure 1.

In this project, we replicate the impact of using bottom-up and top-down attention on a captioning task using the MSCOCO dataset. We also add some hints for further experiments and potential future work.

II.Related work

A wide variety of model has been tested for the image captioning task. Most of these approaches can be solely considered as top-down models since they do use attention on specific parts of the image but they are not using any object detection method to extract visual features as it is done in [1]. Instead, as described by Anderson et al. in [2], they directly extract visual features from one the convolutional layer of the CNN (last for fine-level details or, for example, middle one for coarser details) that is pre-trained on ImageNet dataset. One thing that is worth noting is that the MSCOCO and ImageNet datasets have overlapping objects classes that make transfer learning relevant in this case.

The choice of the aforementioned convolutional layer is actually a hard task as mentioned by Long Chen et al. in [5]. Indeed, extracting visual features from two different convolutional layers could lead to different image's representations with either coarser or finer-level details. Also the arbitrary positioning of the regions with respect to image content may lead to difficulties binding visual concepts associated with the same object. This last insight further motivates the emphasis we put in paper [1] as we first detect the object and then feeding it in the CNN feature extractors.

Long Chen et al.[5] showed promising results by circumventing the choice of the convolutional layer we have to choose, considering all convolutional layers and perform channel-wise in addition to spatial-wise attention in each of them.

Finally, Rennie et al. [3] showed real improvements of existing encoder-decoder approach by directly optimize the network with a sequential metric (e.g. CIDEr) using a Reinforcement Learning framework, instead of the classical cross-entropy loss optimization. The latter method is problematic at test time and needs particular

attention. Indeed, we can't rely solely on our network predictions of next word at the early stages of training because otherwise it could lead to dramatic error propagation leading to mediocre results at test time. Therefore previous work progressively allows the network to train on its own prediction but still, training on ground truth captions is not optimal. Overall, we also use beam search for the decoding part.

III.Approach

In this paper, we mostly focus on the Bottom's Up and Top-Down attention approach. The difference between the standard encoder-decoder approach is shown in Figure 2. The two main differences are the following :

• *Features extraction*: An object detection method is first ran over the image to detect salient objects. It is the output of this Faster R-CNN that is fed to the CNN to obtain a vector representation of the image.

• *Attention mechanism*: The attention mechanism is modeled as an LSTM. It distributes weights to each vector from the set of features vectors and allows to consider visual context in addition to language context to generate the next set of weights (to generate the next word).

The second LSTM is meant to generate caption words, it is standard to consider such an architecture to generate the caption. Then the outputs of the last LSTM is fed into a last layer that compute the distribution of words given previous generated words.

For self-sufficiency of this project, we report the equations from [1] using the same notation :

The output of a LSTM cell is described by the following equation. If we need to specify from which LSTM the output comes from (or input), we use superscript 1 for the attention LSTM and 2 for language LSTM.

$$\boldsymbol{h}_t = \mathbf{LSTM}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \tag{1}$$

First of all all, the input vector of representation is described as follows, where v is the set of extracted feature vectors, We is the word embedding for the specific vocabulary and capital Pi is the one-hot encoding of the input word at time-step t :

$$\boldsymbol{x}_t^1 = [\boldsymbol{h}_{t-1}^2, \bar{\boldsymbol{v}}, W_e \Pi_t] \tag{2}$$

Then we describe how we get the vectors weights computed in the first LSTM :

$$a_{i,t} = \boldsymbol{w}_a^T \tanh\left(W_{va}\boldsymbol{v}_i + W_{ha}\boldsymbol{h}_t^1\right) \tag{3}$$

$$\boldsymbol{\alpha}_t = \mathrm{softmax}\left(\boldsymbol{a}_t\right) \tag{4}$$

So at the end of the first LSTM pass, we get a weighted average of the feature vectors representing the attention on the specific parts of the image :

$$\hat{\boldsymbol{v}}_t = \sum_{i=1}^{K} \alpha_{i,t}\boldsymbol{v}_i \tag{5}$$

The distribution of words given previously generated ones is computed as follows :

Hence, we model the distribution over complete output sequences as follows :

$$p(y_{1:T}) = \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}) \tag{8}$$

We replicate results using the cross-entropy criterion as a loss function. Notice that we use ground truth captions and not current model predictions :

$$L_{XE}(\theta) = -\sum_{t=1}^{T} \log(p_\theta(y_t^* \mid y_{1:t-1}^*)) \tag{9}$$

However, the other interesting approach could have been to optimize directly with respect to sequential metrics using the reinforcement learning framework. This framework allows to sample from the distribution of words of the current network :

$$L_R(\theta) = -\mathbf{E}_{y_{1:T}\sim p_\theta}[r(y_{1:T})] \tag{10}$$

From [3], we get this gradient approximation where the first reward is from the sampled caption and the second one is obtained by greedily decoding the current model.

$$\nabla_\theta L_R(\theta) \approx -(r(y_{1:T}^s) - r(\hat{y}_{1:T}))\nabla_\theta \log p_\theta(y_{1:T}^s) \tag{11}$$

Table 1: Comparison of results between our implementation and paper [1] results. First and second row are from cross-entropy optimization and the third row is under CIDEr optimization.

