Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

1 Citation

Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International conference on machine learning. 2015.

`https://arxiv.org/pdf/1502.03044.pdf`

2 Abstract

We combine a CNN, LSTM, and attention mechanism to do image captioning on the Flickr8k, Flickr30k, and MS COCO datasets - we set state of the art.

3 Introduction

The human visual system can automatically focus on parts of an image. This attention mechanism is important for scene understanding, so we build it (with both a soft and hard variant) into our neural network models.

4 Related Work

People have used encoder/decoder Recurrent Neural Networks (RNN) for translation. People have even used Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM, a kind of RNN) networks together for image captioning. Others learn a joint image/word embedding and treat captioning as a ranking problem. Before neural networks, people would find a similar captioned image and tweak the caption for the new image.

5 Image Caption Generation with Attention Mechanism

Our goal is to predict a caption, which is a sequence of one-hot encoded (vocabulary size is K) words:

$$y = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_C]$$

We use a CNN to extract L descriptors (each from different area of image) of dimension D from the image:

$$a = \mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_L$$

We apply out attention mechanism to get context vector $\hat{\mathbf{z}}_t$ and apply the following LSTM:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} \mathbf{E}\mathbf{y}_{t-1} \\ \mathbf{h}_{t-1} \\ \hat{\mathbf{z}}_t \end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

Where T is a learned affine transform, $\mathbf{E}$ is a word embedding matrix, σ is sigmoid, $\odot$ is element-wise product, and $\mathbf{i}$, $\mathbf{f}$, $\mathbf{g}$, $\mathbf{o}$, $\mathbf{c}$, and $\mathbf{h}$ are the input, forget, input modulation, output, memory, and hidden state of the LSTM, respectively.

Now let's discuss the attention mechanism that creates $\hat{\mathbf{z}}_t$ from a. We create a weight α_i for each $\mathbf{a}_i$. To do this, we do:

$$e_{t,i} = f_{att}(\mathbf{a}_i, \mathbf{h}_{t-1})$$
$$\alpha_{t,i} = \text{Softmax}(e_{t,i})$$
$$\hat{\mathbf{z}} = \phi(\{\mathbf{a}_i\}, \{\alpha_i\})$$

where f_{att} is a multi-layer perceptron and ϕ is different depending on whether we want soft or hard attention.

The initial LSTM state is given by:

$$\mathbf{c}_0 = f_{init,c}(\frac{1}{L}\sum_{i=1}^{L}\mathbf{a}_i)$$

$$\mathbf{h}_0 = f_{init,h}(\frac{1}{L}\sum_{i=1}^{L}\mathbf{a}_i)$$

where $f_{init,c}$ and $f_{init,h}$ are multi-layer perceptrons.

Our final prediction output is:

$$p(\mathbf{y}_t|a, \mathbf{y}_{t-1}) \propto \exp(\mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h\mathbf{h}_t + \mathbf{L}_z\hat{\mathbf{z}}_t))$$

where the matrices above are learned.

6 Learning Stochastic "Hard" vs. Determinstic "Soft" Attention

Let's start with Hard attention.

We can view $\hat{\mathbf{z}}_t$ as a Multinoulli random variable.

Let $\mathbf{s}_t$ be a one-hot vector indicating which $\mathbf{a}_i$ the attention mechanism decides to focus on. We get:

$$p(s_{t,i}|\mathbf{s}_{j<t}, a) = \alpha_{t,i}$$
$$\hat{\mathbf{z}}_t = \sum_i s_{t,i}\mathbf{a}_i$$

We define the following loss function:

$$L_s = \sum_s p(\mathbf{s}|\mathbf{a}) \log p(\mathbf{y}|\mathbf{s}, \mathbf{a}) \leq \log(\sum_s p(\mathbf{s}|\mathbf{a}) p(\mathbf{y}|\mathbf{s}, \mathbf{a})) = \log p(\mathbf{y}|\mathbf{a})$$

We can then compute the gradient with respect to the model weights (collectively denoted as W).

$$\frac{\partial L_s}{\partial W} = \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{a}) (\frac{\partial p(\mathbf{y}|\mathbf{s}, \mathbf{a})}{\partial W} + \log p(\mathbf{y}|\mathbf{s}, \mathbf{a}) \frac{\partial p(\mathbf{s}|\mathbf{a})}{\partial W})$$

We can approximate the gradient with Monte Carlo sampling where $\mathbf{s}_t$ is drawn from $\text{Multinoulli}_L(\{\alpha_i\})$.

