Nebula: A Discourse-Aware Minecraft Builder

Akshay Chaturvedi[†], Kate Thompson*[†], Nicholas Asher[†‡]
[†]IRIT, [‡]CNRS, *LINAGORA Labs
Toulouse, France

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 finetune 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 interaction 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 **Lla**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., 2024b), 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 output reveals two other factors that 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; 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.

The main contributions of this work are: (i) to show that finetuning Llama-3-8B (Dubey et al., 2024) on the entire prior history of the action-

prediction task Nebula doubles the net-action F1 score on the Minecraft dataset as compared with the Neural Builder of Jayannavar et al. (2020); (ii) to show that training on *Narrative arcs* achieves comparable results with baseline Nebula, and that training on narrative arcs performs better than the instruction-action-instruction input template used by Jayannavar et al. (2020); (iii) to highlight the drawbacks of the evaluation metric (i.e., net-action F1), and propose and test a new metric on our synthetic datasets.

After some preliminaries and discussion of prior work (Section 2), we present our model, Nebula, 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 the Minecraft corpus. We try to address these issues in Section 6, where we explain our evaluation metric for underspecified instructions, as well as experiments on our synthetic datasets. We conclude in Section 7.

2 Related Work

MDC Narayan-Chen et al. (2019) construct 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. The corpus has also led to related work on conversational interactive agents (Kiseleva et al., 2022; Mohanty et al., 2023; Madge and Poesio, 2024).

Instructions to code: Neural Builder and variants The MDC (Jayannavar et al., 2020) incentivizes the development of an algorithm that can 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) train a model consisting of a GRU (Cho et al., 2014) to handle textual input coupled with a CNN to
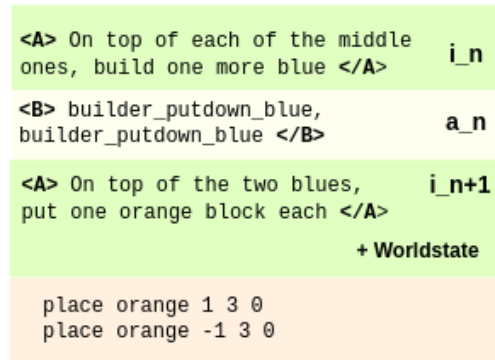

Figure 1: The Neural Builder (Jayannavar et al., 2020) takes as input the sequence $i\_n\ a\_n\ i\_n+1$ and the worldstate to predict the subsequent action sequence.

integrate information from the current state and a GRU to predicted an action sequence. Although they experiment with several training regimes, the best performance comes 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) show that the problem of predicting action sequences from natural language instructions in naturally occurring dialogue remains extremely challenging. Their Neural Builder has 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 annotate all Builder dialogue moves with a taxonomy of dialogue acts. They then specify 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) add dialogue acts to Minecraft utterances, but they do not evaluate the effect of these dialogue acts on the Neural Builder's predictions of actions. Dialogue acts are

	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. EDU and EEU refer to elementary discourse unit, and elementary event units respectively.

