NLP Reading Group — Meeting 4

Matt Smith

UW Data Science Club

21 March, 2019

Coverage

1. Deep contextualized word representations
2. BERT
3. Deep Visual-Semantic Alignments for Generating Image Descriptions

From last time

Continuing from last time

We read the abstract of the ELMO paper, saw a picture of the paper and explained how it worked

There was a question from last time that asked about the difference between high-level and low-level language features

I looked ahead in the slides and found that the difference is mentioned in the part we're reading, so we'll get there soon :)

Section 1 – Introduction

Our representations differ from traditional word type embeddings in that each token is assigned a representation that is a function of the entire input sentence. We use vectors derived from a bidirectional LSTM that is trained with a coupled language model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised word sense disambiguation tasks) while lower-level states model aspects of syntax (e.g., they can be used to do part-of-speech tagging). Simultaneously exposing all of these signals is highly beneficial, allowing the learned models select the types of semi-supervision that are most useful for each end task.

Section 1 – Introduction

Extensive experiments demonstrate that ELMo representations work extremely well in practice. We first show that they can be easily added to existing models for six diverse and challenging language understanding problems, including textual entailment, question answering and sentiment analysis. The addition of ELMo representations alone significantly improves the state of the art in every case, including up to 20% relative error reductions. For tasks where direct comparisons are possible, ELMo outperforms CoVe (McCann et al., 2017), which computes contextualized representations using a neural machine translation encoder. Finally, an analysis of both ELMo and CoVe reveals that deep representations outperform those derived from just the top layer of an LSTM. Our trained models and code are publicly available, and we expect that ELMo will provide similar gains for many other NLP problems.

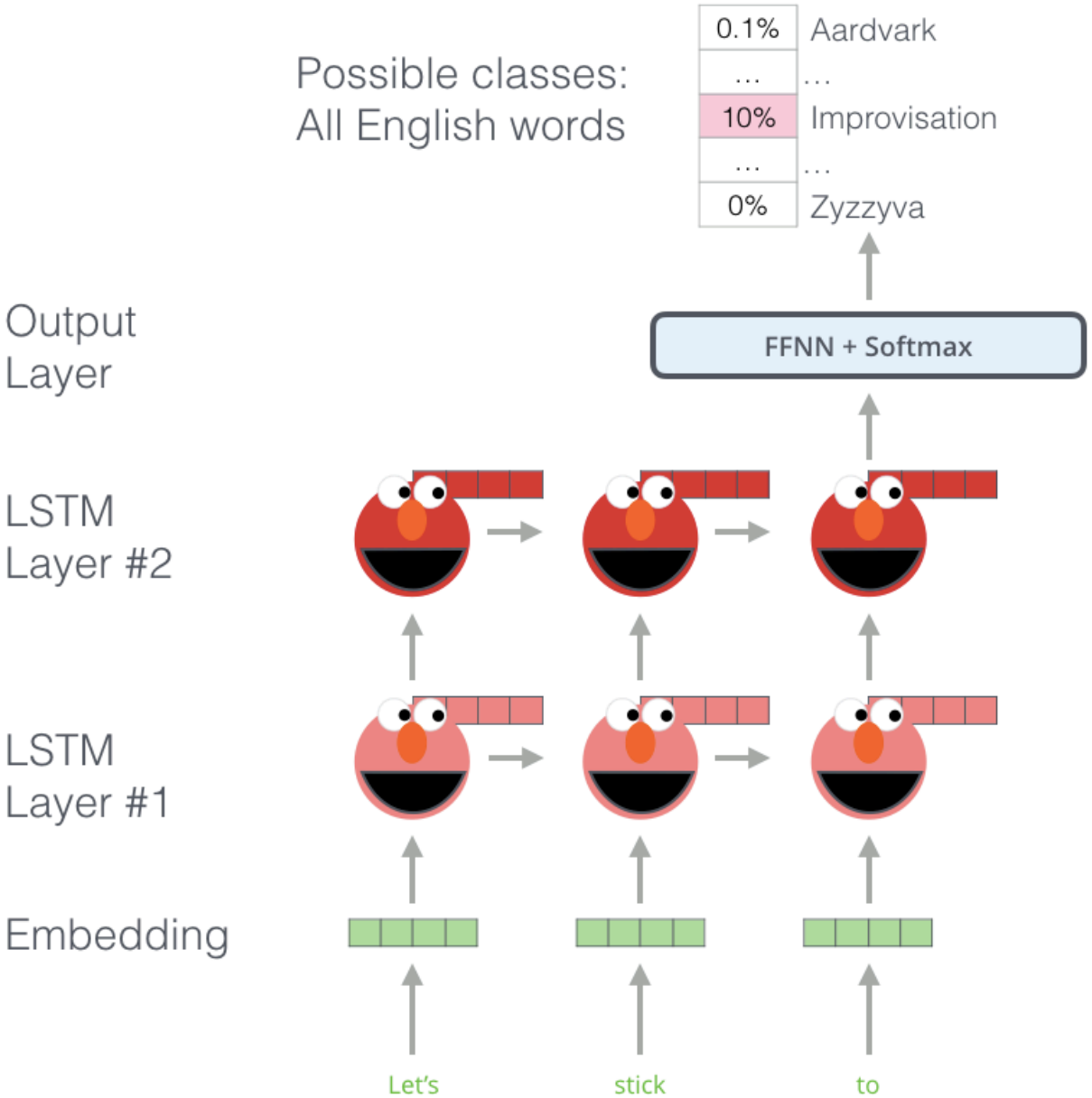
Section 2 – Related Work

Other recent work has also focused on learning context-dependent representations. context2vec (Melamud et al., 2016) uses a bidirectional Long Short Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) to encode the context around a pivot word.

...

Previous work has also shown that different layers of deep biRNNs encode different types of information. For example, introducing multi-task syntactic supervision (e.g., part-of-speech tags) at the lower levels of a deep LSTM can improve overall performance of higher level tasks such as dependency parsing (Hashimoto et al., 2017) or CCG super tagging (Søgaard and Goldberg, 2016). In an RNN-based encoder-decoder machine translation system, Belinkov et al. (2017) showed that the representations learned at the first layer in a 2- layer LSTM encoder are better at predicting POS tags then second layer.

