NeBuLa: A discourse aware Minecraft Builder

Anonymous EMNLP submission

Abstract

When engaging in collaborative tasks, humans efficiently exploit the semantic structure of a conversation to optimize verbal and nonverbal interactions. But in recent "language to code" or "language to action" models, this information is lacking. We show how incorporating the prior discourse and nonlinguistic context of a conversation situated in a nonlinguistic environment can improve the "language to action" component of such interactions. We fine tune an LLM to predict actions based on prior context; our model, NeBuLa, doubles the net-action F1 score over the baseline on this task of Jayannavar et al. (2020). We also investigate our model's ability to construct shapes and understand location descriptions using a synthetic dataset.

1 Introduction

High level building agents use conversation in a collaborative task to combine information about the extant conversation, the world, and prior actions to execute new instructions. Such agents interpret messy or vague language, produce actions, then reassess the situation, ask questions or take in corrections from other agents to optimize their actions. Successful collaborative conversations are vital for efficiently performing complex interactive tasks. In this paper, we study the messy language of ordinary human collaborative conversation, and how a large language model can learn to execute instructions from such conversations. We isolate several factors that affect this task.

The first factor is the interactions between linguistic and nonlinguistic contexts. Previous work has shown that at least some context is needed to understand and carry out conversationally given instructions (Jayannavar et al., 2020). We improve on that work by first establishing a baseline by using the entire exchange up to an instruction i as a context for an LLM to interpret i. Our LLM model, NeBuLa (**Ne**ural **Bu**ilder with L**la**ma), trained on the Minecraft Dialogue Corpus (MDC) (Narayan-Chen et al., 2019), achieves net-action F1 scores that is almost double of Jayannavar et al. (2020). Using the Minecraft Structured Dialogue dataset (MSDC) (Thompson et al., 2024), which provides semantic relations between MDC dialogue moves and nonlinguistic actions, we show that particular discursive components of the linguistic and nonlinguistic context are necessary and sufficient for the LLM to understand an instruction to the degree provided by the baseline.

Analysing NeBuLa's performance revealed two other factors that importantly adversely affect its performance. An instruction in the MSDC has two basic components: a description of a shape in terms of four parameters—numbers of components, colors, arrangement and orientation—and the description of a location where the shape should be placed. Human Architects often use analogies to everyday objects that may be challenging to process, and in addition shape descriptions are often underspecified, meaning that one could perform the instruction correctly in various ways. Location descriptions in the Minecraft world are also quite difficult to process and highly underspecified. For example, *put a tower in a corner* could be correctly located in any of the four corners of the Minecraft board. We address this problem in two ways: first by further finetuning NeBuLa on a synthetic dataset to improve its performance in building basic shapes and locating them appropriately; and secondly, and more importantly, by revising the evaluation metric used by Jayannavar et al. (2020) to reflect more realistically the semantics of location expressions. We show that, on our synthetic dataset, NeBuLa achieves high accuracy as per our intuitive metric in performing basic instructions.

After some preliminaries and discussion of prior work (Section 2), we present the NeBuLa model

and its baseline performance in Section 3, and then a necessary and sufficient discourse feature to get scores equivalent to the baseline in Section 4. In Section 5, we explain several issues associated with Minecraft corpus. We try to address these issues in Section 6. In this section, we explain our evaluation metric for underspecified instructions, as well as experiments on our synthetic datasets.

2 Related Work

MDC Narayan-Chen et al. (2019) introduced a corpus of two person dialogues situated in a simulated Minecraft environment. The dialogues record conversations about collaborative tasks, in which an Architect and a Builder cooperate to build sometimes complex 3-dimensional shapes out of blocks of six different colors. The Architect provides instructions, while the Builder is tasked with translating these instructions into actions. The Builder sometimes asks questions, and the Architect may correct themselves or the Builder, or both, concerning both linguistic and nonlinguistic moves. The corpus accurately reflects the variety and complexity of actual cooperative conversation. Details on the MDC are in Table 1.

Instructions to code: Neural Builder and variants The MDC (Jayannavar et al., 2020) incentivized the development of an algorithm that could predict sequences of actions from instructions. The actions involved basic moves of placing or removing blocks from certain positions in the environment. Jayannavar et al. (2020) trained a model consisting of a GRU (Cho et al., 2014) to handle textual input coupled with a CNN to integrate information from the current state and a GRU to predicted an action sequence. Although they experimented with several training regimes, the best performance came from one in which a sequence of conversational moves after some action sequence, assumed to be instructions are given to the model, are followed by the next action sequence of the Builder, followed by the next sequence of linguistic moves are input to the model to predict the subsequent action sequence. (See Figure 1).

