Victor Project AI: Image Description Generation with Disentangled Language and Description Models

Iacer Calixto and Wilker Aziz

November 7, 2018

Abstract

We present a MLE model of image description that mixes two components: a language model component (agnostic of visual context) and a description component that maps from visual input to words without relying on textual context. In order to better disentangle the two components we employ modelling techniques inspired by topic modelling literature. Furthermore, the mixture model treatment allows for a principled integration of text-only resources such as large unannotated text corpora, for example, by pre-training of the language model component.

Notation guidelines We use capital Roman letters (e.g. X) for random variables and their lowercase counterparts for assignments (e.g. x), x_1^n is shorthand for a sequence $\langle x_1, \ldots, x_n \rangle$ of length n, and $x_{<i}$ is shorthand for a prefix sequence of length $i - 1$ (when $i = 1$ the prefix $x_{<i}$ denotes an empty sequence), we use boldface letters (e.g. $\mathbf{w}$, $\mathbf{W}$) for deterministic vectors and matrices.

1 Image description generation

Image description generation is typically approached as a conditional language modelling task where an image provides additional context for an otherwise standard application of chain rule of probabilities. Figure 1 depicts the conditional independences of the model where a random word observation x depends directly on a deterministic image $\mathbf{v}$ and a random observed history $x_<$, moreover, θ collectively denotes the deterministic parameters of the model. In this framework, the probability $P_\theta(x_1^n|\mathbf{v})$ of a description x_1^n given an image $\mathbf{v}$ factorises without Markov assumptions as shown below.

$$P_\theta(x_1^n|\mathbf{v}) = \prod_{i=1}^{n} P_\theta(x_i|\mathbf{v}, x_1^m) = \prod_{i=1}^{n} \mathrm{Cat}(x_i|f(\mathbf{v}, x_{<i}; \theta)) \ . \quad (1)$$

At each step i, the information available (i.e. image $\mathbf{v}$ and prefix $x_{<i}$) is mapped to a categorical probability distribution over the vocabulary of the language

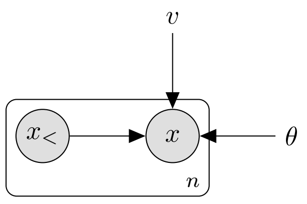


Figure 1: A fully supervised model of image description.

using a parametrised function $f(\cdot;\theta)$—typically a differentiable neural network architecture. Given a dataset of observations, the parameters θ of the model are point-estimated to attain a local optimum of the likelihood function using a stochastic gradient-based optimiser.

2 Latent visual clues

Our model is a maximum likelihood estimator. We have deterministic topic embeddings β_k for $k = 1, \ldots, K$

$$\beta_k \in \mathbb{R}^d. \tag{2}$$

We use image features to predict a sentence-level topic distribution,

$$\pi_0 = \text{softmax}(\alpha_0(v)) \tag{3}$$

that is, a K-dimensional probability vector. Then, for each word position $i = 1, \ldots, n$, we compute a switch

$$B_i = \text{sigmoid}(s(\mathbf{v}, z_0, x_{<i})) \tag{4a}$$

$$z_0 = \sum_{k=1}^{K} \pi_{0k}\beta_k \tag{4b}$$

that alternates between inserting words from language context (when $B_i \geq 0.5$) and mapping from visual clues (when $B_i < 0.5$). The language context ($B_i \geq 0.5$) is computed by

$$X_i = \text{Cat}(f(x_{<i})), \tag{5}$$

and the description context ($B_i < 0.5$) is computed by

$$\pi_i = \text{softmax}(\alpha(\mathbf{v}, z_0, x_{<i})) \tag{6a}$$

$$X_i = \text{Cat}(g(z_0, z_i)) \tag{6b}$$

$$z_i = \sum_{k=1}^{K} \pi_{ik}\beta_k \tag{6c}$$

expressed in $\mathbf{v}$ and also captured by z_0 and z_i. Note that the observed image $\mathbf{v}$ only affects the description component (6b) through the topic distributions π_0 and π_i and topic embeddings β_1^K.

3 Architecture

Function α_0 used to compute the sentence-level topic distributions (Equation (3)) is a two-layer feed-forward neural network with non-linear activation (e.g. ReLU), as in (7):

$$\begin{aligned}\mathbf{r}_1^1 &= ReLU(\mathbf{W}_1^1\mathbf{v} + \mathbf{b}_1^1),\\ \mathbf{r}_2^1 &= ReLU(\mathbf{W}_2^1\mathbf{r}_1^1 + \mathbf{b}_2^1).\end{aligned} \tag{7}$$

Function s used to compute the stochastic switch B_i (Equation (4a)) depends on whether image features $\mathbf{v}$ are global ($\mathbf{v} \in \mathbb{R}^v$) or local ($\mathbf{v} \in \mathbb{R}^{l\times v}$), where v is the feature dimensionality for each image patch, and $l = 49$ is the number of patches in the image and in practice depends on the actual pre-trained CNN being used.

