A Supplemental Material

In this section, we provide implementation details for our models. The dataset we use has 2,493 training, 360 development, and 720 test examples.

A.1 Advice Sentence Generation

Our advice sentences are generated by filling in appropriate regions/directions into varying sentences. For example, given an advice sentence placeholder, *The target is in the* _____, and a coordinate in the lower left, we would generate the restrictive advice sentence: *The target is in the lower left.* At test time, we use variations of this sentence such as: *The block's region is the lower left*, to avoid memorization. We showed in Table 1 the importance of our pre-trained models in enabling sentence variability and advice understanding (`M4` vs `M5`).

A.2 Pre-trained Model Details

All of our pre-trained models are trained on random coordinates and advice sentences.

The model used to comprehend restrictive advice is trained as a binary prediction problem, and must output a positive prediction if the random input coordinate is in the region described by the input advice sentence. This design allows the model to understand the meaning of the advice sentence by determining if the random coordinate follows the sentence.

Our architecture takes as input an advice sentence $s = w_1, w_2, ..., w_n$, passes it through a trained embedding layer of size 100, a LSTM of size 256, and outputs the hidden state representations $\{h_n\}$. The last hidden state h_n is embedded using a Fully Connected (FC) layer of size 100. Each axis of a random input coordinate (x, y, z) is also passed into the network and embedded using a FC layer of size 100. These 4 FC layers are summed up and passed through a final FC layer O of size 2, which is then followed by a softmax. All FC layers use the RELU activation function. We train this as a binary prediction problem using cross-entropy loss, the Adam optimizer, and a learning rate of 0.001. Gradient clipping (Pascanu et al., 2013) is used on the LSTM parameters to avoid exploding gradients.

The model for corrective advice is identical to the one for restrictive advice, except the final FC layer O has size 3, and the model is trained to output a coordinate that follows the advice. If it outputs an advice-following coordinate, the model receives 0 loss. Otherwise a mean square regression loss is provided, where the ground truth is some random coordinate that does follow the advice.

A.3 End-to-end Advice Model Details

The end-to-end model is trained and tested on the training and test split from (Bisk et al., 2016). We load and freeze the LSTM, embedding layer, LSTM hidden state h_n and the FC layer following it from the pre-trained model into the end-to-end model. This last FC layer is passed through a FC layer of size 256, which is then summed with the LSTM hidden state of the original Bisk et al. model. We are able to accurately generate and feed-in the advice to the pre-trained portion of this model (just like a human would), as we have access to the true source/target coordinates from the dataset. The rest of the architecture and training procedure is identical to Bisk et al..

A.4 Model Advice Generation Details

For the model that self-generates the advice (Figure 3), we use a neural network model, passing the instruction from the Bisk et al. dataset into an embedding layer of size 256, followed by a LSTM of size 256. We then embed the blocks-world grid using a FC layer of size 20 (the maximum number of blocks there are), final FC layer of size 4 (when dividing the grid into 4 regions), and softmax with cross-entropy loss. We train using Adam optimizer (Kingma and Ba, 2014) and a learning rate of 0.0001. All FC layers use the Leaky RELU activation function (Maas et al., 2013). We note that if we change the final FC layer to size 3 and train this model as a coordinate prediction problem, the performance is worse than Bisk et al.. This shows that the overall performance improvement shown in Section 3.4 is due to considering self-generated advice.

When using the model self-generated advice in the end-to-end model, we use accurate advice at training time (computed using the train split of Bisk et al.), and the self-generated advice at test time. The self-generated test advice is generated by training the advice generation model on the Bisk et al. dataset, and using the best performing model (evaluated on the dev split) to generate advice for the test data. Thus, no human interaction is needed.

When using retry advice, we also use accurate advice at training time and self-generated advice

at test time. However, we generate two sets of self-generated advice, one for the most confident region prediction, and another for the next most confident one (determined by the softmax scores and explained in Section 3.3). If the general region of the coordinate prediction in the first iteration of running the end-to-end model (with the most confident self-generated advice) is incorrect, the human operator will provide retry advice, and we then feed in the second most confident advice (using that for the final prediction).

Input-specific self-generated advice is generated in two iterations. In the first iteration, we run the trained Bisk et al. model on the test split, and then create an advice region (of the same size as when we divided the board into four quadrants) centered at the predicted coordinate (but not exceeding the boundary of the board). In the second iteration, this advice is fed into the Bisk et al. model just like Section 3.1. Thus, again, no human interaction is needed.