The net-action F1 metric evaluates a model's prediction based on the exact color and coordinate match between the model's predicted sequence and Builder's gold action sequence. In general, Jayannavar et al. (2020) showed that the problem of predicting action sequences from natural language instructions in naturally occurring

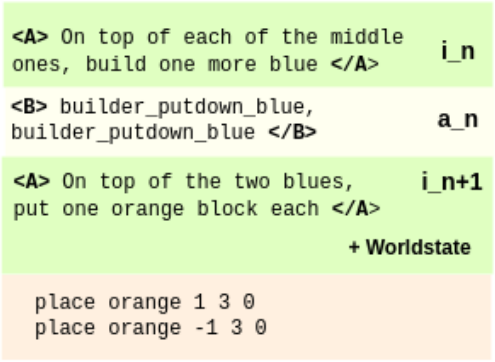

Figure 1: The Neural Builder (Jayannavar et al., 2020) takes as input the sequence $i\_n\ a\_n\ i\_n + 1$ and the world-state to predict the subsequent action sequence.

dialogue remains extremely challenging. Their Neural Builder had net action f1 of 0.20 on the MDC test set.

Shi et al. (2022) propose a somewhat different task from Jayannavar et al. (2020); they try to predict when the Builder should execute an action and when they should instead ask for a clarification question. To this end, they annotated all Builder dialogue moves with a taxonomy of dialogue acts. They then specified a *single* specific action under the execution label instead of a sequence of actions. Thus, their set-up is not directly comparable to that of Jayannavar et al. (2020).

Bonial et al. (2020, 2021) added dialogue acts to Minecraft utterances, but they did not evaluate the effect of these dialogue acts on the Neural Builder's predictions of actions. Dialogue acts are a partial step towards a full discourse structure; they provide labels for various dialogue moves, but the full discourse structure that we propose to use involves relations between moves. These relations are important as they tell us how to link different parts of, for instance, an instruction into a coherent whole. As we aim to demonstrate in this paper, discourse structure can help to clean up datasets for training and thereby improve training.

MSDC Thompson et al. (2024) provided full discourse annotations for the Minecraft corpus, known as the Minecraft Structured Dialogue Corpus (MSDC), using the discourse theory and annotation principles of SDRT (Asher, 1993; Asher and Lascarides, 2003) extended to a multimodal environment, in which both nonlinguistic actions and discourse moves can

	Train+Val	Test	Total
Original MDC			
# Dialogues	410	137	547
MSDC			
# Dialogues	407	133	540
# EDUs	17135	5402	22537
# EEUs	25555	7258	32813
# EEUs *squished*	4687	1473	6160
# Relation instances	26279	8250	34529

Table 1: MDC and MSDC characteristics.

```
<A> on the 3rd block from the ground
add a yellow block on the right side
of the column </A>

   place yellow -1 1 0
   pick -1 1 0
   place yellow -1 4 0

<B> there? </B>

   place yellow -1 3 0
   pick -1 4 0
   place yellow -1 4 0
   pick -1 3 0
```

Figure 2: An excerpt from Minecraft Corpus. The Builder interrupts the action sequence by asking a question.

enter into semantic relations like Elaboration, Correction, and Narration (Hunter et al., 2018; Asher et al., 2020). They followed annotation practices given for the STAC corpus (Asher et al., 2016). Thompson et al. (2024) also adapted the parser from Bennis et al. (2023) to predict discourse structures for the Minecraft corpus with relatively high reliability. Statistics on the MSDC are in Table 1.

LLMs in robotics Parallel to this work, there has been an increasing amount of research in aiding virtual or real robots with tasks by using LLMs to provide translations from natural language instruction to code that programs the robot to perform the relevant actions (Liang et al., 2023; Singh et al., 2023; Yu et al., 2023). This research is directly relevant to our work, as we use LLMs to go from natural language to a pseudo-code of pick and place statements. However, whereas Liang et al. (2023); Singh et al. (2023); Yu et al. (2023) focus on optimizing the translation from instructions, typically one instruction, to various different coding paradigms, we focus on how linguistic and nonlinguistic interactions affect the resulting action sequence. As our results and previous results on the MDC show, producing actions from interactive conversation with frequently underspecified instructions, which are also dependent upon the discourse and nonlinguistic contexts for proper interpretation, is a much more challenging task than translating well crafted unambiguous instructions into code. In addition, we show that to predict a relevant action from the instruction i_{n+1} in the MDC environment, it is not sufficient to use a context with just the penultimate instruction i_n and previous action sequence a_n.

