Virtual World Context Encoding for Grounded Dialogue in Minecraft

Charlotte Lambert[1], Ariel Cordes[2], Elli Kaplan[3], Prashant Jayannavar[4], and Julia Hockenmaier[4]

[1]Vassar College
[2]University of Minnesota Morris
[3]North Carolina State University
[4]University of Illinois Urbana-Champaign
clambert@vassar.edu

Abstract

Our research aims to develop agents that can communicate with each other using natural language (e.g. to give and execute instructions) in a 3D environment. We use Minecraft, a game in which players can explore a 3D environment and build structures out of blocks. In the Minecraft Collaborative Building Task, an Architect (**A**) has to instruct a Builder (**B**) via chat to build a copy of a Target structure given to **A**. Throughout the game, **A** and **B** communicate back and forth via chat, but while **A** can observe **B**, only **B** can place blocks.

We designed and implemented a convolutional neural network (CNN) to recognize shapes in Minecraft's virtual 3D environment. We have begun to use the output of this CNN as input to models that generate utterances for the Architect or Builder agents, and will be incorporating it into models that predict the Builder's next block placement actions.

1 Introduction

The concept of a blocks world opens up the possibility of creating agents that effectively communicate about those blocks in a given environment. Accomplishing this task requires an ability to use relevant language and an understanding of the structures in the world. In this paper, we build upon work done by Narayan-Chen et al. (2019) on the Minecraft Collaborative Building task and use data from the Minecraft Dialogue Corpus (Narayan-Chen et al., 2019). We develop a convolutional neural network to provide the Architect and Builder agents with more context about the shapes present in the 3D environment (Section 3). Additionally, we work with the sub-task of Builder utterance generation and create the first two iterations of a Builder model (Sections 4 and 5). We continue to work with the best Architect agent developed by Narayan-Chen et al. (2019) and encode a new representation of the target structure and the current build structure (Section 6). In Section 7 we describe our experimental setup and analyze the results in Section 8.

2 Related Work

Our work is similar to that of Wang et al. (2017) in that it involves building structures in a three-dimensional blocks world, however, their work is more focused on the use of a programming language. Additionally, Wang et al. (2017) use human participants to dynamically teach their system more language. This differs from our work because we are developing fully interactive agents, and thus won't have our system communicate with human participants. The work of Bisk et al. (2016a,b, 2018) is more closely related to ours since it has a similar need for a system with some complex understanding of certain structures. Bisk et al. (2016a,b, 2018) use the example of a tower, a shape we frequently see in our target structures, to demonstrate concepts their system needs to understand. In both their and our work, we hope to develop systems with an understanding of these simplistic shapes in a blocks world environment. However, our work requires two agents (or two humans) to communicate unlike the work of Bisk et al. (2016a,b, 2018). In addition, our use of Minecraft as the environment for our blocks world gives us more freedom to build more varying types of structures, to manipulate the viewing angle, and to identify sub-shapes of structures based on color. The distinct colors are particularly important seeing as they motivate much of the progress reflected in this paper.

3 Shape Recognition Task

To give both the Architect and Builder agents more context during the game, we develop a convolutional neural net (CNN) to recognize shapes in 3D space. The CNN is designed to recognize the shapes most frequently found in target structures and referenced by Architects in their instructions. There are seven shapes the CNN can recognize: rows, diagonal lines, T-shapes, U-shapes, L-shapes, rectangles, and rectangular prisms. Each of these shapes has various different orientations within 3D space. Summing up all the unique orientations of each shape yields 18 different possible structures to be recognized. These structures are represented in a synthetic coordinate system with the same dimensions as the 3D space used in the Minecraft Collaborative Building Task developed by Narayan-Chen et al. (2019).

The Architecture of the CNN consists of two convolutional layers and three linear layers. The CNN uses ReLU as a nonlinearity. There are six input channels (one for each color channel of the world state) and 18 outputs (one for each type of shape). This is a multilabel task, so the output will reflect every shape recognized within the input world state. The CNN will recognize each shape in world states containing multiple independent structures and all sub-shapes of a given shape (e.g., rows are sub-shapes of rectangles).

4 Seq2Seq Builder Utterance Model

We develop two basic models for Builder Utterance generation. First, we utilize the existing sequence-to-sequence model used for the Architect previously (Narayan-Chen et al., 2019). Similarly, this model relies the dialogue history to determine what utterance should be generated next.

This dialogue history is encoded as a sequence of tokens. Builder utterances are written between start token $< B >$ and end token $< /B >$. All tokens between the start and end tokens comprise an utterance, or one complete message sent from the Builder to the Architect (Narayan-Chen et al., 2019). Just as with the existing Architect model, the tokens go through a word embedding layer and a bidirectional RNN (Schuster and Paliwal, 1997).