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N} \sum_{n=1}^{N} \frac{\partial p(\mathbf{y}|\tilde{\mathbf{s}}^n, \mathbf{a})}{\partial W} + \log p(\mathbf{y}|\tilde{\mathbf{s}}^n, \mathbf{a}) \frac{\partial p(\tilde{\mathbf{s}}^n|\mathbf{a})}{\partial W}$$

Now we don't need to sum over all $\mathbf{s}$. We compute this with minibatches and use a moving average to compute the k^{th} minibatch.

$$b_k = 0.9 b_{k-1} + 0.1 \log p(\mathbf{y}|\tilde{\mathbf{s}}_k, \mathbf{a}))$$

There are two more tricks we can use to decrease variance: (1) Add an entropy term $H[\mathbf{s}]$ (2) With probability 0.5, set the sampled attention $\tilde{\mathbf{s}}$ to its expected value α.

We then get

$$\frac{\partial L_s}{\partial W} \approx \frac{1}{N} \sum_{n=1}^{N} \frac{\partial p(\mathbf{y}|\tilde{\mathbf{s}}^n, \mathbf{a})}{\partial W} + \lambda_r (\log p(\mathbf{y}|\tilde{\mathbf{s}}^n, \mathbf{a}) - b) \frac{\partial p(\tilde{\mathbf{s}}^n|\mathbf{a})}{\partial W} + \lambda_e \frac{\partial H[\tilde{\mathbf{s}}^n]}{\partial W}$$

where λ_r and λ_e are hyperparameters selected with cross-validation.

Thus, for Hard attention, ϕ simply selects one $\mathbf{a}_i$ by sampling from the Multinoulli distribution parametrized by α.

Now let's discuss soft attention. Here, we just compute this smooth and differentiable expression:

$$\phi(\{\mathbf{a}_i\}, \{\alpha_i\} = \sum_{i=1}^{L} \alpha_i \mathbf{a}_i$$

Now, the softmax ensures that $\sum_i \alpha_{t,i} = 1$, but we'd also like to ensure $\sum_t \alpha_{t,i} = 1$ to ensure each part of the image is looked at equally through time (we found this improved BLEU score and made better captions). To enable this for soft attention, we first set

$$\phi(\{\mathbf{a}_i\}, \{\alpha_i\} = \sigma(f_\beta(\mathbf{h}_{t-1})) \sum_{i=1}^{L} \alpha_i \mathbf{a}_i$$

for multi-layer perceptron f_β, which gives log likelihood:

$$L_d = -\log(P(\mathbf{y}|\mathbf{x})) + \lambda \sum_i^L (1 - \sum_t^C \alpha_{t,i})^2$$

We trained with RMSProp for Flickr8k and Adam for MS COCO and Flickr30k. Our CNN was VGG Net (pretrained on ImageNet, not finetuned). We use the output of the fourth conv layer, which is a $14 \times 14 \times 512$ feature map, which we flatten to get a 196×512 map ($L \times D$).

Since our model's running time is proportional to caption length, we ensure minibatches have captions of the same length. We trained for 3 days on an NVIDIA Titan Black. We used the Whetlab system for hyperparameter tuning.

7 Experiments

We set vocabulary size $K = 10000$. We report the BLEU and METEOR metrics. We don't use an ensemble. We set state of the art on all the datasets. Hard attention does better than soft attention.

To visualize the hard attention, look at the selected $\mathbf{a}_i$, unflatten it, scale it to the size of the image, and apply a Gaussian blur. The attention makes sense.

8 Conclusion

CNN + LSTM + Attention sets state of the art on BLEU and METEOR metrics for MS COCO, Flickr8k, and Flickr30k.