3 NeBuLa: an LLM for Predicting Action Sequences

We've seen that Jayannavar et al. (2020)'s evaluation method for neural agents gives rather poor results. Observations of the results of neural Builder anecdotally yielded no ending configurations that matched those in the gold. The training scheme of Jayannavar et al. (2020) assumes, in effect, that Architect instructions and Builder actions follow one another with regularity. An unfortunate consequence of this assumption is that actions are individuated by the conversational turns that immediately precede and follow them. Jayannavar et al. (2020) initiate a new action sequence whenever there is a linguistic move of any kind. But that's not realistic, as bits of text don't always yield a well-formed or even an underspecified instruction. You might have a clarification question from the Builder in between two action sequences that are in fact carrying out one and the same action as in Figure 2. Builders in the MDC frequently ask questions with respect to the initial instruction about the actions they are simultaneously making; answers to those questions may affect the actions, but it doesn't mean that there are two distinct series of actions pertaining to two distinct instructions, one before the question and its response and one after. In addition, the Builder sometimes starts to build before the instruction sequence is complete; intuitively, the initial actions form a coherent action sequence with the actions that are subsequent to the

further instruction. These observations show that the assumptions of Jayannavar et al. (2020) about how actions are individuated are too simple.

Different conversational moves will change and make more precise the shape and position of the structure intended by the initial instruction. Hunter et al. (2018) note that different conversational moves can help conceptualize actions differently. For example, in many Minecraft sessions, an initial instruction gives the Builder an *action type* that might be realized in many different ways. Something like *build a tower of 5 blocks* is an action type for which a concrete realization would have to specify the color, perhaps the nature of the building blocks, and a location. As the conversation evolves and unless the Architect corrects their instruction, the type of action to be performed becomes more and more specified.

A simple baseline alternative to the scheme proposed by Jayannavar et al. (2020) that addresses the difficulties we just mentioned is to see how a model performs with the complete prior conversation and action sequences up to the predicted action. This was not an option for Jayannavar et al. (2020)'s model, but more recent LLMs are capable of doing this.

We used Llama-2-7B, Llama-2-13B and Llama-3-8B models to take as context all the conversation and action sequences up to action sequence a_n to predict a_n. We fine-tuned Llama on the MDC's (Jayannavar et al., 2020) training set. All the models were finetuned for 3 epochs using QLoRA method (Dettmers et al., 2023). Table 8 in the appendix provides details of computing resources and the hyperparameters for finetuning. Table 2 shows the net-action F1 scores on the validation and test set of MDC. All the finetuned LLMs significantly improved scores in comparison with the F1 0.20 score of Neural Builder (Jayannavar et al., 2020). Llama-3-8B essentially doubled the baseline score of 0.20. In the rest of the paper, we refer to Llama-3-8B finetuned on MDC as NeBuLa. The finetuned model, NeBuLa, will be made publicly available.

Dataset	Llama-2-7b	Llama-2-13b	Llama-3-8b
Validation	0.292	0.323	0.398
Test	0.326	0.338	0.392

Table 2: Net-Action F1 scores on Minecraft Validation and Test set for predicting action sequences for LLMs using the entire preceding linguistic and non linguistic actions in the game

4 Using Discourse Structure to Improve NeBuLa

The ideal way to model the instructional interactions is to have two ongoing, interleaved processes that interact and influence each other. On the one hand, there is the evolving conversational structure that helps conceptualize the nonlinguistic actions; on the other, there is the sequence of actions that also affects continuations of the given conversational context.

Using the discourse parser of Thompson et al. (2024), we made a first approximation of these interleaved processes by determining necessary and sufficient situated, conversational conditions for computing instructions. An analysis of the discourse structure in the MSDC shows a large scale pattern of so-called *Narrative arcs*. These arcs delimit portions of discourse structure linked by Narration relation. Each portion begins with an instruction i_n by the Architect, terminates with an action sequence a_m, and involves a negotiation between Architect and Builder about the action sequence to be performed. The negotiation may be extremely short, where the narrative portion then contains just i_n, a_m. On the other hand, it might be complex negotiation involving a number of EDUs related by relations like Elaboration. It may also involve questions of clarification or confirmation question by the Builder, in which case the instruction evolves through the portion. A narrative arc may also involve actions by the Builder that the Architect will correct with a linguistic move that will then result in a nonlinguistic action that revises or corrects the prior actions of the Builder. The end of the negotiation is the action sequence that finally carries out the instructions to the satisfaction of the Architect.