Model


Possible classes:
All English words
0.1%
Aardvark
...
...
10%
Improvisation
...
...
0%
Zyzzyva
Output Layer
FFNN + Softmax
LSTM Layer #2
LSTM Layer #1
Embedding
Let's
stick
to

Getting embeddings from ELMO

The embedding of the tth word is the weighted sum of each layer's output at time t.

For example: Suppose there are L layers. If the output of layer i at word t is x_i^t, then the output embedding for the ith word is

$$\gamma \sum_{i=i}^{L} s_i x_i^t$$

γ and s_i are trainable in the model that uses ELMO weights, while the rest of the weights in the model are fixed.

Other Embeddings

So far we have seen:

- *A Neural Probabilistic Language Model*
- word2vec
- ELMO

I think today the two most common word embeddings are these, which we might not cover:

- GloVe
- fastText

These are fixed embeddings like word2vec and unlike ELMO, but they perform very wel without ELMO's performance considerations.

Coverage

1. Deep contextualized word representations
2. BERT
3. Deep Visual-Semantic Alignments for Generating Image Descriptions

Idea

Combines:

- ELMO
- OpenAI-style transformer
- Bidirectional transformers via bidirectional masking
- ULMfit (finetuning LMs)

Homework

Practice: Read the BERT paper. `https://arxiv.org/abs/1810.04805`

Exercises:

- Familiarize yourself with CNNs
- Use the TensorflowHub version of ELMO to create a NLP model for a task of your choice, compare it with other embeddings
- Implement ELMO
- Try to reproduce the results from *Neural Machine Translation by Jointly Learning to Align and Translate*, either on the paper's dataset or try generalizing the results to your own dataset

Coverage

1. Deep contextualized word representations
2. BERT
3. Deep Visual-Semantic Alignments for Generating Image Descriptions

Summary

An excellent paper about image captioning by Andrej Karpathy and Li Fei-Fei.

The goal of the paper:

- Take an image as input
- Produce a caption describing the image
- Use attention to show how the caption was generated

Example

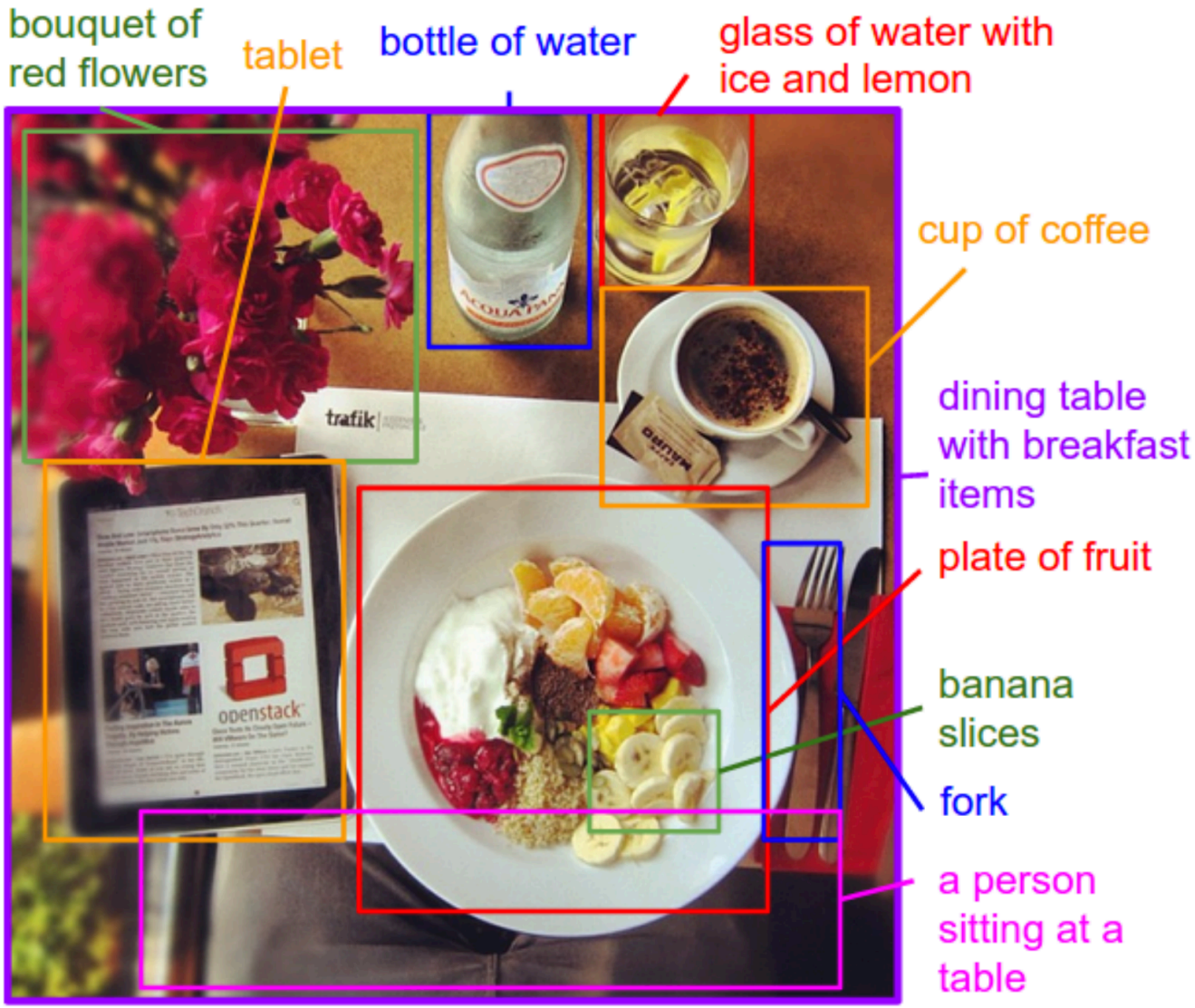


Figure 1. Motivation/Concept Figure: Our model treats language as a rich label space and generates descriptions of image regions.

Training

Trained on Flickr8K, Flickr30K, and MSCOCO.

Flickr datasets are collections of pictures + captions from the photo sharing website `flickr.com`.

AFAIK they have been superseded by the Yahoo-Flickr datatset YFCC100M (note that 100M $>>$ 30K).

COCO is a very popular and general dataset for image-based tasks

What is COCO?

