Show and Tell: A Neural Image Caption Generator

1 Citation

Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

`https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Vinyals_Show_and_Tell_2015_CVPR_paper.pdf`

2 Abstract

Our Neural Image Captioning (NIC) model combines a CNN with an LSTM to create descriptions of images. It gets state of the art BLEU scores (ours/previous/human) on PASCAL (59/29/69), Flikr30k (66/56/68), SBU (28/19/?), COCO (28/?/?).

3 Introduction

The goal is to take an image I and create a sentence of words $S = [S_1, S_2, ...]$ to describe the image.

In the world of translation, people solve this by using a Recurrent Neural Network (RNN) for encoding the input sequence into a fixed size state and then another RNN for decoding the state into a sequence. In our case, we get rid of the encoder RNN and use a Convolutional Neural Network (CNN) to encode the image into a fixed size vector.

4 Related Work

Some previous systems used And-Or graphs with custom rules. These were extremely brittle and did not generalize.

Graph-based systems are slightly better, but they require a lot of hand-engineered pieces.

Other work takes a predetermined set of descriptions and ranks them for a given image. We don't think this approach will generalize well.

Some work uses a feed-forward neural network or a general RNN (not an LSTM like we do). Recent work uses a two-stream approach with a CNN and LSTM, but they focus on ranking.

5 Model

We parametrize our model by θ. We seek:

$$\theta^* = \arg\max_\theta \sum_{(I,S)} \log p(S|I,\theta)$$

For a single sentence $S = [S_1, S_2, ..., S_N]$, the $p(S|I, \theta)$ decomposes into the following (we omit θ for brevity).

$$\log p(S|I) = \sum_{t=0}^{N} \log p(S_t|I, S_0, S_1, ..., S_{t-1})$$

We model the probability with a Long Short-Term Memory (LSTM) network, which takes the following form:

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

where i, f, and o are the input, forget, and output gates, respectively. σ is the sigmoid function. c is the memory cell of the LSTM.

Now, we need to pick our x_t. We begin by setting $x_{-1} = \text{CNN}(I)$. We use the GoogLeNet CNN. Then, we set $x_t = W_e S_t$ for all $t \in \{0...N-1\}$, where S_t is the one-hot encoded t^{th} word in the ground truth sentence and W_e is an embedding matrix. Notice that our image descriptor and word embedding vector must have the same size in order for this to work.

Our loss function is then:

$$L(I, S) = -\sum_{t=1}^{N} \log p_t(S_t)$$

We backpropagate to optimize the LSTM parameters, W_e, and top layer of the CNN.

At test time, we don't have a ground truth sentence, so how do we pick the x_t values for $t \neq -1$? One option is to sample from p_1 and and then feed the LSTM's predictions back into itself. A better option is Beam Search, where we keep track of the k best sentences of length t and use them to generate the best x_{t+1} (we keep the top k of these too). We set $k = 20$.

6 Experiments

We measure performance using BLEU-1 (or BLEU-4 for MS COCO). We also set up an Amazon Mechanical Turk experiment to ask humans to rate our descriptions as well. Another metric is perplexity, which we use for hyperparameter tuning, but we don't actually use as a final metric.

We train on SBU (large but noisy), MS COCO (great dataset), Flickr30k (smaller), Flickr8k (smaller) and evalate on held-out portions of each and on PASCAL.

We use a CNN pretrained on ImageNet and fine-tune the last layer. We tried training an embedding matrix W_e from a big news dataset, but it didn't help much so we just train it from scratch along with the LSTM. The LSTM and word embedding has 512 dimensions.

We get a huge improvemnt over state-of-the-art for SBU and PASCAL because the previous work did not use neural networks. We also set state of the art for the other datasets, but we aren't at human level performance yet.

We tried doing transfer learning between Flick30k and Flickr8k and find that the model transfers well and beats training from scratch on Flickr8k. Going from Flickr30k to MS COCO is not great (it's better to train from scratch), but the descriptions are still ok.

Looking at some of the top k beam search outputs, we see that captions look pretty good (and diverse too).

We also compare our system to other ranking based approaches and we do reasonably well on the ranking metrics as well.

The human evaluation of our model shows that it is worse than the human captions, more so than the BLEU scores would suggest. This indicates that BLEU is not a perfect alternative to human evaluation.

Looking at our word embeddings, we find that nearest words make sense. For example, ”car” is near ”van” and ”cab”.

7 Conclusion

NIC is an end-to-end neural network for image captioning. It combines a CNN with an LSTM to maximize the probability of a sentence. It does well on BLEU and human evaluation. Future work should see how unsupervised learning can improve the system.