Figure 3 illustrates a narrative arc starting at Architect turn one with a new instruction that results (in green) in an action sequence in Builder turn two. The Builder then asks a complex, alternative question to confirm that this is the right move. The Architect replies to the question, in effect correcting (in red) the Builder's previous action, which then results in an action sequence in Builder turn four that corrects the previous builder action.

These arcs are relatively self-contained and are recoverable automatically to a relatively high

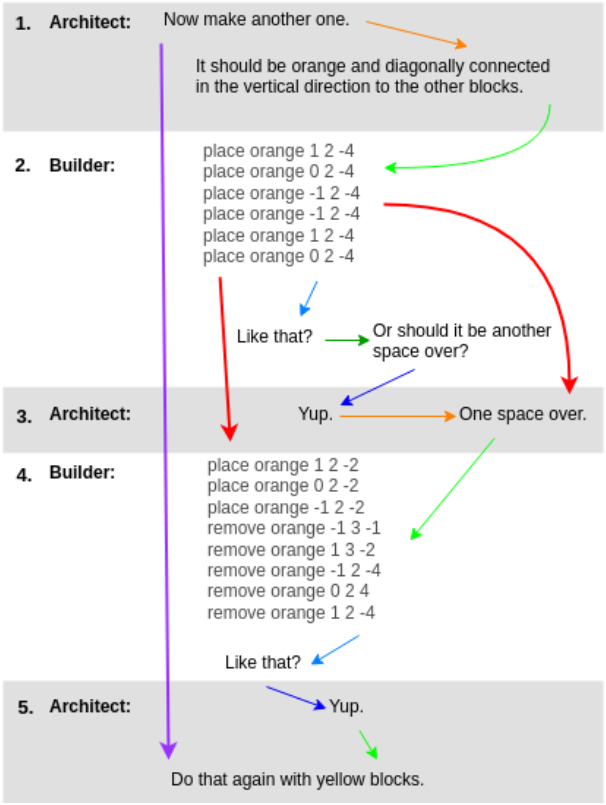

Figure 3: Excerpt of a narrative arc from the MSDC. Here the arc is purple and connects the instruction in Architect turn one to the following instruction in turn five.

degree by the parser of Thompson et al. (2024). So, instead of providing the entire conversation history as in Section 3 to the model, we provide the world-state at the beginning of the Narrative arc in terms of net place actions, and the discourse within the Narrative arc up to the present instruction i_n. We finetune Llama-3-8B on MDC training set using this input. We refer to the resultant model as NeBuLa+N (NeBuLa trained on Narrative arcs).

Table 3 gives scores on the validation and test set of MDC for NeBuLa+N(arration). From the table, we can see that the scores are comparable with original NeBuLa. This shows that the Narrative arc is *sufficient* for action prediction. To study whether the Narrative arc is *necessary* as well, we evaluated NeBuLa+N on those cases where $i_n a_n i_{n+1}$ had less content than the Narrative arc. There were 254 such cases in MDC test set. For these samples, we looked at performance of NeBuLa+N when worldstate along with the Narrative arc is given as input. This is denoted as NeBuLa+N/N in Table 3. Similarly, we looked at performance of NeBuLa+N when worldstate along with $i_n a_n i_{n+1}$ is given as input. This is denoted as NeBuLa+N/$i_n a_n i_{n+1}$ in Table 3. As we can see, the score for NeBuLa+N/$i_n a_n i_{n+1}$ is significantly lower ($\sim$ 10%) than NeBuLa+N/N. This shows that Narrative arcs are crucial for the task of action prediction.

Model	Validation	Test
NeBuLa+N	0.363	0.380
NeBuLa+N/N		0.349
NeBuLa+N/$i_n a_n i_{n+1}$		0.311

Table 3: Net-Action F1 scores on Minecraft Validation and Test set for predicting action sequences for LLMs. NeBuLa+N refers to NeBuLa trained on narrative arcs. The next two rows look at those 254 examples in the test set where $i_n a_n i_{n+1}$ has less content than the relevant Narrative arc. NeBuLa+N/N gives score of NeBuLa+N on these samples when world-state and narrative arc is given as input. Similarly, NeBuLa+N/$i_n a_n i_{n+1}$ gives score of NeBuLa+N on the same samples when world-state and $i_n a_n i_{n+1}$ is given as input.