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- **Object segmentation**
- **Recognition in context**
- **Superpixel stuff segmentation**
- **330K images (>200K labeled)**
- **1.5 million object instances**
- **80 object categories**
- **91 stuff categories**
- **5 captions per image**
- **250,000 people with keypoints**

Results

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

Results

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

Results

"woman is holding bunch of bananas."

"black cat is sitting on top of suitcase."

"a woman holding a teddy bear in front of a mirror."

"a horse is standing in the middle of a road."

Results

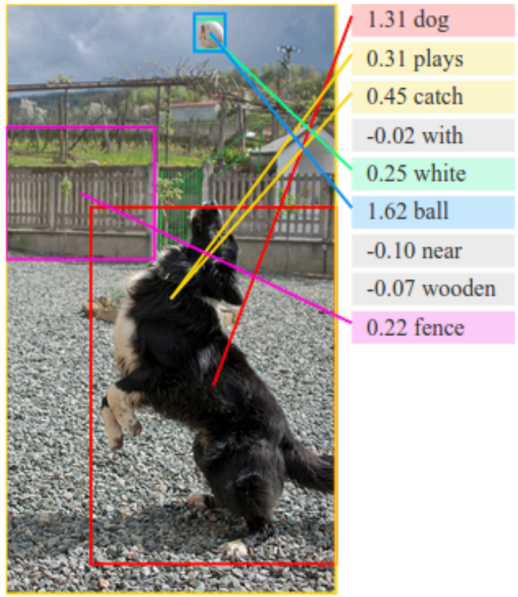
"little girl is eating piece of cake."

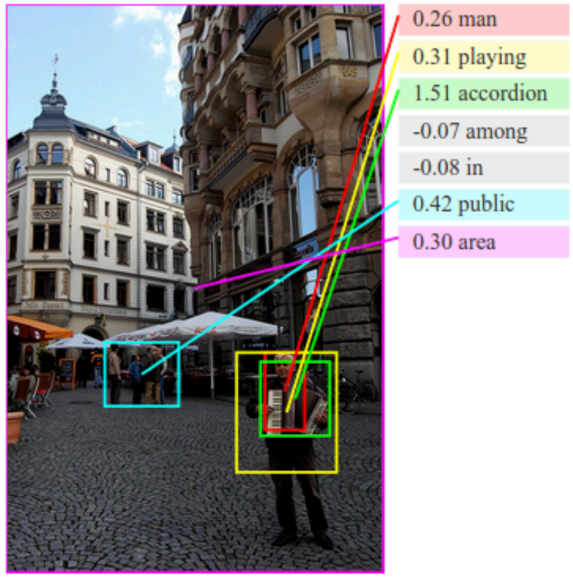
"baseball player is throwing ball in game."

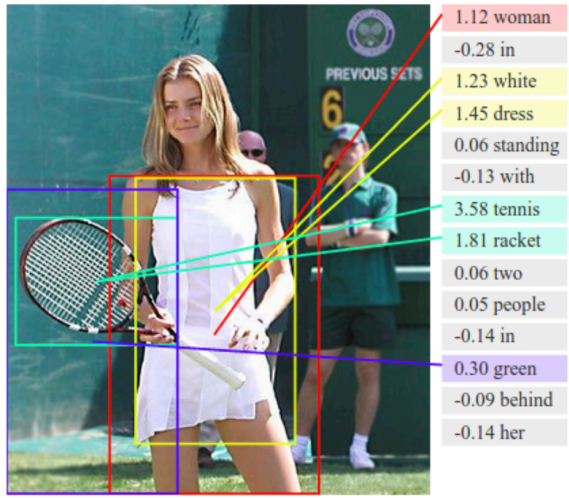
"a young boy is holding a baseball bat."

"a cat is sitting on a couch with a remote control."

Results


1.31 dog
0.31 plays
0.45 catch
-0.02 with
0.25 white
1.62 ball
-0.10 near
-0.07 wooden
0.22 fence


0.26 man
0.31 playing
1.51 accordion
-0.07 among
-0.08 in
0.42 public
0.30 area


1.12 woman
-0.28 in
1.23 white
1.45 dress
0.06 standing
-0.13 with
3.58 tennis
1.81 racket
0.06 two
0.05 people
-0.14 in
0.30 green
-0.09 behind
-0.14 her

How did they do this?

Fusion of a lot of ideas from NLP and CV (Computer Vision):

- CNNs
- Language models
- Sequence attention
- Visual attention
- Region proposals

Abstract

We present a model that generates natural language descriptions of images and their regions. Our approach leverages datasets of images and their sentence descriptions to learn about the inter-modal correspondences between language and visual data. Our 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. We then describe a Multimodal Recurrent Neural Network architecture that uses the inferred alignments to learn to generate novel descriptions of image regions. We demonstrate that our alignment model produces state of the art results in retrieval experiments on Flickr8K, Flickr30K and MSCOCO datasets. We then show that the generated descriptions significantly outperform retrieval baselines on both full images and on a new dataset of region-level annotations.

