Deep Visual-Semantic Alignments for Generating Image Descriptions

1 Citation

Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

`https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Karpathy_Deep_Visual-Semantic_Alignments_2015_CVPR_paper.pdf`

2 Abstract

We take image captioning datasets and build a model that can describe regions of an image. To do this, we learn an embedding of image regions and words in the same space using a Convolutional Neural Network (CNN) and a bidirectional Recurrent Neural Network (RNN). We align sentence fragments to image regions with a Markov Random Field. We use a multimodal RNN that uses our joint embedding representation to do image captioning . Our datasets are MS COCO, Flickr8k, and Flickr30k.

3 Introduction

Previous work uses small vocabularies, sentence templates, and rigid structures. We relax all these assumptions. We treat sentences are weak labels of images and learn how to align parts of the sentence to the corresponding parts of the image.

4 Related Work

Some work labeled images with fixed set of categories. Others posed image description as ranking the best captions from a dataset for a given image. Many people use RNNs, but our approach is simpler (it is not the most performant though). Others have worked on joint image/word embeddings. We had a previous paper that did this with dependency trees, but we don't use those in this paper. And of course, there are many applications of CNNs to the image domain and RNNs to the language domain.

5 Our Model

First, we train a model that aligns sentence snippets to image regions with a multimodal embedding. Second, we train a neural network that uses this embedding to caption images.

Our dataset consists of image + sentence pairs.

First, we use RCNN to detect objects in the image. We turn each detected object (represented by the pixels in its bounding box), I_b, into a descriptor as follows:

$$v = W_m(CNN_{\theta_c}(I_b)) + b_m$$

where CNN_{θ_c} is the CNN feature extractor that produces a 4096 length vector and W_m is the $h \times 4096$ embedding matrix.

To embed sentences, we use a bidirectional RNN (BRNN) instead of dependency trees. Let our N word sentence consist of 1-of-K encoded words: $\mathbb{I}_1, \mathbb{I}_2, ..., \mathbb{I}_N$. We get:

$$\begin{aligned} x_t &= W_w \mathbb{I}_t \\ e_t &= f(W_e x_t + b_e) \\ h_t^f &= f(e_t + W_f h_{t-1}^f + b_f) \\ h_t^b &= f(e_t + W_b h_{t+1}^b + b_b) \\ s_t &= f(W_d(h_t^b + h_t^f) + b_d) \end{aligned}$$

The first equation looks up the embedding for the word. The second applies a fully connected layer. The third and fourth equations compute the forward and backward hidden states. The final equation combines both hidden states to produce a descriptor. We initialzed W_w with word2vec, but you can train it from scratch and get the same performance. Notice that each s_t depends on all the words in the sentence. f is ReLU.

To measure how similar an image k and sentence l are to each other, we compute the following, where g_k represents the image regions and g_l represents the words:

$$S_{kl} = \sum_{t \in g_l} \max_{i \in g_k} v_i^T s_t$$

Assuming that $k = l$ represents an image-sentence pair from our dataset, our cost function is:

$$C(\theta) = \sum_k \left(\sum_l \max(0, S_{kl} - S_{kk} - 1) + \sum_l \max(0, S_{lk} - S_{kk} - 1) \right)$$

Now, given a sentence s and image v, how do we assign words to image regions? A naive approach to simply pair each word with the highest scoring $(v_i^T s_t)$ region. A better approach is to use a Markov Random Field to capture interactions between neighboring words. Given an N-word sentence and M-region image, introduce latent variables $a_j \in \{1...M\}$ for $j = 1...N$. We get:

$$\begin{aligned} E(\mathbf{a}) &= \sum_{j=1...N} \psi_j^U(a_j) + \sum_{j=1...N-1} \psi_j^B(a_j, a_{j+1}) \\ \psi_j^U(a_j = t) &= v_i^T s_t \\ \psi_j^B(a_j, a_{j+1}) &= \beta \mathbb{I}[a_j = a_{j+1}] \end{aligned}$$

where β is a hyperparameter. We minimize the energy E with dynamic programming.

To summarize our progress so far, we've found a joint embedding of image regions and words. We combined this with a Markov Random Field to align sentence words to image regions. Now let's see how we can do captioning of an image or image region.

For training data, we consider an image (or image region) I (we feed this through our CNN) and its associated sentence (or sentence fragment), $(x_1, x_2, ..., x_T)$ (these come from our BRNN). We then compute the following for $t = 1...T$.

$$b_v = W_{hi}[CNN_{\theta_c}(I)]$$
$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + \mathbb{I}[t = 1] \odot b_v)$$
$$y_t = \text{Softmax}(W_{oh}h_t + b_o)$$

We designate a special END word and add it to our vocabulary. The dimensionality of the hidden state is 512. We only use the image information, b_v, at the first timestep ($\mathbb{I}[t = 1] \odot b_v$).

At training time, we set $h_0 = \vec{0}$ and $x_1 =$ START (another special vocabulary word). We then run the RNN, feeding the ground truth word x_t as input for timestep t. We finish with the last word END. We maximize the log probability of predicting the words in the sentence.

At test time, we can't feed in ground truth words, so we sample from the softmax or take the argmax. An even better option is to do beam search with a beam size of 7.

We train with minibatch (100) SGD with momentum (0.9) where weight decay and learning rate are chosen with cross validation. We use dropout on all non-recurrent layers and clip gradients elementwise at 5 (important). It's hard to train the generative RNN, so we use RMSProp.

6 Experiments

Our preprocessing is: (1) lowercase, (2) remove non-alphanumeric, (3) discard words that occur less than 5 times in training data.

To evaluate alignment, we feed in either an image or sentence and query for the top-K (sorted by decreasing S_{kl}) vectors in opposite modality. Our metric is Recall@K. We use AlexNet as our CNN, and we beat other neural network approaches that also use AlexNet. We beat the dependency tree method.

Qualitatively, our alignments look good. Also, words that are important in identifying image regions (e.g. pumpkin, kayak) have high magnitude embedding vectors while low-importance words (e.g. now, actually) have near-zero magnitude.

We train our multimodal RNN on image/sentence pairs (i.e we don't use the alignment model). We measure BLEU, METEOR, and CIDEr. Our model tends to re-use sentence fragments from the training data. We beat a nearest-neighbors baseline model. We use VGG net this time and beat previous work.

We also trained our our multimodal RNN on the alignment model. We evaluate on a testing set (i.e. we asked for snippets for image regions) that we collected using Amazon Mechanical Turk. This model gives much better performance than training on the raw image/sentence pairs, but it tends to write longer sentences.

There are three key limitations in our model. First, it takes a fixed size image input. Second, it incorporates image information through an additive bias (b_v) rather than through a multiplication. Third, it consists of two separate neural networks (and a Markov Random Field) instead of one end-to-end differentiable model.

7 Conclusion

Our CNN + BRNN learns a good embedding of image regions and words. The Markov Random Field aligns sentence fragments to image regions. The multimodal RNN generates a caption for an image (or image region).