5 Problems with the Minecraft Corpus

In Minecraft, the Architect makes use of several location descriptions. These descriptions are often anaphoric to blocks placed in prior instructions, such as *place another block next to that one* (one that was placed on previous Builder turn); locations are also sometimes vaguely designated (towards the centre) or underspecified (in a corner, along an edge, n blocks/spaces in from an edge/from the centre). Although the Minecraft environment presents (x, y, z) coordinates, the human participants never used them. This could be because, in the Minecraft environment, players can move their avatars around the board to get different perspectives, which makes it hard to establish an absolute coordinate system.

As a result, the net-action F1 metric, which evaluates a model's action sequence based on whether the block placements match exactly in terms of block color and coordinates with the corresponding gold builder action, is often inappropriate. For instance, if the Builder puts down a block at one corner after receiving the instruction *in a corner* whereas NeBuLa chooses another corner, the metric would give NeBuLa zero credit whereas intuitively it still did the right thing. To summarize, the evaluation metric treats vague instructions as completely precise ones, and considers one instantiation of an instruction (i.e. the action sequence of Builder in the gold data) to

be the only ground truth. Another related issue is highlighted in Figure 2 where the action sequence for the Architect's instruction gets truncated by a question from the Builder *"there?"*. In this case, for the aforementioned instruction, only the first three actions (*place yellow -1 1 0*, *pick -1 1 0*, *place yellow -1 4 0*) constitute the ground truth.

To conclude, the underspecified instructions with multiple plausible instantiations, coupled with the strict nature of the metric, puts an upper bound on how much the net-action F1 score can improve on this dataset. More importantly, it doesn't reveal what a model with a high F1 score actually does learn. We attempt to answer this in the next section.

6 Evaluating NeBuLa on Synthetic Dataset

Given the issues associated with Minecraft Corpus and the evaluation metric, we test NeBuLa on simple scenarios using a more just metric. We begin by testing NeBuLa's ability to construct simple shapes, such as, square, row, rectangle, tower, diagonal, diamond, cube of specific size and understand location (i.e. corner, centre, edge) and orientation descriptions (i.e. horizontal/vertical). We refer to all these shapes as **level-1 structures**. To do so, we construct a level-1 dataset of 1368 instructions. Some of these instructions simply ask to construct a shape of specific size like "Build a 3×3 red square.", while others are more detailed, for example, "Build a 3×3 red horizontal square at the centre."

For rows/diagonals/towers, we vary size from 3 to 9. For squares, the size varies from 3×3 to 5×5. For cubes, we only use $3 \times 3 \times 3$. For rectangles, we use sizes $m \times n$, where $m \neq n$, $m \times n < 30$ and $4 <= m <= 8$. For diamonds, we use two variants to describe size "m blocks on a side" and "axes $2m+1$ long", where $3 <= m <= 6$. We use orientation descriptions (i.e. horizontal/vertical) for squares, rectangles, and diamonds.

To evaluate NeBuLa on these instructions, we use simple binary functions $is\_square(C)$, $is\_tower(C)$ etc. for each shape. These functions take as input the predicted construction C and returns *True* if C is the desired shape, and *False* otherwise. For example, is_tower checks whether all the blocks have the same value for X and Z (as Y is the vertical dimension) and Y values are distinct and form a sequence $1, 2, ..., n$ where n is the number of predicted blocks.

For an instruction, we first evaluate if the predicted shape is correct. For correct shapes, we evaluate whether the size/color and location/orientation is correct (for instructions where location/orientation was specified). For an instruction with location description like *Build a red tower in a corner*, the location is considered correct if the predicted tower is in any of the four corners.

Table 4 gives the result of NeBuLa on level-1 dataset. We don't report color accuracy in the table, as NeBuLa always got the color correct. From the table, we can see that NeBuLa already has a decent command of basic shapes like towers, rows, and diagonals. However, it struggled with shapes like rectangle, square, cube, and diamond. It never correctly constructed diamonds, which might be because there were very few instances of diamonds in Minecraft corpus. For squares and rectangles which were correctly predicted, the model scored very high on orientation accuracy. However, the model has quite low location accuracy across all the correctly predicted shapes. The model rarely achieved an accuracy of above 50%, even with our relaxed evaluation method for locations.