Goal

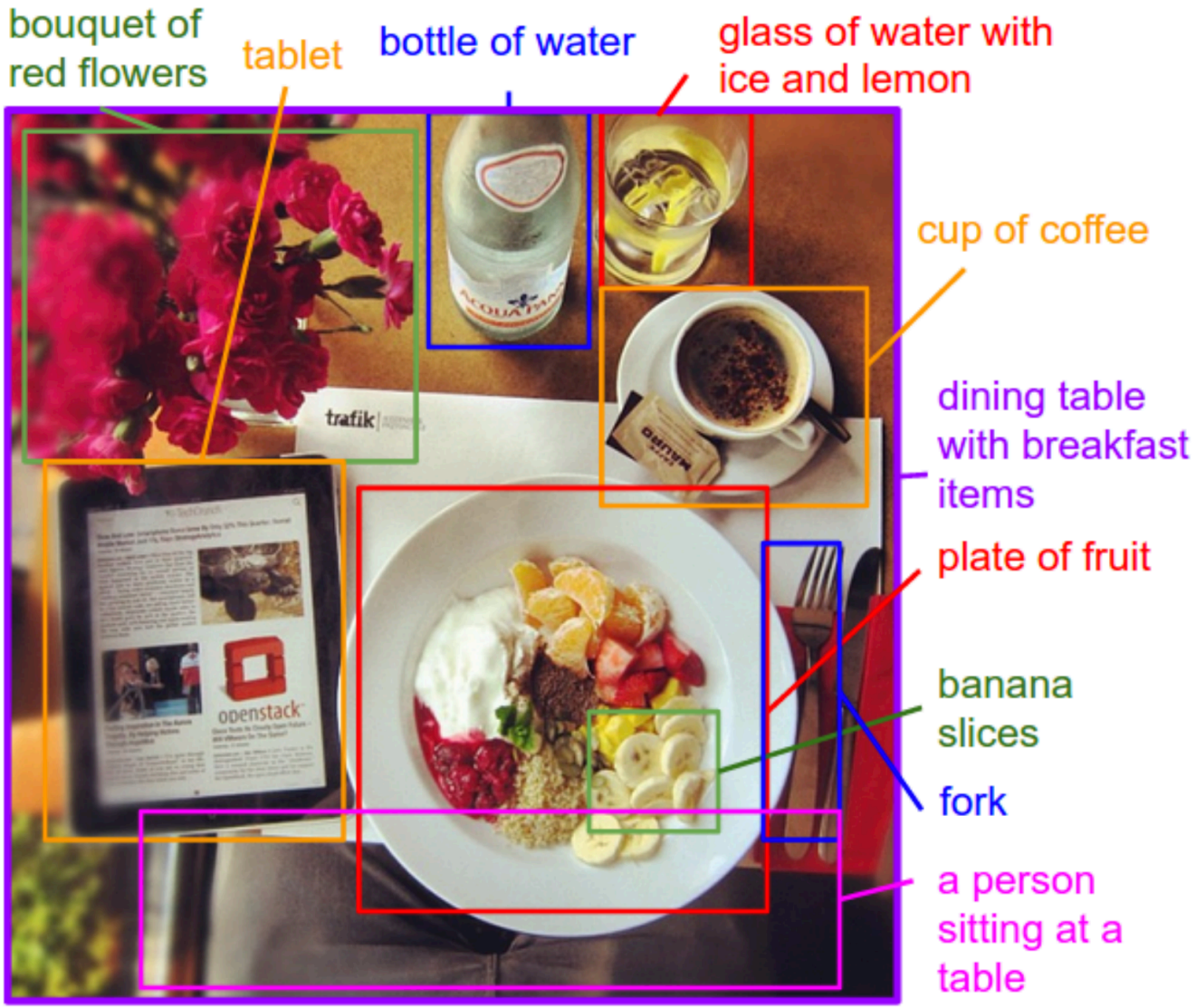


Figure 1. Motivation/Concept Figure: Our model treats language as a rich label space and generates descriptions of image regions.

This model

Two parts:

- We develop a deep neural network model that infers the latent alignment between segments of sentences and the region of the image that they describe. Our model associates the two modalities through a common, multimodal embedding space and a structured objective. We validate the effectiveness of this approach on image-sentence retrieval experiments in which we surpass the state-of-the-art.
- We introduce a multimodal Recurrent Neural Network architecture that takes an input image and generates its description in text. Our experiments show that the generated sentences significantly outperform retrieval-based baselines, and produce sensible qualitative predictions. We then train the model on the inferred correspondences and evaluate its performance on a new dataset of region-level annotations.

Model

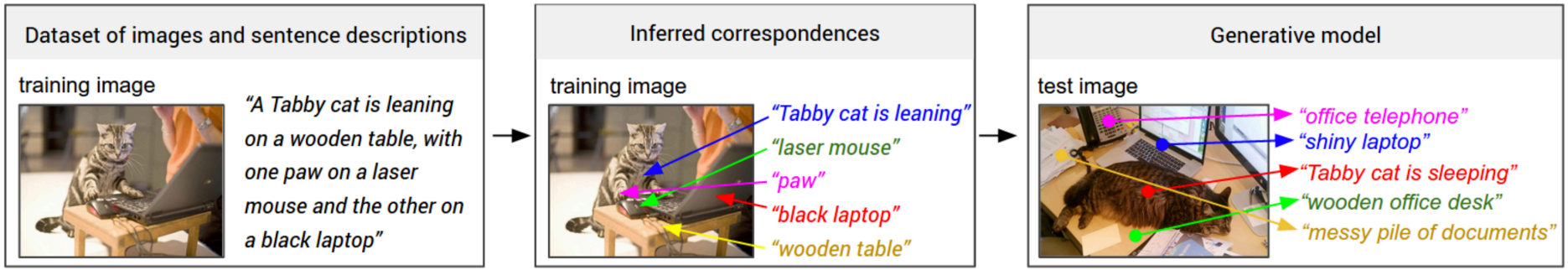
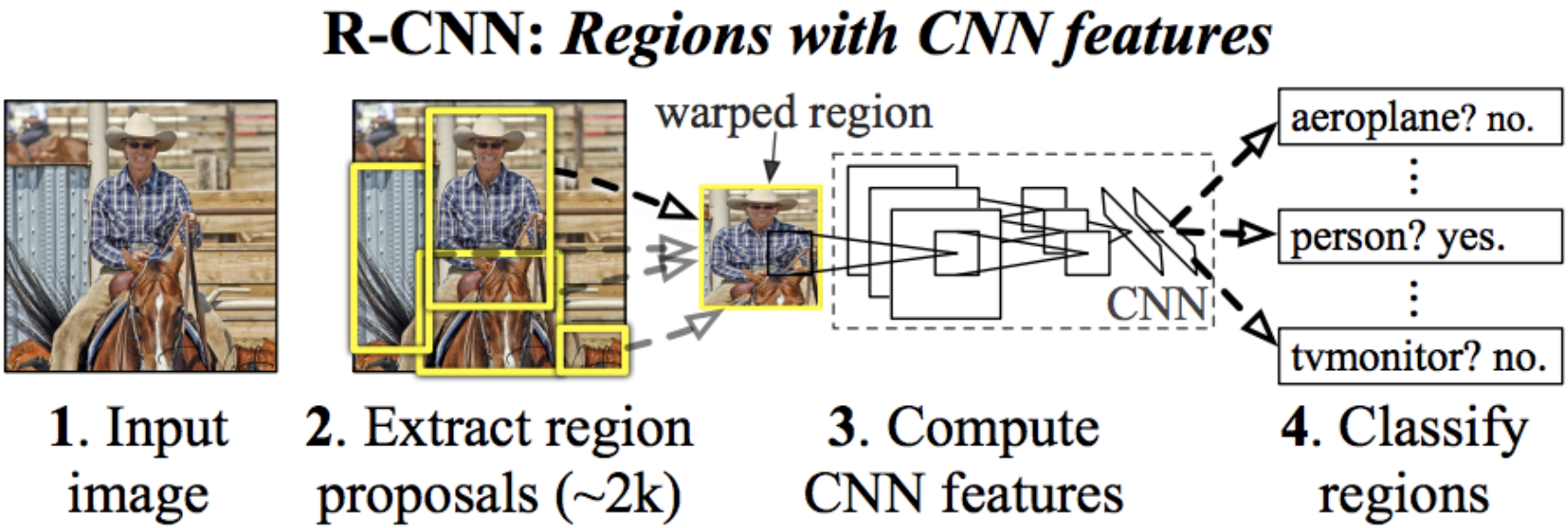


