INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

CSE471

Statistical Methods in AI

Describing Images

Submitted By:	*Roll No. :*
Kshitij Paliwal	2018201063
Anand Sagar Sethi	2018201100
Aishwary Dewangan	2018202016
Indraneel Das	2018202019

Submitted To:
Dr. Santosh Ravi Kiran

Mentored By:
Mr. Ranajit Saha

Contents

1 Introduction

A quick glance at an image is sufficient for a human to point out and describe an immense amount of details about the visual scene . However, this remarkable ability has proven to be an elusive task for our visual recognition models. Generating descriptions for images has long been regarded as a challenging perception task integrating vision, learning and language understanding. One not only needs to correctly recognize what appears in images but also incorporate knowledge of spatial relationships and interactions between objects. Even with this information, one then needs to generate a description that is relevant and grammatically correct. With the recent advances made in deep neural networks, tasks such as object recognition and detection have made significant breakthroughs in only a short time.

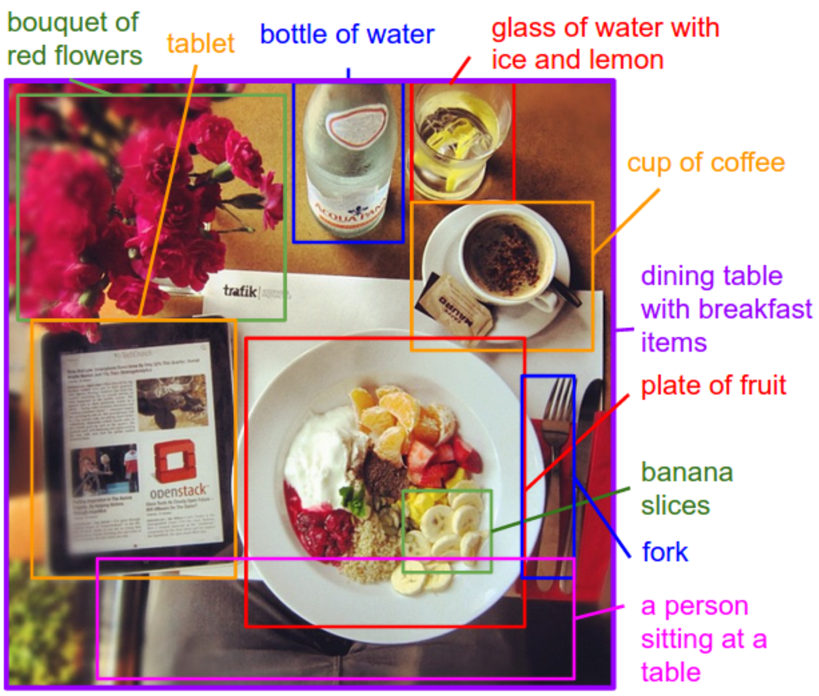
bouquet of red flowers
tablet
bottle of water
glass of water with ice and lemon
cup of coffee
dining table with breakfast items
plate of fruit
banana slices
fork
a person sitting at a table

Figure 1. Motivation/Concept Figure: Our model treats language as a rich label space and generates descriptions of image regions.

Some pioneering approaches that address the challenge of generating image descriptions have been developed [2]. However, these models often rely on hard-coded visual concepts and sentence templates, which imposes limits on their variety..

The primary challenge towards this goal is in the design of a model that is rich enough to simultaneously reason about contents of images and their representation in the domain of natural language. Additionally, the model should be free of assumptions about specific hard-coded templates, rules or categories and instead rely on learning from the training data. The second, practical challenge is that data sets of image captions are available in large quantities on the internet, but these descriptions multiplex mentions of several entities whose locations in the images are unknown.

Our core insight is that we can leverage these large image sentence data sets by treating the sentences as weak labels, in which contiguous segments of words correspond to some particular, but unknown location in the image. Our approach is to infer these alignments and use them to learn a generative model of descriptions.

