A Corpus of Natural Language for Visual Reasoning (Supplementary Material)

Alane Suhr[†], **Mike Lewis**[‡], **James Yeh**[†], **and Yoav Artzi**[†]

[†] Dept. of Computer Science and Cornell Tech, Cornell University, New York, NY 10044
{suhr, yoav}@cs.cornell.edu, jamesclyeh@gmail.com

[‡] Facebook AI Research, Menlo Park, CA 94025
mikelewis@fb.com

1 Examples

Table 1 shows ten labeled examples of sentence-image pairs from our dataset. Figure 1 shows an example of the JSON structured representation.

```
[[{"y_loc": 80, "color": "Black",
"type": "square", "x_loc": 40,
"size": 20}, {"y_loc": 59, "color":
"Yellow", "type": "square",
"x_loc": 40, "size": 20}, {"y_loc":
38, "color": "#0099ff", "type":
"square", "x_loc": 40, "size": 20}],
[{"y_loc": 80, "color": "#0099ff",
"type": "square", "x_loc": 40,
"size": 20 }, {"y_loc": 59, "color":
"Yellow", "type": "square", "x_loc":
40, "size": 20 }, {"y_loc": 38,
"color": "Yellow", "type": "square",
"x_loc": 40, "size": 20 }],
[{"y_loc": 80, "color": "#0099ff",
"type": "square", "x_loc": 40,
"size": 20},{"y_loc": 59, "color":
"Yellow", "type": "square", "x_loc":
40, "size": 20}, {"y_loc": 38,
"color": "Yellow", "type": "square",
"x_loc": 40, "size": 20 }]]
```

Figure 1: The JSON structured representation for the last example in Table 1.

2 Features

In the systems that use the structured representations instead of the images, we use two types of features.

2.1 Image and Sentence Features

Each feature in the MaxEnt and MLP models is a conjunction between a property of the structured representation and a property of the sentence. When using n-grams, we use $2 \leq n \leq 6$.

Property-based Features Given a sentence-representation pair, for each property listed in Table 2, we compute if it holds for the representation. For each property that holds and for each n-gram in the sentence we trigger a feature. Consider the first example in Table 1. The features triggered for this example include $\mathrm{touches\text{-}wall\#two\text{-}boxes\text{-}have}$ and $\mathrm{touches\text{-}wall\#touching\text{-}the\text{-}side}$ computed from the property $\mathrm{touches\text{-}wall}$ and the tri-grams *two boxes have* and *touching the side*. We observe that the MaxEnt model learns a higher weight for features which combine similar properties of the world and the sentence, such as $\mathrm{touches\text{-}wall\#touching\text{-}the\text{-}side}$.

Count-based Features We identify all n-grams in the sentence containing a numerical word type. Numerical word types include integers (e.g. *7*, *12*, *zero*, *two*), *no*, and *a*. We identify all counts listed in Table 3 of the structured representation. For each numerical n-gram and count, we trigger a feature based on the relation between the numbers. We only trigger features when the number in the count is greater-than-or-equal or equal to the number from the numerical n-gram. In the features, we use a generalized n-gram, where we replace the number word with a placeholder. The feature name uses the generalized n-gram and the relation. For example, consider the fourth example in Table 1. There is a box that contains exactly four objects, and the n-gram *a four block tower* appears in the sentence. Therefore, we trigger the feature $\mathrm{in\text{-}box\text{-}count} = \mathrm{a\text{-}\_\text{-}block\text{-}tower}$. We also trigger the feature $\mathrm{in\text{-}box\text{-}count} \geq \mathrm{a\text{-}\_\text{-}block\text{-}tower}$, because the count of 4 is greater than or equal to the number word *four*.

Ablation Table 4 shows ablation development accuracies. We ablate (a) the size of n-grams used, (b) number relations, and (c) number templates.

2.2 Image-only Features

In the Features+RNN model, we compute features from the structured representation independently

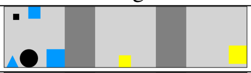
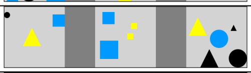
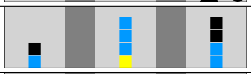
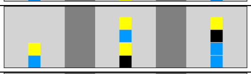
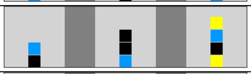
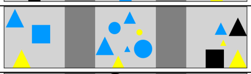
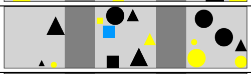
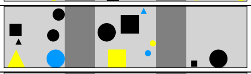
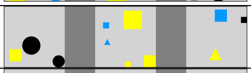
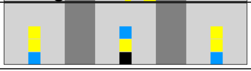
Image	Sentence	Label
	*t least two boxes have the same number of objects each and the same number of object touching the side.*	true
	*There is a box with only blue and yellow items of which there are only yellow squares.*	true
	*there is at least one tower with four blocks with a yellow block at the base and a blue block below the top block*	true
	*There is a four block tower where the base and second blocks are the same in color.*	true
	*There is a blue block as the base of a tower with at most three blocks.*	true
	*There is a box with all 3 different colors and a black triangle touching the wall with its top.*	false
	*There is a box with seven items and the three black items are the same in shape.*	false
	*There is a box with multiple items and only one item has a different color.*	false
	*None of the yellow objects are touching the edge*	false
	*There is exactly one tower with a blue block at the base and yellow block at the top*	false

Table 1: Examples from the Cornell Natural Language Visual Reasoning (NLVR) corpus.[1]

Property name	Description
Top colors	For each of the three colors, whether the topmost object(s) in a box is this color.
Bottom colors	For each of the three colors, whether the lowest object(s) in a box is this color.
Touching	Whether any object is touching any walls in any box.