Figure 2. Overview of our approach. A dataset of images and their sentence descriptions is the input to our model (left). Our model first infers the correspondences (middle, Section 3.1) and then learns to generate novel descriptions (right, Section 3.2).

R-CNN

Use an algorithm to predict regions that may have objects.



Squish and stretch these regions into the correct input size for the CNN and run object classification on them.

Model – Each Circle is a Dense Layer

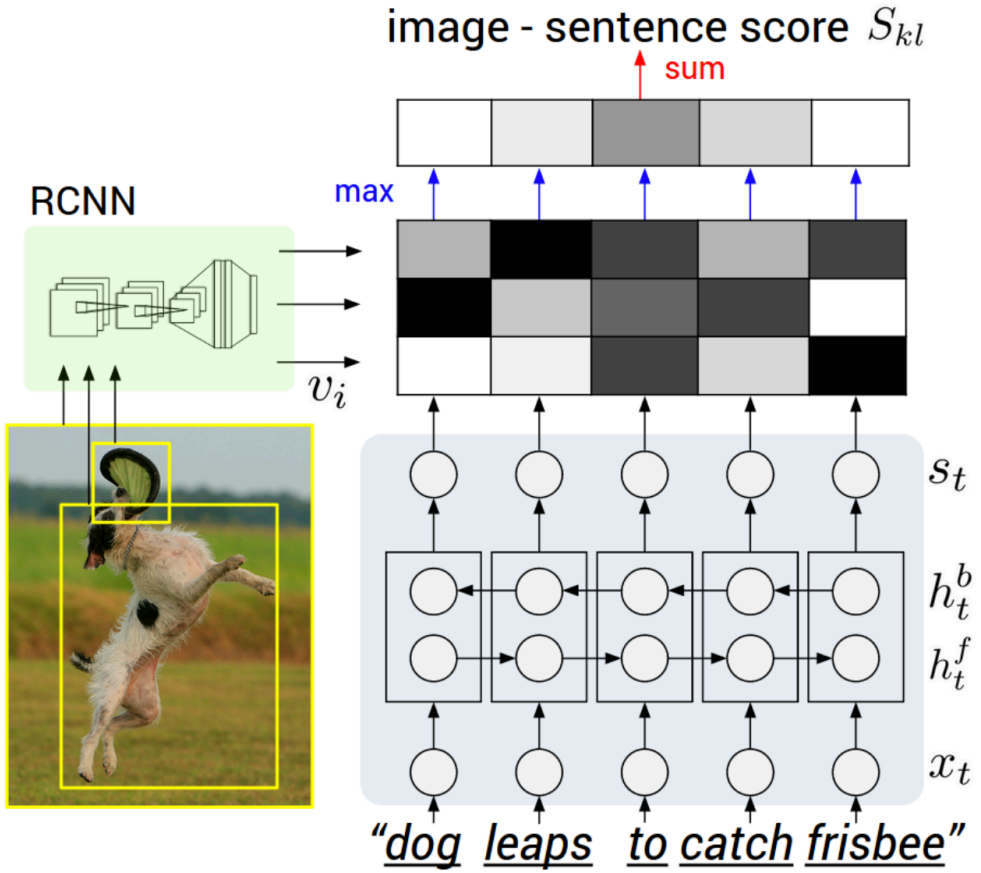


Figure 3. Diagram for evaluating the image-sentence score S_{kl}. Object regions are embedded with a CNN (left). Words (enriched by their context) are embedded in the same multimodal space with a BRNN (right). Pairwise similarities are computed with inner products (magnitudes shown in grayscale) and finally reduced to image-sentence score with Equation 8.

Computing Alignments

The similarity between word s and region v is

$$\text{similarity}(s, v) := \max(0, s^T v)$$

The authors then compute the similarity between a sentence l with word set g_l and an image k with region set g_k as

$$S_{lk} := \sum_{s \in g_l} \sum_{v \in g_k} \text{similarity(s, v)}$$

Finally, the authors found that a simpler formulation worked better:

$$S_{lk} := \sum_{s \in g_l} \max_{v \in g_k} s^T v$$

They trained the model to make $S_{kk} = 1$ and $S_{lk} = 0$ for all $l \neq k$.