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. (2024b) provide full discourse annotations for the MDC, 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 enter into semantic relations like Elaboration, Correction, and Narration (Hunter et al., 2018; Asher et al., 2020). They follow annotation practices given for the STAC corpus (Asher et al., 2016). Thompson et al. (2024b) also adapt the parser from Bennis et al. (2023) to predict discourse structures for the MDC 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

```
<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 MDC. The Builder interrupts the action sequence by asking a question.

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 current instruction i_{n+1} in the MDC environment, it is not sufficient to use a context of 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 Neural Builder performs poorly as per their evaluation method, i.e., net-action F1. The training scheme of Jayannavar et al. (2020) assumes, in effect, that Architect instructions and the Builder actions which fulfill these instructions follow one another in a regular succession. A consequence of this assumption is that actions are individuated by the conversational turns that immediately precede and follow them, and likewise that a new action sequence is initiated whenever there is a linguistic move of any kind. Neural Builder predicts actions from a preceding context containing an

instruction, and the instruction just before it, and a representation of the builder moves in between: $i_n a_n i_{n+1}$ predicts a_{n+1}. Additionally, Neural Builder is provided a worldstate representation (See Figure 1).

But this is not realistic, as these bits of text don't always yield a well-formed instruction or even an underspecified one. In addition, often actions and instructions occur simultaneously. We might have a clarification question from the Builder in between two action sequences that are in fact carrying out 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 currently carrying out; 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.

The shape and position of the structure, intended by the initial instruction, can change or be made more precise by different conversational moves. 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, and perhaps the nature of the building blocks, as well as 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 these difficulties 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 use Llama-2-7B, Llama-2-13B and Llama-3-8B models to take as context all the conversation

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

and action sequences up to action sequence a_n to predict a_n. We finetune Llama on the MDC's (Jayannavar et al., 2020) training set. All the models are finetuned for 3 epochs using QLoRA method (Dettmers et al., 2023). Table 2 shows the net-action F1 scores on the validation and test set of MDC. All the finetuned LLMs significantly improve scores in comparison with the 0.20 F1 score of Neural Builder (Jayannavar et al., 2020). Llama-3-8B essentially doubles the baseline score of 0.20. In the rest of this paper, we refer to Llama-3-8B finetuned on MDC as Nebula. The finetuned model, Nebula, and the synthetic datasets used in Section 6 are available here[1]. Table 6 in the Appendix provides details of computing resources and the hyperparameters for finetuning.

4 Using Discourse Structure to Improve Nebula

The ideal way to model collaborative, instructional interactions like those featured in the MDC is to have two simultaneous, interleaved processes that interact with each other. On one hand, there is the evolving conversational structure that conceptualizes the nonlinguistic actions; on the other, there is the sequence of actions that also affects continuations of the conversational structure.

This interleaved process is modeled by the discourse structure provided by the MSDC of Thompson et al. (2024b). The MSDC shows a large scale pattern of *Narrative arcs*. These arcs delimit portions of discourse structure linked by a Narration relation. Each portion begins with an instruction i_n from the Architect, terminates with an action sequence a_m, and involves a negotiation between the Architect and the Builder about the action sequence to be performed.

The negotiation may be extremely short, where the arc just contains a single instruction and resulting (correct) action i_n, a_m. But it can also contain a complex negotiation involving a number