If image features are global, we concatenate all available features together $[\mathbf{v}; z_0; h_i]$ (where h_i is the hidden state of the decoder for the current timestep) and use it as input to a two-layer feed-forward neural network with non-linear activation (e.g. ReLU), as in (8):

$$\begin{aligned}\mathbf{r}_1^2 &= ReLU(\mathbf{W}_1^2[\mathbf{v}; z_0; h_i] + \mathbf{b}_1^2),\\ \mathbf{r}_2^2 &= ReLU(\mathbf{W}_2^2\mathbf{r}_1^2 + \mathbf{b}_2^2).\end{aligned} \tag{8}$$

If image features are local, we concatenate the hidden state of the decoder for the current timestep with the global topic embedding $[z_0; h_i]$ and use it as a query to select $\mathbf{v}$ (implemented as an attention mechanism). Assuming local image features are vectorised as $\{\mathbf{v}_1, \mathbf{v}_2 \cdots, \mathbf{v}_l\}$, the attention mechanism can be computed as in (9):

$$\begin{aligned}\mathbf{q}_i &= [\mathbf{z}_0; \mathbf{h}_i],\\ e_{il} &= \text{score}(\mathbf{q}_i, \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_l\}),\\ \mathbf{v}_{att} &= \text{softmax}(e_{il}),\end{aligned} \tag{9}$$

where the score function can be implemented as the bilinear attention as in (10):

$$\text{score}(\mathbf{x}_i, \mathbf{y}_i) = \mathbf{x}_i\mathbf{W}_{xy}^1\mathbf{y}_i, \tag{10}$$

where W_{xy}^1 is a trained model parameter.

Language model component The language model component is a standard LSTM-RNN.

Description model component Similarly to function s described above (Equation (4a)), function α used to compute the mixing coefficients π_i (Equation (6a)) depends on whether image features $\mathbf{v}$ are global ($\mathbf{v} \in \mathbb{R}^v$) or local ($\mathbf{v} \in \mathbb{R}^{l\times v}$), where v is the feature dimensionality for each image patch, and $l = 49$ is the number of patches in the image and in practice depends on the actual pre-trained CNN being used.

If image features are global, we concatenate all available features together $[\mathbf{v}; z_0; h_i]$ (where h_i is the hidden state of the decoder for the current timestep) and use it as input to a two-layer feed-forward neural network with non-linear activation (e.g. ReLU), as in (11):

$$\begin{aligned}\mathbf{r}_1^3 &= ReLU(\mathbf{W}_1^3[\mathbf{v}; z_0; h_i] + \mathbf{b}_1^3),\\ \mathbf{r}_2^3 &= ReLU(\mathbf{W}_2^3\mathbf{r}_1^3 + \mathbf{b}_2^3).\end{aligned} \tag{11}$$

If image features are local, we concatenate the hidden state of the decoder for the current timestep with the global topic embedding $[z_0; h_i]$ and use it as a query to select $\mathbf{v}$ (implemented as an attention mechanism). Assuming local image features are vectorised as $\{\mathbf{v}_1, \mathbf{v}_2 \cdots, \mathbf{v}_l\}$, the attention mechanism can be computed as in (12):

$$\begin{aligned}\mathbf{q}_i &= [\mathbf{z}_0; \mathbf{h}_i],\\ e_{il} &= \text{score}(\mathbf{q}_i, \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_l\}),\\ \mathbf{v}_{att} &= \text{softmax}(e_{il}),\end{aligned} \tag{12}$$

where the score function can be implemented as the bilinear attention as described in (10), with distinct learned parameters W_{xy}^2. Finally, we either use $\mathbf{r}_2^3$ (global $\mathbf{v}$) or $\mathbf{v}_{att}$ (local $\mathbf{v}$) as input to yet another two-layer feed-forward neural network with non-linear activation (e.g. ReLU), as in (13):

$$\begin{aligned}\mathbf{q} &= \begin{cases}\mathbf{r}_2^3, & \text{if } \mathbf{v} \text{ are global features}\\ \mathbf{v}_{att}, & \text{otherwise}\end{cases}\\ \mathbf{r}_1^4 &= ReLU(\mathbf{W}_1^4\mathbf{q} + \mathbf{b}_1^4),\\ \mathbf{r}_2^4 &= ReLU(\mathbf{W}_2^4\mathbf{r}_1^4 + \mathbf{b}_2^4),\end{aligned} \tag{13}$$

Finally, function $g()$ in Equation (6b) can be implemented as a two-layer feed-forward neural network with non-linear activation (e.g. ReLU) as in (14):

$$\begin{aligned}\mathbf{r}_1^5 &= ReLU(\mathbf{W}_1^5[z_0; z_i] + \mathbf{b}_1^5),\\ \mathbf{r}_2^5 &= ReLU(\mathbf{W}_2^5\mathbf{r}_1^5 + \mathbf{b}_2^5),\end{aligned} \tag{14}$$

which output can be fed into a softmax activation to obtain the next word probabilities.

4 Schedule

4.1 Original Plan

First I will take two to three weeks for implementing one or two baseline models. Two weeks will be used for implementing and tuning the novel multimodel.
Finally four to 5 weeks will be used to do an analysis of the model. Each of the different components will be analyst on it contribution to the results. Furthermore, the report will be written during this time.