As a second step, we look at NeBuLa's ability to understand location descriptions, in particular ones that are anaphorically specified. To do so, we start with an instantiation (randomly chosen from the set of correct instantiations) for the 1368 instructions in level-1 dataset. So, for a level-1 instruction such as "Build a 3×3 red square.", we have a 3×3 red square already present in the grid. Now given a level-1 structure in the grid, we design **level-2 instructions** which require placing or removal of a specific color block. For place instructions, we use location descriptions like *on top of*, *to the side of*, *touching*, and *not touching*. So an example of level-2 place instruction is *"place a blue block on top of that."* where *that* refers to the level-1 structure in the grid. Similarly, for removal instructions, we have the simple instruction "remove a block" and more complex instructions including location descriptions like *you just placed*. We also have additional location descriptions for certain level-1 structures such as *end* for rows, diagonals; *top, bottom* for towers; *corner* for cube; *centre* for cube, odd-size squares and towers. An example of level-2 remove instruction is *"remove the top block ."*

Similar to level-1, we evaluate NeBuLa on level-2 dataset by making use of binary functions like

shape	corr#	acc-shape	acc-size	loc-spec	acc-loc	or-spec	acc-or
Tower	504	100%	100%	378	56.0%		
Row	168	100%	100%	126	30.0%		
diagonal	168	78.6%	95.0%	102	2.0%		
rectangle	140	39.6%	12.0%	44	7.0%	31	100%
square	216	59.3%	96.0%	88	26.0%	75	81.0%
cube	24	58.3%	85.0%	8	37.0%		
diamond	144	0%	0%				
total	1368	73.0%	83.0%	746	38.0%	106	86%

Table 4: Evaluation of NeBuLa on shapes and basic locations. Shape accuracy gives how many of the shapes NeBuLa made were correct given the instruction. Additionally: acc-size–of those correct shapes how many were of the correct size; loc-spec–of the correct shapes how many had location specified; loc acc–of those correct shapes with specified locations how many were correctly located. We also tested rectangle and square for orientation (horizontal or vertical).

$is\_ontopof(b, C)$, $is\_touching(b, C)$ where C is the level-1 structure already present in the grid and b is the predicted block. As an example, for *on top of*, we check whether there is no block in C which is directly above the block b, and there is a block in C underneath block b. Mathematically, this can be expressed as $Coord_{\{y\}}(b) \cap \{y+1 : y \in max[Coord_{\{y\}}(C)]\} \neq \varnothing$; $Coord_{\{y\}}(b) \cap \{y : y < max[Coord_{\{y\}}(C)]\} = \varnothing$, and $Coord_{\{x,z\}}(C) \cap Coord_{\{x,z\}}(b) \neq \varnothing$ where $\text{Coord}_u(C)$ denotes the set of coordinates of the construction C for dimensions $u \subset \{X, Y, Z\}$.

Instruction	Accuracy	#correct	#total
Overall	80.4%	1100	1368
Overall place	67.9%	472	695
Overall removal	93.3%	628	673
Place on top of	74.2%	132	178
Place to the side of	98.1%	151	154
Place touching	99.4%	175	176
Place not touching	7.5%	14	187
Removal any	95.3%	223	234
Removal you just placed	95.3%	206	216
Removal top	100%	44	44
Removal bottom	100%	65	65
Removal centre	60.7%	34	56
Removal corner	100%	2	2
Removal end	96.4%	54	56

Table 5: Evaluation of NeBuLa place and remove instructions with anaphoric locations.

Table 5 shows that the model did quite well, with the exception of the instruction involving *not touching* as location description. Otherwise, the results indicate that NeBuLa has a good knowledge of basic anaphoric location descriptions.

We then examined NeBuLa's errors with *on top of*. We found that the failure cases mostly were a result of the model placing multiple blocks instead of just one on the given level-1 structure. That is, the model does not always understand *a block* as *a single block*. In light of these cases, when we check whether all the blocks in predicted b are on top of C, the accuracy improves from 74.2% to 97.2%. Thus, some of the difficulties NeBuLa had with instructions come from what might be a limited understanding of the semantics and pragmatics of indefinite and numerical noun phrases.