Results for different metrics							
Metric / Implementation	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	CIDEr
Our results	74.250	57.23	42.34	32.43	53.92	24.342	106.234
Paper results	77.2	-	-	36.2	56.4	27.7	113.5
Paper results with CIDEr opti.	79.8	-	-	36.3	56.9	27.7	120.1

IV.Evaluation

We mainly use the Poojahira's Github implementation [6], to replicate results from [1]. We replicate results for the MSCOCO caption dataset. For validation we use the famous Karpathy splits which contains 113,287 training images with five captions each, and 5000 images respectively for validation and testing. We used already extracted bottom's up features. The main change we made from this implementation is the ability to train on multiple GPU's. We successfully changed the code to allow training on 4 Tesla K80. Indeed, changes were necessary as each of the 4 parts of the initial batch didn't have knowledge of the maximum caption length in the original batch. Therefore it was not possible at first to gather safely the results from all device. The time gained compared to the use of a single GPU was not as big as we thought (only 2.5 times faster than a single GPU). We suspect our implementation to have a bottleneck in computations distribution due to the master worker (the first GPU) having too much tasks to complete before continuing with the next batch. We should probably further improve our implementation by considering to send some tasks to the CPU, for example the loss computations.

To evaluate caption quality, we consider the standard metrics namely SPICE, CIDEr, METEOR, ROUGE-L and BLEU. The results are summed up in Table 1.

Qualitative results :

In the Figure 3 we showcase an example of an image for which the model struggles to output a meaningful caption. This illustrates the lack of visual concepts binding to language concept of our model. An interesting analysis would have been to generate the output of the same image with the model trained on CIDEr metric.

V.Conclusion

In this project, we successfully replicate results from [1] with the cross-entropy loss optimization. We also witness some of the limits of this model using the trained network. Further qualitative analysis should be conducted using the SCST approach to see if this fine-tuning procedure helped to improve some of this lack of scene understanding we showcased.

Finally one last experiment should be to investigate the choice of the convolutional layer considered to extract the set of vectors representing the original image. One could try different layers to see how results differ, if they differ.

References

1. Anderson et al. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. CVPR 2018
2. Xu et al. Show, Attend and Tell : Neural Image Caption Generation with Visual Attention. PMLR 2015.
3. Rennie et al. Self-Critical Sequence Training for Image Captioning. CVPR 2017.
4. Lu et al. Neural Baby Talk. CVPR 2018.
5. Long Chen et al. SCA-CNN: Spatial and Channel-wise Attention in Convolutional Networks for Image Captioning. 2017
6. Github's implementation of [1] : https://github.com/poojahira/image-captioning-bottom-up-top-down