2 Literature Survey

2.1 Deep Visual-Semantic Alignments for Generating Image Descriptions [1]

They present a model that generates natural language descriptions of images and their regions. Their approach lever- ages datasets of images and their sentence descriptions to learn about the inter-modal correspondences between lan- guage and visual data. Their alignment model is based on a novel combination of Convolutional Neural Networks over image regions, bidirectional Recurrent Neural Networks over sentences, and a structured objective that aligns the two modalities through a multimodal embedding. They then describe a Multimodal Recurrent Neural Network architec- ture that uses the inferred alignments to learn to generate novel descriptions of image regions. They demonstrate that their alignment model produces state of the art results in re- trieval experiments on Flickr8K, Flickr30K and MSCOCO datasets.They then show that the generated descriptions sig- nificantly outperform retrieval baselines on both full images and on a new dataset of region-level annotations.

2.2 Using Deep Features of Only Objects to Describe Images [3]

Inspired by recent advances in leveraging multiple modalities in machine translation, they introduce an encoder- decoder pipeline that uses specific objects within an image and their object labels a language model for decoding joint embedding of object features and the object labels. Their pipeline merges prior detected objects from the image and their object labels and then learns the sequences of captions describing the particular image. The decoder model learns to extract descriptions for the image from scratch by decoding the joint representation of the object visual features and their object classes conditioned by the encoder component. The idea of the model is to concentrate only on the specific objects of the image and their labels for generating descriptions of the image rather than visual feature of the entire image.

3 Methodology

The ultimate goal of our model is to generate descriptions of image regions. During training, the input to our model is a set of images and their corresponding sentence descriptions. We first present a model that aligns sentence snippets to the visual regions that they describe through a multi-modal embedding. We then treat these correspondences as training data for a second, multi-modal Recurrent Neural Network model that learns to generate the snippets.

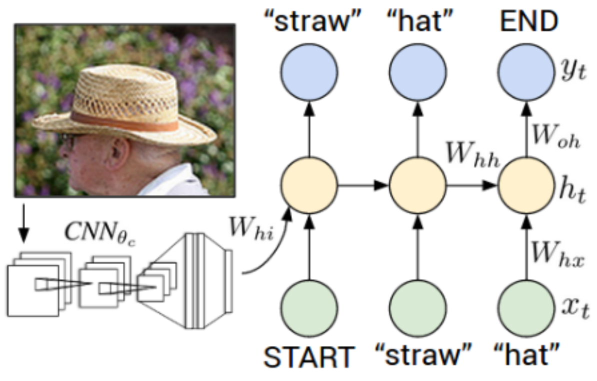

Figure 2. Workflow of our model

Our multimodal Recurrent Neural Network generative model. The RNN takes a word, the context from previous time steps and defines a distribution over the next word in the sentence. The RNN is conditioned on the image information at the first time step. START and END are special tokens.

3.1 Detailed Analysis

We will use the VGG16 model that has been pre-trained for classifying images. But instead of using the last classification layer, we will redirect the output of the previous layer. This gives us a vector with 4096 elements that summarizes the image-contents - similar to how a "thought-vector" summarized the contents of an input-text. We will use this vector as the initial state of the Gated Recurrent Units (GRU). However, the internal state-size of the GRU is only 512, so we need an intermediate fully-connected (dense) layer to map the vector with 4096 elements down to a vector with only 512 elements.

The decoder then uses this initial-state together with a start-marker ”ssss” to begin producing output words. In the first iteration it will hopefully output the word ”big”. Then we input this word into the decoder and hopefully we get the word ”brown” out, and so on. Finally we have generated the text ”big brown bear sitting eeee” where ”eeee” marks the end of the text.