5 Utterances and Shape Builder Utterance Model

Since the Builder, unlike the Architect, does not have access to the target structure, we do not augment our model with a comparison of the two structures. Instead, for our second Builder model, we augment the seq2seq model with the output of our shape recognition CNN evaluated on the built structure to provide the Builder with more context. The encoder continues to utilize the dialogue history along with the output of the CNN to predict the next Builder utterance to be generated.

6 Seq2Seq Architect Utterance Model

We also augment the basic seq2seq Architect utterance model with the output of our shape recognition CNN. We evaluate both the target structure and the built structure using the CNN and compute the explicit difference between them (target minus built). This difference reflects any shapes present in the target structure but not the built structure, thus indicating which shapes are left to be built. All three of these values are used alongside the dialogue history to improve the Architect utterance generation by providing as much context as possible.

6.1 Block Counters

Along with the addition of the output from the CNN, we augment the Architect's seq2seq model with global block counters, as was done in models developed by Narayan-Chen et al. (2019). This gives the Architect more information about expected overall placements, next placements, and removals for all six color channels over the entire build region.

7 Experimental Setup

7.1 Shape Recognition CNN

Data We synthetically generated 104,600 random 3D configurations composed of at least one of 7 possible shapes. The training, test, and validation sets contain 66,666, 4,600, and 33,334 representations of 3D world states respectively. Half of each data set is composed of simple shapes (i.e., one shape in the world) and the remaining data is composed of composite shapes (i.e., between two and four shapes in one world).

Training We trained for a maximum of

	Shape						
Metric	Row	Plane	Rectangular Prism	Diagonal Line	L-shape	U-shape	T-shape
Precision	100	100	100	99	100	98	99
Recall	100	100	100	98	100	99	100

Table 1: Precision and accuracy of shape recognizing CNN per shape

150 epochs using the SGD optimizer. We ended training early when validation loss increased for 6 epochs. The loss was calculated using MultiLabelSoftMarginLoss. We varied the number of channels in the CNN's two convolutional layers and the size of the remaining linear layers until we reached the model that performed best. At that point we varied the sizes of the training, validation, and test sets (since our data is generated synthetically) to improve accuracy.

7.2 Builder Utterance Model

Data We used the utterance data from the Minecraft Dialogue corpus (Narayan-Chen et al., 2019). The training, test, and validation data contain 2,394, 986, and 870 Builder utterances respectively.

Training We followed the same setup used in previous Architect experiments (Narayan-Chen et al., 2019) and trained for a maximum of 40 epochs using the Adam optimizer (Kingma and Ba, 2015). As before, we ran a grid search over model Architecture hyperparameters adapted for the Builder and included new hyperparameters related to the addition of the shape recognizing CNN.

7.3 Architect Utterance Model

Data Again, we used utterance data from the Minecraft Dialogue corpus (Narayan-Chen et al., 2019). The training, test, and validation data contain 6,548, 2,855, and 2,251 Architect utterances respectively.

Training The training for the Architect utterance model followed the same structure as that of the Builder, however we performed more in-depth grid searches. We started off by using the hyperparameters from the best performing Architect model developed by Narayan-Chen et al. (2019) along with additional hyperparameters needed for our CNN. Once we found the best model, we fine-tuned by varying dropout parameters (Srivastava et al., 2014).

8 Results and Analysis

8.1 Shape Recognition

Table 1 shows the results of running our shape recognizing CNN over our validation set using the best set of parameters. While rows, planes, and rectangular prisms performed very well for most variations of parameters, this iteration proved to yield the best results for the most complicated shapes, namely diagonals and U-shapes.

8.2 Builder Dialogue Act Annotation

Similar to the Architect dialogue acts defined by Narayan-Chen et al. (2019), we evaluated all Builder utterances and define eight dialogue acts to categorize the types of utterances. Each utterance falls into exactly one category. These categories are defined as follows:

- *Greeting*: the utterance includes a welcome message or recognition of the start of the mission and occurs early in the game (*"Hello. What are we building this time?"*) (5.95% of all utterances)

- *Verification Questions*: the utterance includes a question to the Architect requesting that they confirm that the previous action(s) were correct (*"like this?"*) (23% of all utterances)

- *Clarification Questions*: the utterance requests that the Architect clarifies a given instruction or statement (*"can I put them anywhere?"*) (26.36% of all utterances)

- *Suggestions*: the utterance conveys something the Architect can do to make communication more effective (*"can you give me an instruction for a single purple block first?"*) (1.95% of all utterances)

- *Extrapolation*: the utterance shows the Builder making an educated guess as to how the build structure should be built and takes the initiative to carry out the assumption (*"I*

	BLEU			
Metric	B-1	B-2	B-3	B-4
seq2seq	0.193	0.11	0.081	0.045
+shapes	0.183	0.098	0.078	0

Table 2: Relevant BLEU scores for Builder utterance models over the validation set