6.1 Finetuning NeBuLa on Shapes and Locations

Our evaluation on level-1 and level-2 data shows that NeBuLa struggles with squares, rectangles, diamonds, and "not touching" place instructions. To tackle this, we used a subset of the two datasets to augment the training data for NeBuLa. From level-1 data, we took the following subset for training: squares of size 3×3, diamonds of size 3 (or axes 5 spaces long), and rectangles of sizes 4×3 and 5×4. From level-2 data, we took those "touching/not touching" instances where the level-1 structure is square or rectangle. Out of total 363 instances for touching/not touching, there were 109 such instances. We then finetuned NeBuLa by combining the Minecraft training with this subset of level-1 and level-2 data. The rest of the level-1 and level-2 data was used for testing.

Table 6 shows NeBuLa's performance on level-1 test set after finetuning. As before, we found that NeBuLa always got the color correct. From the table, we can see that the shape accuracy improved significantly for squares, rectangles, and diamonds in comparison with Table 4. Although the location accuracy is still low, it has improved in comparison with original NeBuLa. Interestingly, we also see

shape	tot#	acc-shape	acc-size	loc-spec	acc-loc	or-spec	acc-or
Tower	504	99.0%	100%	377	42.0%		
Row	168	99.0%	100%	125	48.0%		
diagonal	168	74.0%	80.0%	101	39.0%		
rectangle	102	95.0%	49.0%	76	32.0%	65	100%
square	144	89.0%	100%	93	45.0%	86	100%
cube	24	100%	100%	18	66.0%		
diamond	108	18.0%	0%			12	100%
total	1218	87.0%	90.0%	715	46.0%	163	100%

Table 6: Evaluation of NeBuLa after finetuning on shapes and basic locations. Shape accuracy (acc-shape), acc-size, location-spec, acc-loc, or-spec and acc-or as in Table 4

Instruction	Accuracy	#correct	#total
Overall	89.6%	1128	1259
Overall place	88.7%	520	586
Overall removal	90.3%	608	673
Place on top of	79.7%	142	178
Place to the side of	87.7%	135	154
Place touching	93.3%	112	120
Place not touching	97.8%	131	134
Removal any	94.4%	221	234
Removal you just placed	84.7%	183	216
Removal top	100%	44	44
Removal bottom	100%	65	65
Removal centre	66.1%	37	56
Removal corner	100%	2	2
Removal end	100%	56	56

Table 7: Evaluation of NeBuLa with additional finetuning on touching/not touching.

that NeBuLa has perfect shape accuracy on cube although cube was not part of the training set. Finally, for correctly predicted shapes, NeBuLa achieved a perfect orientation accuracy.

Table 7 shows the results on level-2 test set for NeBuLa after finetuning. Here also, we can see that NeBuLa's accuracy remains very high on almost all of the simple instructions with the anaphoric location descriptions. Furthermore, its accuracy increased significantly for "not touching" instructions. This jump in accuracy is significant enough to conclude that NeBuLa has learned the concept of "contact", at least for our synthetic dataset. On the minecraft test set, we found that NeBuLa's performance remained high with an average precision of 0.40, recall of 0.414 and a net action F1 of 0.391. As we can see, these scores are at-par with the baseline NeBuLa.

7 Conclusions and Future Work

We have introduced NeBuLa, an LLM based action prediction model, for the Minecraft Dialogue Corpus. As a baseline, NeBuLa uses the entire Minecraft dialogue up to action a_n to predict a_n. We showed that this baseline doubles the net action F1 scores of Jayannavar et al. (2020). We then showed that certain discourse structures provided necessary and sufficient information for inferring actions to the level of the baseline. We also analyzed NeBuLa's errors on Minecraft corpus and provided additional finetuning to improve the model's ability to interpret underspecified shape descriptions and anaphorically-specified locations using our synthetic dataset. This allowed us to analyze the shortcomings of the net-action F1 metric, and address them using a more realistic evaluation metric. Our evaluation metric captures the notion of relative location, but leaves exact locations typically underspecified, in accordance with our semantic intuitions. For future work, we plan to apply this metric (or a similar relative location metric) on the Minecraft corpus. Given the improvement in performance of NeBuLa after finetuning on our synthetic dataset, we hypothesize that in a more controlled collaborative task, with some pedagogical instructions to the Architect, NeBuLa could contribute as a useful interface for conversational robots that interact with humans.