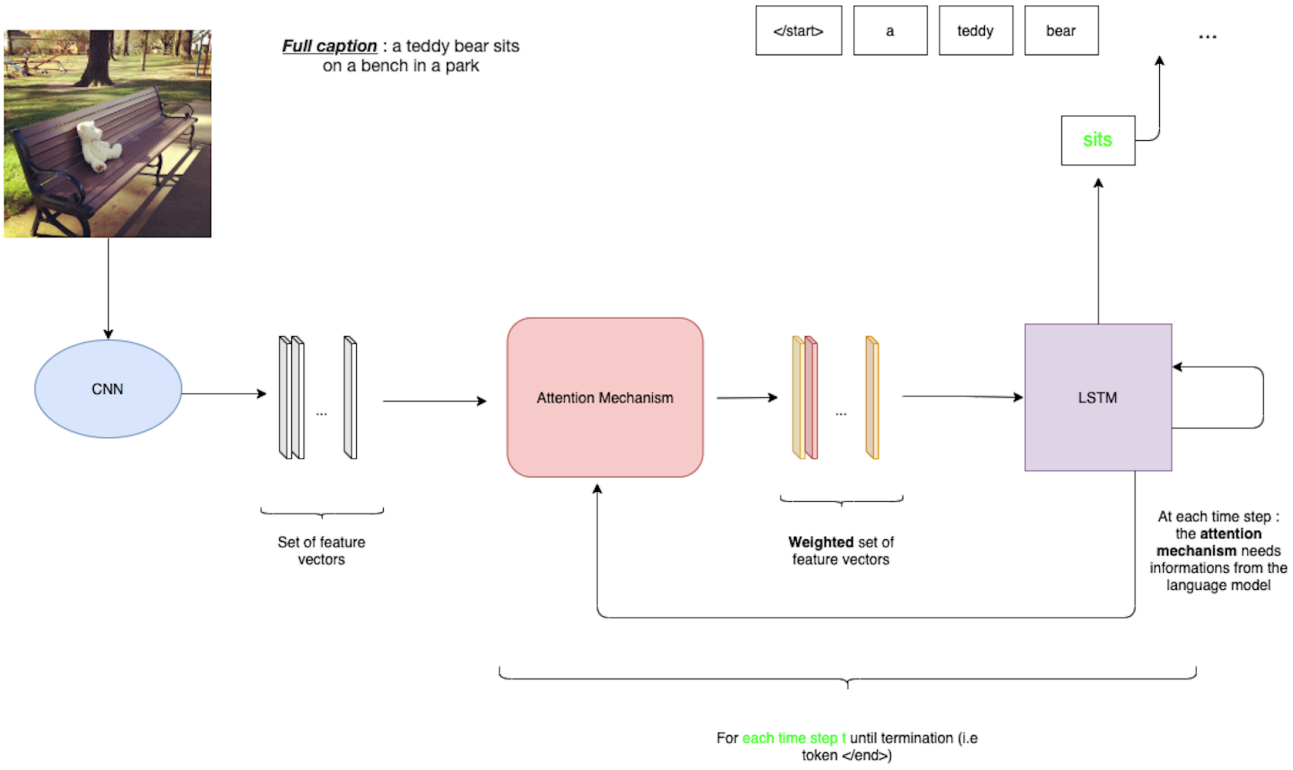

Figure 1 : General Encoder-Decoder paradigm approach.

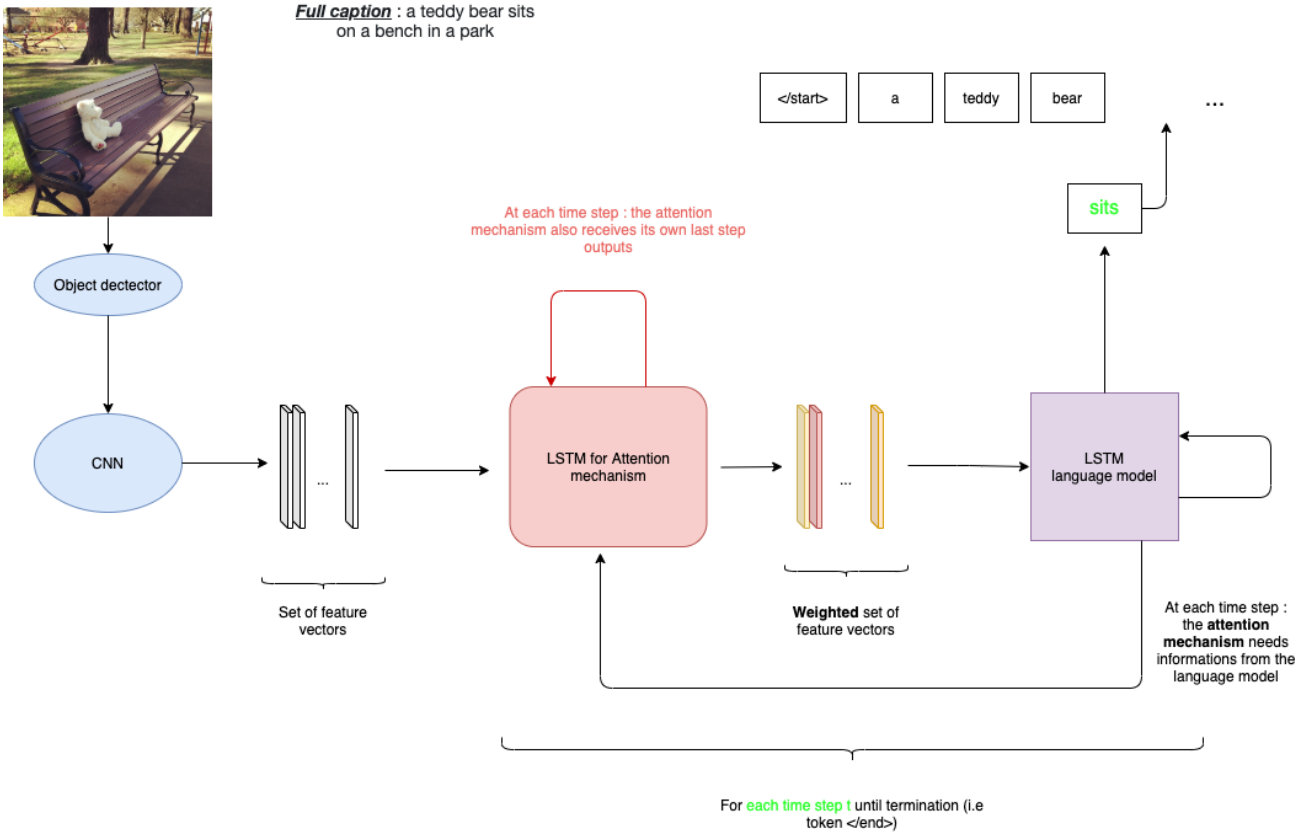

Figure 2: Bottom's up Top-Down attention mechanism. Notice the difference with the general encoder-decoder approach.

Figure 3: Trained network fails to generate a relevant caption. **Caption**: a group of men riding on the back of horses