4.2 Updated Plan

At this point, after 3 weeks of time, the baseline model (without attention) is implemented. I have been working on the implementation of the validation, to make sure it is comparable to that of the show and tell paper.

- **Done on Sept. 7. However, just notized it is not optimizing**I will need two days to finish the validation and make sure it will run on the LISA surfer. (since we were able to quickly get access, this will be better and easier to setup).

IC: Implement model selection as described further. First (re-)implement the baseline so that it reproduces the results reported in the paper. Only then move on to implementing our own model.

  - Fix model selection using validation/development data. Do this at the end of every epoch. Use patience as described further.
  - Make sure mini-batching works as expected.
  - Make sure beam search works as expected.
  - Perhaps run experiments on a smaller data set (e.g. Flickr8k or Flickr30k) to be able to get results faster.
- **Sept 11** discuss and fix not training problem.
- **Sept 7 - Sept 19** Three to five days will be used for implementing the novel model, with the binary switch and the topic models.
- **Sept 20 - Sept 21** I will implement the code for tuning and finding optimal hyper parameters. Maybe a grid search.
- **Sept 21 - Sept 25** Fix the jobscripts for advanced running.
  - run beyond the walltime of 48 hours.
  - Use the multiple GPU's in a single node.
  - maybe use multiple nodes?
- **Sept 26 - Oct 17, here is space to use less time, if previous steps would delay.** During the time the models are training, I will start writing the report.
  - Related Work
  - Baseline Implementation
  - Training setup
  - (start of) Introduction
- **Oct 17 - Oct 18** After the training is done, I will start an analysis of the results.
  - Create tables of the results. Show clear comparison between different datasets/papers/models.
  - Plot figures showing the learning processes.

- **Oct 23 - Nov 9** After finding the best model, I will do the analysis of different parts of the model.
  - Did final layer of CNN optimize.
  - What does de binary switch do? does it favor one of the values? Is there a pattern in switching?
  - What is the performance of the language model?
  - what is the performance of the description model.
- **Oct 24 - Nov 16** Finish writing the report.
  - write about the analysis (in parallel with previous step).
  - Experiment Results, including the figures.
  - Conclusion.
  - Finish Introduction.
  - Dubbel Check, grammar check, more checks.

5 Show and Tell settings

1. held out 4k images from validation for testing.
2. trained on BLEU.
3. Also report Meteor and Cider to help the discussion.
4. Perplexity also calculated for tuning purposes, not reported.
5. Initialized with pretrained CNN on ImageNet.
6. Used Dropout and Ensembling (to avoid overfitting).
7. Embedding size = 512
8. Size of LSTM memory = 512
9. beamsearch = 20

6 Resources

https://github.com/elliottd/satyrid
https://github.com/KranthiGV/Pretrained-Show-and-Tell-model
http://cocodataset.org/#download
https://github.com/tensorflow/models/tree/master/research/im2txt
https://link.springer.com/article/10.1007/s11263-015-0816-y?sa_campaign=email/event/articleAuthor/onlineFirst#

7 Other Details

One of the hyper-parameters of the model is the patience. The patience is the number of epochs the model is allowed to train on the training data without improving on the development data. This assumes that the model is validated at the end of every training epoch. This improvement is measured in BLEU. The final model is the best performing model on the development data according to BLEU.

8 Results

Some preliminary results of Baseline. currently using:

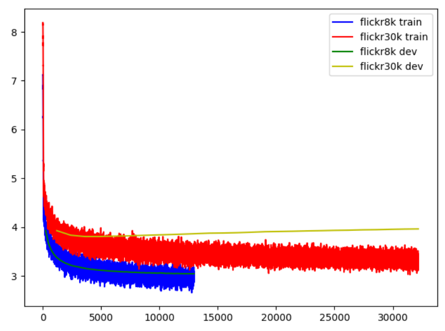
- batch size of 25 images(125 captions)
- 1 lstm layer
- dropout after embeddings, p=0.5
- gain clipping
- tried Adam, adagrad and RMSProp

Because of the difficulty training, as can be seen in Figure 2, I made the following updates:

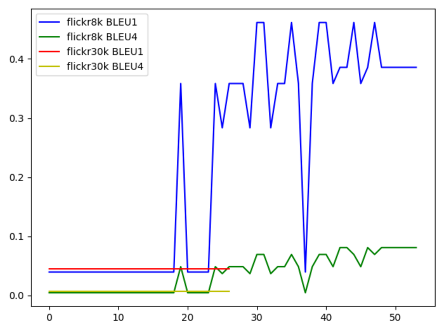
- implemented dataloader and data into dataset. 5 captions of the same image not required to be in the same batch anymore.
- can now parallize over multiple gpus.
- 2 lstm layers.
- weight initialization

The reasoning behind these updates is that I suspect the model is underfitting. It doesn't learn to generate different sentences for different images. It constantly generates: `a man in a <unk> .`, or something similar. furthermore, the loss decreases rapidly and then becomes constant. The results in Figure 2 are for Adam with grad clipping at a value of 5.

References



(a) Losses.



(b) Scores.

Figure 2: Results for the Flickr datasets.