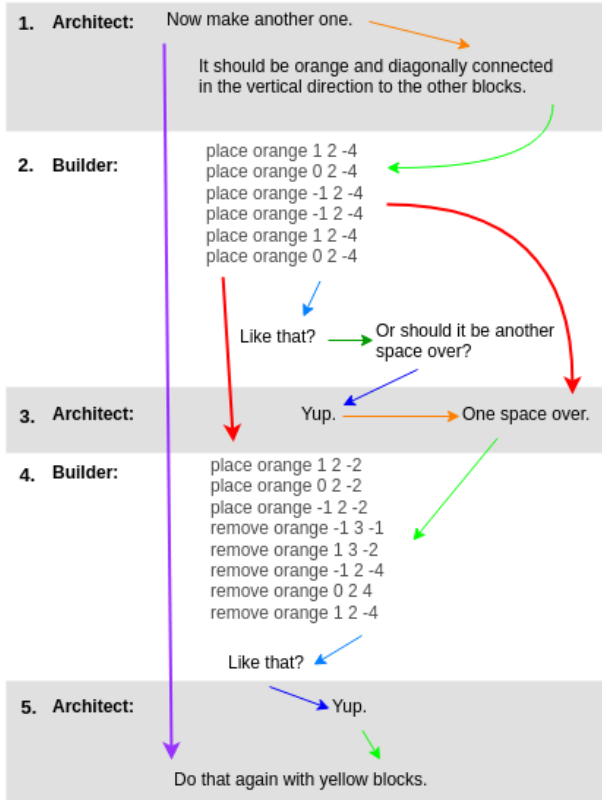
---

[1] https://huggingface.co/linagora/Nebula

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.

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 sets 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 associated Narrative. Nebula+N/N gives score of Nebula+N on these samples when worldstate 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 worldstate and $i_n a_n i_{n+1}$ is given as input.

of discourse moves that serve to elaborate or clarify the initial instruction. In this way, the content of a single instruction i_n may evolve over many discourse moves. Further, if the initial builder actions are incorrect, this will prompt corrective moves by the Architect and subsequent revisions by the Builder, leading to longer collaborative negotiations until the Builder carries out the action sequence that satisfies the Architect's initial instruction.

Figure 3 illustrates a long negotiation enclosed by a Narrative arc. The instruction in the first turn results (in green) in a Builder action sequence in turn two. The Builder then asks a question to confirm that the actions in turn two were correct. The Architect replies to the question by correcting (in red) the actions in turn two, which then results in a corrective action sequence in turn four.

Guided by the assumption that these Narrative arcs contain all the contextual information—the initial instruction and all subsequent discourse moves—that is needed to predict the correct actions, we use the Narrative arcs provided by

the MSDC to assign the previous context to each action sequence in the MDC. Since the arcs are automatically recoverable to a relatively high degree by the parsers discussed in Thompson et al. (2024a,b), we make the expedient choice to use the gold arcs to test our assumption.

We finetune Llama-3-8B on the MDC training set using instruction-action pairs, where the instruction is only the conversation within the Narrative arc up to the present instruction i_{n+1} in the pair, not the entire conversation history, which we fed to baseline Nebula. Since Neural Builder supplements attenuated contexts with a worldstate representation (see Section 3) we also include a "worldstate" at the beginning of the Narrative arc in terms of net place actions. We refer to the resulting model as Nebula+N (Nebula trained on Narrative arcs).

Table 3 shows scores on the validation and test sets of the MDC for Nebula+N. We can see that the scores are comparable with Nebula, 0.38 compared to 0.39 F1, (see Llama-3-8B in Table 2). This provides evidence that the discourse information present in a Narrative arc is *sufficient* for action prediction within that arc.

The majority of the Narrative arcs are shorter than or of equivalent length to their $i_n a_n i_{n+1}$ counterparts used to train the Neural Builder (1321 out of 1575 samples in the test set). Going on length alone, it is reasonable to assume Narrative arcs and $i_n a_n i_{n+1}$ are interchangeable. However, if Narrative arcs are not only sufficient but also *necessary* for action prediction, then in the 254 samples where Narrative arc is longer than $i_n a_n i_{n+1}$ we would expect Nebula+N to perform better on those samples when given the Narrative

arc. And this is the case.

In Table 3, we compare the performance of Nebula+N when worldstate along with the Narrative arc is given as input (denoted as Nebula+N/N), with Nebula+N when worldstate along with $i_n a_n i_{n+1}$ is given as input (denoted as Nebula+N/$i_n a_n i_{n+1}$). As we can see, the score for Nebula+N/$i_n a_n i_{n+1}$ is considerably lower than Nebula+N/N. This provides evidence that Narrative arcs play a crucial role, in that the information they provide is both necessary and sufficient for action prediction in the MDC.

5   Problems with the MDC

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 executed the instruction correctly. To summarize, the net-action F1 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.

Thus, 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 Data

Given the issues associated with MDC and the evaluation metric, we test baseline Nebula (i.e., the one trained on entire conversation history) on simple scenarios using a more just metric. To do so, we construct synthetic datasets at two different levels. The goal of these datasets is to test what basic shapes and location descriptors Nebula learns after being trained on MDC. For the first level, we test 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**. The resulting dataset, referred to as level-1 dataset, consists 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