Model – Similarity Computation Visualized

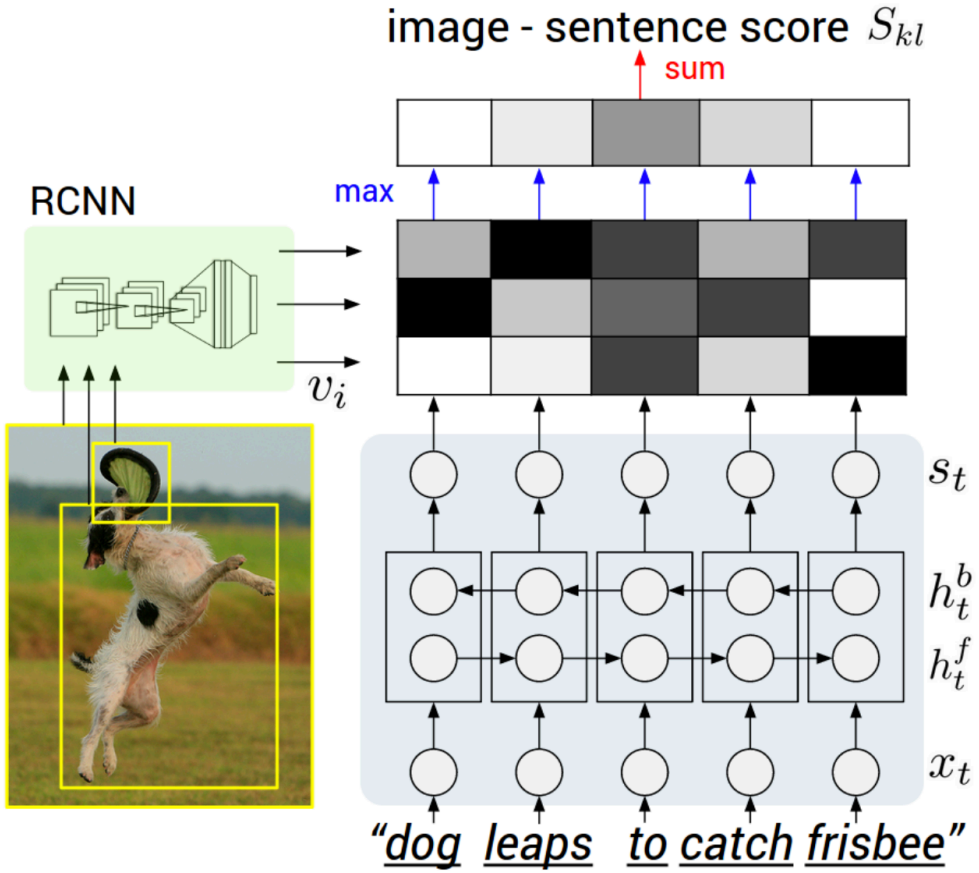


Figure 3. Diagram for evaluating the image-sentence score S_{kl}. Object regions are embedded with a CNN (left). Words (enriched by their context) are embedded in the same multimodal space with a BRNN (right). Pairwise similarities are computed with inner products (magnitudes shown in grayscale) and finally reduced to image-sentence score with Equation 8.

Example Alignments

Our alignment model produces this:

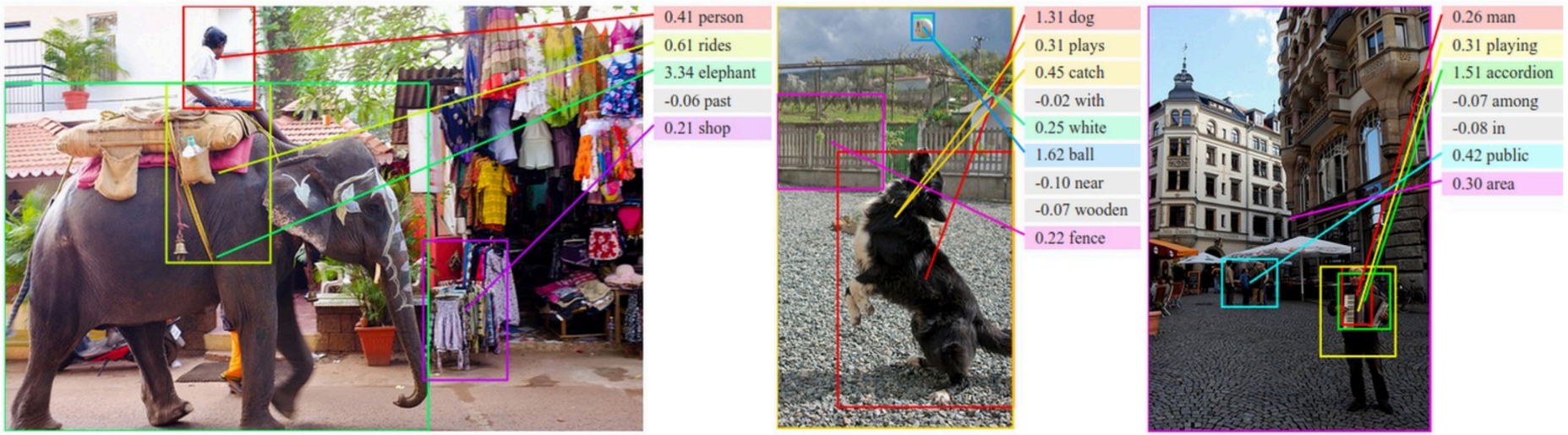


Figure 5. Example alignments predicted by our model. For every test image above, we retrieve the most compatible test sentence and visualize the highest-scoring region for each word (before MRF smoothing described in Section 3.1.4) and the associated scores ($v_i^T s_t$). We hide the alignments of low-scoring words to reduce clutter. We assign each region an arbitrary color.

Each word is primarily associated with one bounding box. But we want snippets associated with bounding boxes.

Example Regions

We want each region to be associated with a sentence fragment:

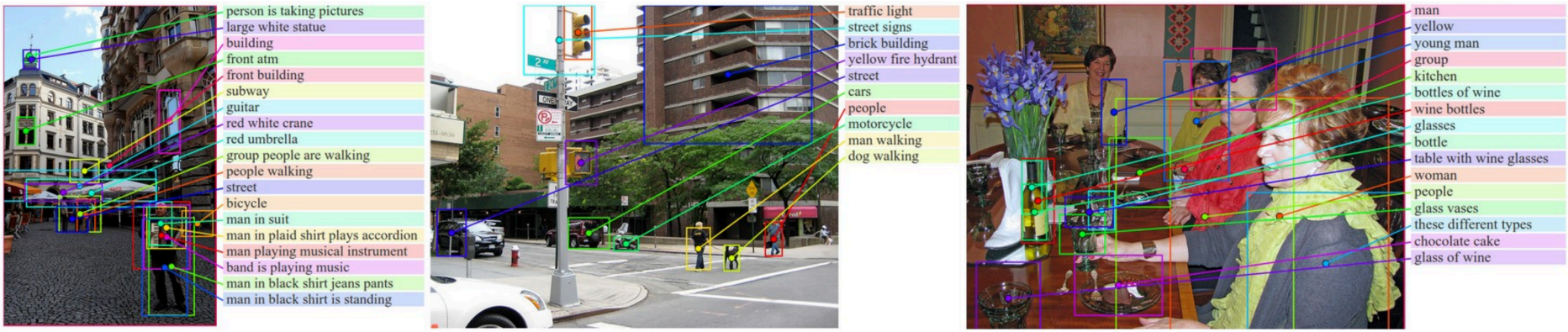


Figure 7. Example region predictions. We use our region-level multimodal RNN to generate text (shown on the right of each image) for some of the bounding boxes in each image. The lines are grounded to centers of bounding boxes and the colors are chosen arbitrarily.