Table 2: Description of properties used to generate property-based features.

of the sentence. The features are then combined with the RNN sentence representation.

For each object in the structured representation, we construct a set of features using the following properties: color, shape, size, and touching the left, right, top, or bottom walls. We process the objects in order and generate a unique feature for every property and object index. We also compare the location of each object to all other objects in the same box and create a feature for each other object depending if it is below, above, left, or right to the object. The object's representation is its set of property features. We use the object ordering as it appears in the JSON string. The number in each box is bounded and there are three boxes. Therefore, the maximum number of features is known, and we can represent each feature as a one-hot vector.

[1]Any typos in the sentences shown are directly from the data.

3 Implementation details

In all models except MaxEnt and NMN, we use a patience stopping procedure. We start with an initial patience counter value of 5, and test on the development set after every epoch. If the accuracy improves, we reset the counter to the value it was last assigned upon improvement multiplied by 1.1. We decrease the counter at each epoch, and terminate when the counter is below zero. For all neural network models, except NMN, we tune the hyperparameters on the development set, including tuning the size and number of hidden layers, the size of embeddings, and learning rate.

MaxEnt We use Megam[2] and run learning for up to 100 iterations.

MLP We use 32-dimensional feature embeddings, and a hidden layer of size 32. We initialize all learned parameters uniformly in $[-0.01, 0.01]$. We use ADAM with an initial learning rate of 0.01 and a mini-batch size of 128.

CNN+RNN We use an image size of 20×80, a CNN with three layers, each of size 32 with filter sizes of 8×8, 4×4, and 2×2 and strides of 4, 2, and 2. The CNN output is processed with a fully connected layer with 256 units. We use 16 dimensions for word embeddings, and 128 LSTM units

[2]https://www.umiacs.umd.edu/~hal/megam/

Count name	Description
Color-shape combinations	Count of the number of objects with a certain color and shape in the entire image (nine possible counts, from three colors and three shapes).
Color-shape combinations in a box	For each box, count of the number of objects with a certain color and shape (27 possible counts, from nine possible color/shape combinations, and three boxes to count in).
All one color	For each color, count of the number of boxes in the image which contain entirely objects of that color only (three possible counts; one for each color).
Objects in box	For each box, count of the number of items in the box (three possible counts; one for each box).
Color-shape combination touching the wall	Count of the number of objects with a certain color and shape combination in the image which are touching any wall (nine possible counts, from three colors and three shapes).
Color-shape combination touching the wall in a box	For each box, count of the number of objects with a certain color and shape that are touching the wall (27 possible counts, from nine possible color/shape combinations, and three boxes to count in).
Total number touching	Count of the number of objects touching any wall in the image.

Table 3: Description of counts used to generate count-based features.

Feature Configuration	Accuracy
n-gram length	
$2 \leq n \leq 5$	65.82
$2 \leq n \leq 4$	65.22
$2 \leq n \leq 3$	63.20
Counting Relations	
w/o greater-than-or-equal relation	67.13
w/o equality relation	60.87
w/o count-based features	57.53
w/o property-based features	66.45

Table 4: Development accuracies for ablation of image and sentence features (Section 2.1).

in the RNN. We concatenate the sentence embedding with the image embedding and pass through a fully connected layer of size 128 with ReLu non-linearity. We then perform a linear-transformation and a softmax. We initialize learned parameters uniformly in $[-0.01, 0.01]$. We use ADAM with an initial learning rate of 0.001 and a mini-batch size of 128.

Image Features+RNN We use the same RNN architecture as in CNN+RNN. We use the concatenation of the one-hot image features as described in Section 2.2 to compute the image embeddings with two layers of size 32. We concatenate the image and sentence embeddings, pass through a fully connected layer of size 32, and then apply a softmax. We initialize learned parameters uniformly in $[-0.01, 0.01]$. We use ADAM with an initial learning rate of 0.0075 and a mini-batch size of 128.

NMN We use the public implementation.[3] We tune the maximum leaves parameter to 5. We also preprocess our sentences using several simple rules (e.g. transpose “there are” to “are there”) to construct questions out of our statements. We run learning for 100 epochs, and observe the performance on all sets at the epoch with the highest development performance.

Text and Image Only We use the RNN and CNN architectures described above, using the same hyperparameter settings.

[3]https://github.com/jacobandreas/nmn2