	BLEU						
Metric	B-1	B-2	B-3	B-4	colors	spatial	dialogue
seq2seq + global & local	0.17	0.084	0.045	0.023	0.17	0.105	0.161
seq2seq	0.149	0.069	0.038	0.021	0.071	0.084	0.182
+shapes	0.151	0.074	0.042	0.025	0.107	0.093	0.144
+diff	0.157	0.077	0.043	0.025	0.12	0.09	0.175
+global	0.174	0.085	0.045	0.028	0.196	0.1	0.163

Table 3: Relevant BLEU scores for Architect utterance models over the validation set

think I know what you want. Let me try”) (0.3% of all utterances)

- *Display Understanding*: the utterance expresses understanding of a given instruction (*“oh I see”*) (27.1% of all utterances)
- *Materials Update*: the utterance states something about the current quantity of build materials or responds to an Architect’s inquiry about quantity of materials (*“I am out of blocks”*) (0.78% of all utterances)
- *Other/Chit-Chat*: any other statement not relevant to the completion of the task, including friendly chit-chat (*“look at this beautiful Architecture”*) (14.53% of all utterances)

8.3 Builder Utterance Generation

In table 2, we report the results of running our two different versions of the Builder utterance model. These results are measured with standard BLEU scores (Papineni et al., 2002). We discovered that adding the additional context to the Builder model did not improve accuracy. However, the Builder still performed better than the best Architect model (see table 3). This can be likely attributed to the simplicity of Builder utterances. This is demonstrated by the fact that the best Builder model, for example, generated 540 instances of the utterance *“like that?”* out of 870 total utterances, yet it resulted in high BLEU scores.

8.4 Architect Utterance Generation

Table 3 shows the results from the best Architect model and the basic seq2seq model, both developed by Narayan-Chen et al. (2019), along with the variations to the seq2seq model that we developed. On the first line, we show the best results previously developed. Next we report the results from the most basic seq2seq Architect model followed by all additions we made. With each augmentation to the seq2seq model, the BLEU-1 score increased. In particular, the addition of the global block counters drastically improved the accuracy. Similar to the previous experiments run by Narayan-Chen et al. (2019), the augmentations had large impacts on the color-BLEU score. For example, the model demonstrates some understanding of color when generating the utterance *“now a row of three orange blocks on top of that”* when instructing the Builder to build the structure pictured in Figure 1. Our best Architect model surpassed the BLEU scores of the previously best model. However, the issue of repeated words within generated Architect utterances cited

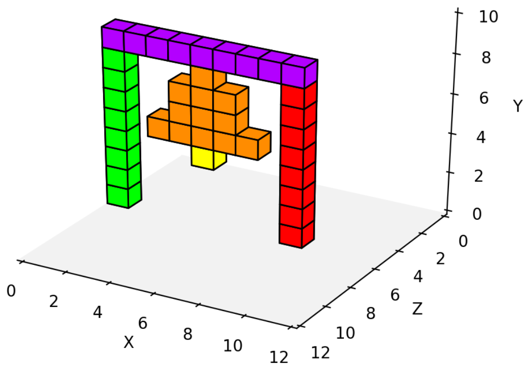

Figure 1: Example target structure

in Narayan-Chen et al. (2019) persists.

9 Conclusions and Future Work

We have expanded on the previous first steps made towards developing a fully interactive Architect agent for the Minecraft Collaborative Building Task. The work has built upon the existing Architect agents as well as introducing a simple Builder agent. With this progress, we move a step closer to creating two fully interactive agents, however many steps remain to be taken. Primarily, the Builder agent must be able to determine when to make a block placement or removal. Additionally, the Builder needs knowledge of what particular action needs to be taken based on dialogue history and the current state of the build region. Along with this new behavior, there is much room to improve Builder utterance generation. Despite obtaining the highest BLEU scores as of yet, more work should be done to fine-tune the best Builder utterance generation model to get the best performance possible. Finally, we will work towards using Builder dialogue acts to improve accuracy of Builder utterance generation in addition to creating a new metric to measure that accuracy.

Acknowledgments

We would like to thank the NSF and the DREU program through the CRA-W for this opportunity. Additionally, we would like to express gratitude to our mentors, Julia Hockenmaier and Prashant Jayannavar for their support.

References

Yonatan Bisk, Daniel Marcu, and William Wong. 2016a. Towards a dataset for human computer communication via grounded language acquisition.

Yonatan Bisk, Kevin Shih, Yejin Choi, and Daniel Marcu. 2018. Learning interpretable spatial operations in a rich 3d blocks world.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. 2016b. Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 751–761, San Diego, California. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier. 2019. Collaborative dialogue in Minecraft. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 5405–5415, Florence, Italy. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Sida I. Wang, Samuel Ginn, Percy Liang, and Christopher D. Manning. 2017. Naturalizing a programming language via interactive learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 929–938, Vancouver, Canada. Association for Computational Linguistics.