Snippet alignments

Find a set of alignments that prefer to align adjacent words to the same bounding box. a_j is an alignment for word j, and it takes of values of $1, 2, \ldots M$ where there are M bounding boxes.

$$\psi_j^U(a_j = t) = v_j^T s_t$$

$$\psi_j^B(a_j, a_{j+1}) = \beta \mathbb{1}[a_j = a_{j+1}]$$

$$E(a) = \sum_{j=1..N} \psi_j^U(a_j) + \sum_{j=1..N-1} \psi_j^B(a_j, a_{j+1})$$

What the heck is β?

β is treated as a hyperparameter here. Higher values of β gives a larger reward for assigning adjacent words to the same bounding box.

Learning snipped alignments

This is why some people think this paper is sketchy.

Maximize $E(a)$ using DP (!) to select the optimal a_j for each $j = 1..N$.

Error in the paper?

The paper mentions that the energy is minimized, but I think they mean maximized.

This is not really a 'machine learning' approach although it's a statistical model (Markov Random Field) paramaterized by learned weights.

The authors don't really give a justification for why this isn't done with ML, although there are other Markov-like DP problems done in NLP (see beam search or CRFs for decoding).

Caption Generation

We can now take a set of regions and a caption and find the regions that best describe snippets of the caption. Now we would like to generate the caption ourselves.

The authors use a RNN language model to predict the caption for an image using a CNN encoding to initialize the RNN

RNN Choice

One notable thing is that both this and the previous RNN are vanilla RNNs and not LSTM/GRUs!

$$b_v = W_{hi}[\text{CNN}_{\theta_c}(I)]$$

$$h_t = relu(W_{hx}x_t + W_{hh}h_{t-1} + b_h + \mathbb{1}(t=1)b_v)$$

$$y_t = softmax(W_{oh}h_t + b_o)$$

Model

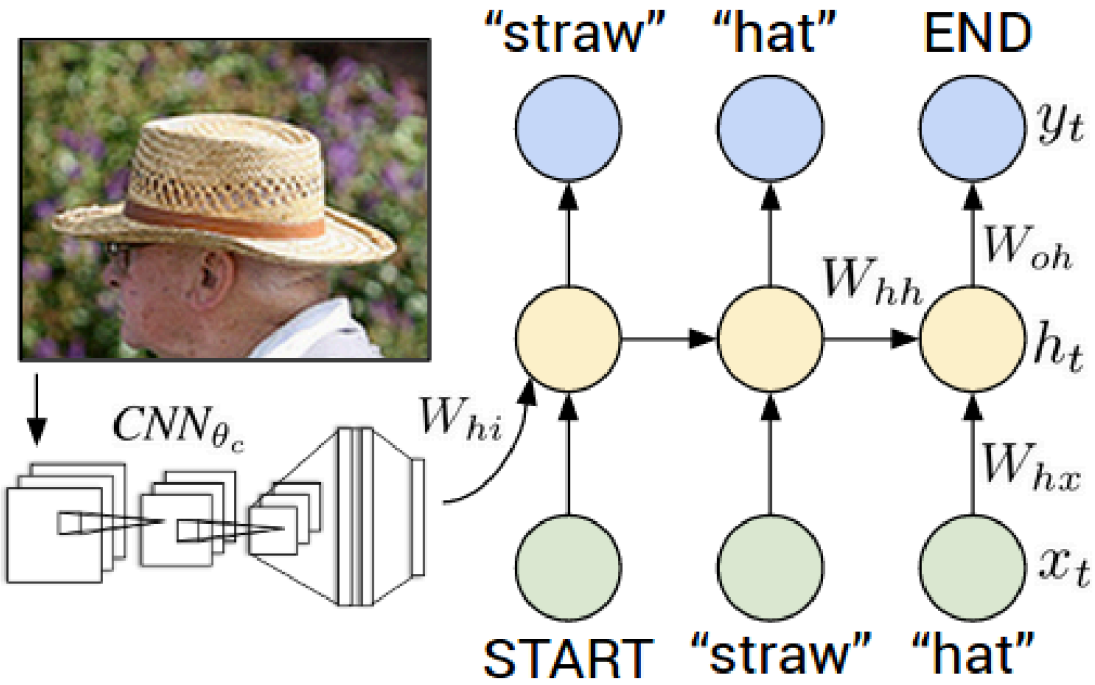


Figure 4. Diagram of our multimodal Recurrent Neural Network generative model. The RNN takes a word, the context from previous time steps and defines a distribution over the next word in the sentence. The RNN is conditioned on the image information at the first time step. START and END are special tokens.

Training

We use SGD with mini-batches of 100 image-sentence pairs and momentum of 0.9 to optimize the alignment model. We cross-validate the learning rate and the weight decay. We also use dropout regularization in all layers except in the recurrent layers and clip gradients elementwise at 5 (important). The generative RNN is more difficult to optimize, party due to the word frequency disparity between rare words and common words (e.g. ” a” or the END token). We achieved the best results using RMSprop, which is an adaptive step size method that scales the update of each weight by a running average of its gradient norm.

Momentum

When we perform gradient descent, we can often get stuck in small plateaus or local minima.

Instead of performing a standard gradient update, we take a weighted sum of our current gradient and our previous update, called our 'momentum'.

The idea is that this momentum will carry us over plateaus or small local minima to find a much better local minimum.

Gradient Clipping

Many RNNs have cliff-like behaviour in their loss functions.

Taking large steps with SGD near these cliffs can cause bad convergence behaviours.

It's also just a good idea in general to limit your step size.

Gradient clipping reduces the magnitude of the loss gradient if it is above a certain threshold.

Dropout

Randomly zero-out some of the nodes in our neural network, changing the nodes we zero-out on each training iteration.