Shape	Total #		ShapeAcc%		SizeAcc%		Loc-spec		LocAcc%		Orient-spec		OrientAcc%	
	BL	FT	BL	FT	BL	FT	BL	FT	BL	FT	BL	FT	BL	FT
Tower	504	504	100	99.0	100	100	378	377	56.0	42.0				
Row	168	168	100	99.0	100	100	126	125	30.0	48.0				
Diagonal	168	168	78.6	74.0	95.0	80.0	102	101	2.0	39.0				
Rectangle	140	102	39.6	95.0	12.0	49.0	44	76	7.0	32.0	31	65	100	100
Square	216	144	59.3	89.0	96.0	100	88	93	26.0	45.0	75	86	81.0	100
Cube	24	24	58.3	100	85.0	100	8	18	37.0	66.0				
Diamond	144	108	0	18.0	0	0						12		100
Total	1368	1218	73.0	87.0	83.0	90.0	746	790	38.0	46.0	106	163	86.0	100

Table 4: Evaluation of baseline Nebula (BL) and Nebula finetuned further on a part of synthetic data (FT), on shapes and basic locations. ShapeAcc% gives percentage of cases where the given shape was correct. Additionally, SizeAcc% denotes, for the correct shapes, percentage of cases where it is of the correct size; Loc-spec denotes, for the correct shapes, how many have location specified; LocAcc% denotes location accuracy for such cases. We also test rectangle and square for orientation (horizontal or vertical). Orient-spec denotes, for the correct shapes, the number of cases where orientation is specified; and OrientAcc% denotes the orientation accuracy for the same.

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 baseline Nebula (refer to BL in the table) on level-1 dataset. We don't report color accuracy, as Nebula always gets 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 struggles with shapes like rectangle, square, cube, and diamond. It never correctly constructs diamonds, which might be because there are very few instances of diamonds in MDC. For squares and rectangles which are correctly predicted, the model scores very high on orientation accuracy. However, the model has quite low location accuracy across all the correctly predicted shapes. The model rarely achieves an accuracy of above 50%, even with our relaxed evaluation method for locations.

We also evaluate the performance of Neural Builder (Jayannavar et al., 2020) on our level-1 dataset. The results are shown in Table 7 of the Appendix. Apart from rows and towers, Neural Builder model fails to build any shape correctly.

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."* Both level-1 and level-2 datasets were generated in an automated fashion.

Similar to level-1, we evaluate Nebula on level-2 dataset by making use of binary functions like $is\_ontopof(b, C)$, $is\_touching(b, C)$ where C is the level-1 structure already present in the grid and b is the predicted block. For example, binary function for *on top of* checks whether there is no block in C which is directly above the block b, and there is a block in C underneath block b.

Table 5 shows that baseline Nebula performs quite well (refer to BL in the table), with the exception of the instruction involving *not touching* as location description. These results indicate that Nebula has a good knowledge of basic anaphoric location descriptions. Similar to level-1 dataset, we look at the performance of Neural Builder (Jayannavar et al., 2020) on the level-2

Instruction	Total #		Accuracy(%)	
	BL	FT	BL	FT
Overall	1368	1259	80.4	89.6
Place...				
on top of	178	178	74.2	79.7
to the side of	154	154	98.1	87.7
touching	176	120	99.4	93.3
not touching	187	134	7.5	**97.8**
Place Overall	695	586	67.9	88.7
Remove...				
any block	234	234	95.3	94.4
block just placed	216	216	95.3	84.7
top block	44	44	100	100
bottom block	65	65	100	100
centre block	56	56	60.7	66.1
corner block	2	2	100	100
end block	56	56	96.4	100
Remove Overall	673	673	93.3	90.3

Table 5: Evaluation of baseline Nebula (BL) and Nebula finetuned further on a part of synthetic data (FT), on location descriptors for *place* and *remove* instructions. The FT model performs considerably better for "not touching" place instructions while remaining at-par for other instruction types.

dataset. Table 8 in the Appendix shows that the performance of the Neural Builder is considerably worse than Nebula for all the location descriptors.

We then examine Nebula's errors with *on top of*. We find 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. We provide visual comparisons of Neural Builder and baseline Nebula on level-1 and level-2 dataset examples in Tables 9 and 10 of the Appendix.