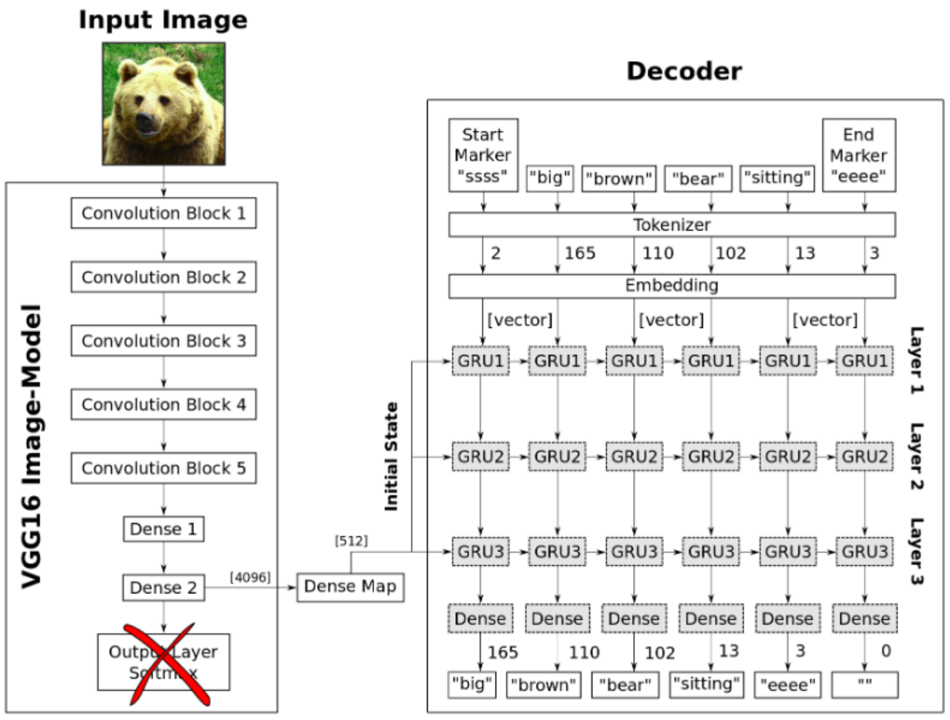

Figure 3. Flow chart pf our algorithm

We process all images in the data-set using the pre-trained image-model and saving the transfer-values in a cache-file so they can be reloaded quickly.

We effectively create a new data-set of the transfer-values. This is because it takes a long time to process an image in the VGG16 model. We will not be changing all the parameters of the VGG16 model, so every time it processes an image, it gives the exact same result. We need the transfer-values to train the image-captioning model for many epochs, so we save a lot of time by calculating the transfer-values once and saving them in a cache-file.

3.2 Tokenization

Neural Networks cannot work directly on text-data. We use a two-step process to convert text into numbers that can be used in a neural network. The first step is to convert text-words into so-called integer-tokens. The second step is to convert integer-tokens into vectors of floating-point numbers using a so-called embedding-layer.

Before we can start processing the text, we first need to mark the beginning and end of each text-sequence with unique words that most likely aren't present in the data.

3.3 Data Generation

Each image in the training-set has at least 5 captions describing the contents of the image. The neural network will be trained with batches of transfer-values for the images and sequences of integer-tokens for the captions. If we were to have matching numpy arrays for the training-set, we would either have to only use a single caption for each image and ignore the rest of this valuable data, or we would have to repeat the image transfer-values for each of the captions, which would waste a lot of memory.

A better solution is to create a custom data-generator for Keras that will create a batch of data with randomly selected transfer-values and token-sequences.

One epoch is a complete processing of the training-set. We would like to process each image and caption pair only once per epoch. However, because each batch is chosen completely at random in the above batch-generator, it is possible that an image occurs in multiple batches within a single epoch, and it is possible that some images may not occur in any batch at all within a single epoch. So we need to manually calculate the approximate number of batches required per epoch.

3.4 Recurrent Neural Network Architecture

We will now create the Recurrent Neural Network (RNN) that will be trained to map the vectors with transfer-values from the image-recognition model into sequences of integer-tokens that can be converted into text.

We want to use the transfer-values to initialize the internal states of the GRU units. This informs the GRU units of the contents of the images. The transfer-values are vectors of length 4096 but the size of the internal states of the GRU units are only

512, so we use a fully-connected layer to map the vectors from 4096 to 512 elements. We use a tanh activation function to limit the output of the mapping between -1 and 1, otherwise this does not seem to work.

The GRU layers output a tensor with shape [batch_size, sequence_length, state_size], where each "word" is encoded as a vector of length state_size. We need to convert this into sequences of integer-tokens that can be interpreted as words from our vocabulary.