Dropout at Evaluation Time

Remember not to apply dropout when you are evaluating your network. Most deep learning frameworks will do this for you

This gives a powerful regularizing effect.

The motivation for dropout is that it is like ensembling many smaller networks, and ensembling is Really Good ™.

Cross-Validation

Earlier we talked about having a training set and a dev set. This is one way of validating a choice for hyperparameter, and is called 'holdout cross-validation'.

k-fold CV

Another way of doing cross-validation is k-fold CV, where your dataset is partitioned into k folds. The model is trained k times, once on each subset of $k-1$ folds, and the remaining fold is used to validate the model. The results are averaged over the k runs.

Cross-validation is a way of partitioning your training data in order to validate your model. Here the authors are using cross-validation to pick the best learning weight and weight decay for their optimizer.

Section 4

Section 4 of the paper details experiments they did to evaluate their model as well as try to extend it.

I don't really like this part of the paper but you can read it if you wish.

Let's just look at some more pretty pictures and call it a day.

But first, they mention how they got their training data...

Datasets: We use the Flickr8K, Flickr30K and MSCOCO datasets in our experiments. These datasets contain 8,000, 31,000 and 123,000 images respectively and each is annotated with 5 sentences using Amazon Mechanical Turk. For Flickr8K and Flickr30K, we use 1,000 images for validation, 1,000 for testing and the rest for training. For MSCOCO we use 5,000 images for both validation and testing.

Data Preprocessing: We convert all sentences to lowercase, discard non-alphanumeric characters. We filter words to those that occur at least 5 times in the training set, which results in 2538, 7414, and 8791 words for Flickr8k, Flickr30K, and MSCOCO datasets respectively.

Amazon Mechanical Turk

Mechanical Turk: Humans that you pay to label data for you.

Often used to generate training data or evaluate the output of your model to make sure it's sane.

They can answer yes/no questions, draw bounding boxes, object detection, facial landmark detection, etc.

The other big one is FigureEight/Crowdflower but Google has their own offering as well for computer vision data.

More Examples

Figure 11. Additional examples of alignments. For each query test image above we retrieve the most compatible sentence from the test set and show the alignments.

More Examples

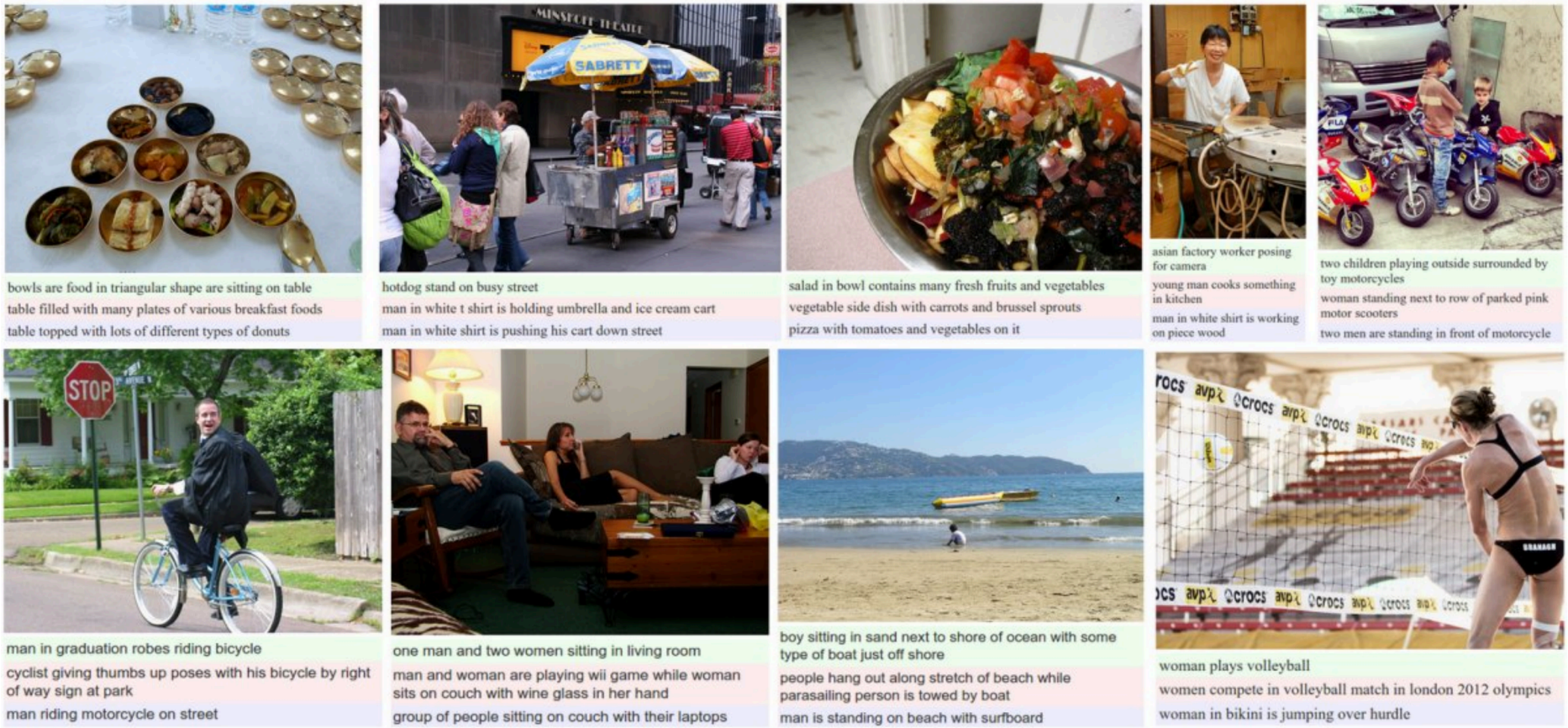


Figure 12. Additional examples of captions on the level of full images. Green: Human ground truth. Red: Top-scoring sentence from training set. Blue: Generated sentence.

Homework

Practice: Read the BERT paper. `https://arxiv.org/abs/1810.04805`

Exercises:

- Familiarize yourself with CNNs
- Use the TensorflowHub version of ELMO to create a NLP model for a task of your choice, compare it with other embeddings
- Implement ELMO
- Try to reproduce the results from *Neural Machine Translation by Jointly Learning to Align and Translate*, either on the paper's dataset or try generalizing the results to your own dataset