Show and Tell

Explanantion accompaniment of solution code for Lab2 of ScaDaMaLe

Colin Desmarais
Department of Mathematics
Uppsala University
colin.desmarais@math.uu.se

Daniel Gedon
Department of Information Technology
Uppsala University
daniel.gedon@it.uu.se

November 18, 2020

1 Overview of the implementation

We implemented an image caption generator from the methods outlined in [2], we followed the following tutorial [1] as a guideline. Our PyTorch implementation is freely available at github.com/dgedon/ShowAndTell. As an 'encoder' we used a ResNet-152 from the Torchvision library pre-trained on the ImageNet dataset. A linear transformation is then applied to the extracted feature vector from the ResNet blocks. As for the decoder, we used an LSTM. The model was trained on COCO 2014 data set. Inference is performed using BeamSearch, where a beam of size 1 simply corresponds to greedy sampling.

2 Explanation of the code

`vocabulary.py` : contains functions needed to manipulate the list of words, from loading the dictionary, to extracting list of indices for a sentence, and producing a sentence from a list of indices.

`utils.py` : contains functions necessary for training.

`models.py` : contains the encoder and decoder.

- `class Encoder` : pretrained ResNet-152 is retrieved. The last trained linear layer is replaced by a randomly initialised one fitting our output size
  - `def forward` : **input**: source image. **output**: feature vector of source image. The image is fed through ResNet, the resulting output fed through a single linear layer, resulting feature vector returned.
- `class Decoder` : LSTM defined, as well as single linear layer and word embeddings.
  - `forward` : **input**: visual features of batch, captions of batch, lengths of captions of batch. **output**: predictions of captions. The visual features and embeddings of the captions are combined, then fed into LSTM. Different to [2] we use a dropout of 0.3 for the embeddings which yielded empirically better validation loss. The output is fed into a single linear layer, and the resulting vectors for each batch element returned. Note that we need to pad the captions for the longest caption in the sentence. Since the padding tokens are therefore the most common token, we have to avoid only learning the paddings. This is done by combining the input sequence with the specific torch function `pack_padded_sequence` which avoids this.
  - `sample` : **input:** visual features of image. **output**: A cation for the image. The sample function used greedy search to sample the caption.
  - `beam_search_sample` **input**: visual features of image, integer beam_size. **output**: A caption for the image. This function uses BeamSearch to sample a caption, using beams of size beam_size.

`main.py`: this file contains the training, testing, and sample of images.

- File starts with list of arguments that can be passed on to main.

- Images are pre_processed using `preprocess_images` from `utils.py`. Different to [2] we resize the input images to images of size 256×256. However, the input size to the ResNet is 224×224. Hence, we apply random cropping and random horizontal flipping as data augmentation techniques.
- vocabulary retrieved from the captions using `get_vocabulary` from `vocabulary.py`. We only take a word into the vocabulary if it appears at least 5 times in the training data captions. We use the natural language toolkit `nltk` tokenizer.
- encoder and decoder initialized defined by calling `Encoder` and `Decoder` from `models.py`
- If no_training set to False:
  - Adam optimizer defined
  - training performed using `train` from `utils.py`
  - model saved using `save_model` from `utils.py`
- If no_training set to True:
  - prepare image by resizing to correct input size
  - extract visual features using encoder
  - produce caption ids using `beam_search_sample`
  - extract caption from caption ids using `vocab.decode` from `vocabulary.py`

How to use the code:

- For training: pass arguments to main, set no_training to False.
- For captioning a single image: set path_test_image to the path of image to caption, set no_training to True.

3 Results

A model was trained with the following parameters:

- batch size: 128
- epochs: 5
- learning rate: 0.001
- minimum caption length (vocab_threshold): 5
- maximum caption length (max_words): 25

The following 6 pictures were chosen at random from the COCO dataset, with captions sampled with beam size (beam_size) of 5.

References

[1] Yunjey Choi. *Image Captioning*. URL: `https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning` (visited on 11/2020).

[2] Oriol Vinyals et al. *Show and Tell: A Neural Image Caption Generator*. 2015. arXiv: `1411.4555 [cs.CV]`.

Figure 1: a bride and groom are talking on a cell phone

Figure 2: a table with a plate of food and a cup of coffee

Figure 3: a train traveling down train tracks next to a railway platform

Figure 4: a man holding a tennis racquet on top of a tennis court

Figure 5: a group of people riding on the back of motorcycles

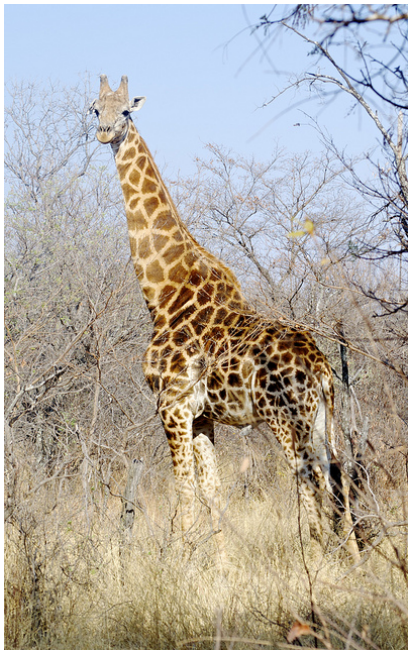

Figure 6: giraffe standing in the middle of a field