One way of doing this is to convert the GRU output to a one-hot encoded array. It works but it is extremely wasteful, because for a vocabulary of e.g. 10000 words we need a vector with 10000 elements, so we can select the index of the highest element to be the integer-token. The activation-function is set to linear instead of softmax as we would normally use for one-hot encoded outputs, because there is apparently a bug in Keras so we need to make our own loss-function, as described in detail further below.

3.5 Loss Function

The output of the decoder is a sequence of one-hot encoded arrays. In order to train the decoder we need to supply the one-hot encoded arrays that we desire to see on the decoder's output, and then use a loss-function like cross-entropy to train the decoder to produce this desired output.

However, our data-set contains integer-tokens instead of one-hot encoded arrays. Each one-hot encoded array has 10000 elements so it would be extremely wasteful to convert the entire data-set to one-hot encoded arrays. We could do this conversion from integers to one-hot arrays in the batch generation.

A better way is to use a so-called sparse cross-entropy loss-function, which does the conversion internally from integers to one-hot encoded arrays. Unfortunately, there seems to be a bug in Keras when using this with Recurrent Neural Networks, so inbuilt loss sparse cross entropy loss function does not works.

The decoder outputs a 3-rank tensor with shape [batch_size, sequence_length, num_words] which contains batches of sequences of one-hot encoded arrays of length num_words. We will compare this to a 2-rank tensor with shape [batch_size, sequence_length] containing sequences of integer-tokens.

This comparison is done with a sparse-cross-entropy function directly from TensorFlow. There are several things to note here.

Firstly, the loss-function calculates the softmax internally to improve numerical accuracy - this is why we used a linear activation function in the last dense-layer of the decoder-network above.

Secondly, the loss-function from TensorFlow will output a 2-rank tensor of shape [batch_size, sequence_length] given these inputs. But this must ultimately be reduced to a single scalar-value whose gradient can be derived by TensorFlow so it can be optimized using gradient descent. Keras supports some weighting of loss-values across the batch but the semantics are unclear so to be sure that we calculate the loss-function across the entire batch and across the entire sequences, we manually calculate the loss average.

4 Conclusion

We introduced a model that generates natural language descriptions of image regions based on weak labels in form of a dataset of images and sentences, and with very few hardcoded assumptions. Our approach features a novel ranking model that aligned parts of visual and language modalities through a common, multimodal embedding. We showed that this model provides state of the art performance on image-sentence ranking experiments.

We used a pre-trained image-model (VGG16) to generate a ”thought-vector” of what the image contains, and then we trained a Recurrent Neural Network to map this ”thought-vector” to a sequence of words.

This works reasonably well, although it is easy to find examples both in the training- and validation-sets where the captions are incorrect.

It is also important to understand that this model doesn't have a human-like understanding of what the images contain. If it sees an image of a giraffe and correctly produces a caption stating that, it doesn't mean that the model has a deep understanding of what a giraffe is; the model doesn't know that it's a tall animal that lives in Africa and Zoos.

The model is merely a clever way of mapping pixels in an image to a vector of floating-point numbers that summarize the contents of the image, and then map these numbers to a sequence of integers-tokens representing words. So the model is basically just a very advanced function approximator rather than human-like intelligence.

5 Future Scope

- The model needs to be calibrated more by adjusting the parameters and settings to result in better accuracy and performance.
- The model needs to trained on bigger data set to produce better results.As due to shortage of resources we were not able to train the model on bigger data set.

References

[1] Deep visual-semantic alignments for generating image descriptions. `https://cs.stanford.edu/people/karpathy/cvpr2015.pdf`.

[2] G. kulkarni, v. premraj, s. dhar, s. li, y. choi, a. c. berg,and t. l. berg. baby talk: Understanding and generating simple image descriptions. in cvpr, 2011.

[3] Using deep features of only objects to describeimages. `https://arxiv.org/pdf/1902.09969.pdf`.

Git Repo: https://gitlab.com/anand13sethi/pagerank-optimization