Limitations

The MSDC contains a great deal of discourse information, including a full discourse structure analysis. We have only used some of this information. Potentially, we could leverage more information from this dataset to improve NeBuLa's action prediction performance. We also need to extend our constraints to cover other frequent anaphoric location descriptions in addition to *on top of X* and *to the side of X*. Locutions like *in front of/ behind, underneath, hanging off, next to* (X) all have underspecified parameters of either

orientation, distance or direction that allow for several correct placements, once X has been identified. For instance, $\mathit{Under}(X, Y)$ holds if $Coord_{\{y\}}(X) = \{z - n : y \in Coord_{\{y\}}(Y)\}$ and $Coord_{\{x,z\}}(X) \cap Coord_{\{x,z\}}(Y) \neq \varnothing$. We need to evaluate NeBuLa on these expressions as well. Finally, we need to reevaluate NeBuLa's predictions as well as builder actions in the MDC with our more appropriate metric, which is suited to the underspecified shape and location descriptions used in the corpus.

Ethics Statement

Our work here has been to improve the capacities of AI systems in interactive tasks where conversation can be used to optimize performance on collaborative actions. We see no direct ethical concerns that arise from this work. Though conversationally more capable robots, which could be one downstream application of this work, might require additional conversational strategies as constraints to ensure that participating humans retain the final say with regards to the actions in the collaborative tasks.

References

Asher, N. (1993). *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers.

Asher, N., Hunter, J., Morey, M., Benamara, F., and Afantenos, S. (2016). Discourse structure and dialogue acts in multiparty dialogue: the stac corpus. In *10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2721–2727.

Asher, N., Hunter, J., and Thompson, K. (2020). Modelling structures for situated discourse. *Dialogue & Discourse*, 11:89–121.

Asher, N. and Lascarides, A. (2003). *Logics of Conversation*. Cambridge University Press, New York, NY.

Bennis, Z., Hunter, J., and Asher, N. (2023). A simple but effective model for attachment in discourse parsing with multi-task learning for relation labeling. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3404–3409.

Bonial, C., Abrams, M., Traum, D., and Voss, C. (2021). Builder, we have done it: evaluating & extending dialogue-amr nlu pipeline for two collaborative domains. In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pages 173–183.

Bonial, C., Donatelli, L., Abrams, M., Lukin, S., Tratz, S., Marge, M., Artstein, R., Traum, D., and Voss, C. (2020). Dialogue-amr: abstract meaning representation for dialogue. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 684–695.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In Wu, D., Carpuat, M., Carreras, X., and Vecchi, E. M., editors, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Hunter, J., Asher, N., and Lascarides, A. (2018). Situated conversation. *Semantics and Pragmatics*, 11(10). doi: 10.3765/sp.11.10.

Jayannavar, P., Narayan-Chen, A., and Hockenmaier, J. (2020). Learning to execute instructions in a minecraft dialogue. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 2589–2602.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. (2023). Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE.

Narayan-Chen, A., Jayannavar, P., and Hockenmaier, J. (2019). Collaborative dialogue in minecraft. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5405–5415.

Shi, Z., Feng, Y., and Lipani, A. (2022). Learning to execute actions or ask clarification questions. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2060–2070.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. (2023). Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE.

Thompson, K., Hunter, J., and Asher, N. (2024). Discourse structure for the Minecraft corpus. In Calzolari, N., Kan, M.-Y., Hoste, V., Lenci, A., Sakti, S., and Xue, N., editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4957–4967, Torino, Italia. ELRA and ICCL.

Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Arenas, M. G., Chiang, H.-T. L., Erez, T., Hasenclever, L., Humplik, J., et al. (2023). Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*.

A Appendix

GPUs	
4 NVIDIA Volta V100	
Hyperparameters	
Training epochs	3
batch size	4
optimizer	Adam
learning rate	2e-4
learning rate scheduler	linear warm-up and cosine annealing
warm-up ratio	0.03
gradient clipping	0.3
lora r	64
lora (alpha)	16
lora dropout ratio	0.1
lora target modules	Only Attention Blocks (q_proj, v_proj)
quantization for LLaMA3	4-bit NormalFloat

Table 8: Details on computing resources and hyperparameters for finetuning NeBuLa.

Table 8 gives the hyperparameters used for finetuning NeBuLa along with the computing resources. We adapted the finetuning code from the following repository[1]. We provide level-1 and level-2 synthetic data as part of the supplementary material.

[1] https://github.com/mlabonne/llm-course/blob/main/Fine_tune_Llama_2_in_Google_Colab.ipynb