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 use a subset of the two datasets to augment the training data for Nebula. From level-1 data, we take 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 take those "touching/not touching" instances where the level-1 structure is square or rectangle. Out of total 363 instances for touching/not touching, there are 109 such instances. We then finetune 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 is used for testing.

Table 4 shows Nebula's performance on the level-1 test set after finetuning (refer to FT in the table). As before, we find that Nebula always got the color correct. From the table, we can see that the shape accuracy improves significantly for squares, rectangles, and diamonds in comparison to baseline Nebula (BL in the table). Although the location accuracy is still low, it improves in comparison with baseline Nebula. Interestingly, we also see that Nebula has perfect shape accuracy on cube, although cube is not part of the training set. Finally, for correctly predicted shapes, Nebula achieves a perfect orientation accuracy.

Table 5 shows the results on the 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 increases drastically 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 find that Nebula's performance remains high with a net action F1 of 0.391. As we can see, these scores are at-par with the baseline Nebula (refer to llama-3-8b in Table 2).

7 Conclusions and Future Work

We introduce 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 show that this baseline doubles the net action F1 scores of Jayannavar et al. (2020). We then show that certain discourse structures provide necessary and sufficient information for

inferring actions at-par with the baseline setting. We also analyze Nebula's errors on MDC and provide additional finetuning to improve the model's ability to interpret underspecified shape descriptions and anaphorically-specified locations using our synthetic dataset. This allows 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 explore metrics similar to our relative location metric that can be applied more generally, including on the MDC. 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 only use 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. We need to evaluate Nebula on these expressions as well. Finally, we need to reevaluate Nebula's predictions as well as the original 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. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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.

Kiseleva, J., Li, Z., Aliannejadi, M., Mohanty, S., ter Hoeve, M., Burtsev, M., Skrynnik, A., Zholus, A., Panov, A., Srinet, K., et al. (2022). Interactive grounded language understanding in a collaborative environment: IGLU 2021. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 146–161. PMLR.

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.

Madge, C. and Poesio, M. (2024). Large Language Models as Minecraft Agents. *arXiv preprint arXiv:2402.08392*.

Mohanty, S., Arabzadeh, N., Kiseleva, J., Zholus, A., Teruel, M., Awadallah, A., Sun, Y., Srinet, K., and Szlam, A. (2023). Transforming Human-Centered AI Collaboration: Redefining Embodied Agents Capabilities through Interactive Grounded Language Instructions. *arXiv preprint arXiv:2305.10783*.

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., Chaturvedi, A., Hunter, J., and Asher, N. (2024a). LLaMIPa: An Incremental Discourse Parser. *arXiv preprint arXiv:2406.18256*.

Thompson, K., Hunter, J., and Asher, N. (2024b). 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 Llama-3	4-bit NormalFloat

Table 6: Details on computing resources and hyperparameters for finetuning Nebula.

Table 6 gives the hyperparameters used for finetuning Nebula along with the computing resources. We adapt the finetuning code from the following repository[2].

---

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

Shape	Total #	ShapeAcc%	SizeAcc%	Loc-spec	LocAcc%	Orient-spec	OrientAcc%
Tower	504	75.6	20.5	303	0	0	0
Row	168	97.6	16.6	126	32.5	0	0
Diagonal	168	0	0	0	0	0	0
Rectangle	140	0	0	0	0	0	0
Square	216	0	0	0	0	0	0
Cube	24	0	0	0	0	0	0
Diamond	144	0	0	0	0	0	0
Total	1368	39.8	19.4	429	9.5	0	0

Table 7: Evaluation of Neural Builder (Jayannavar et al., 2020) on shapes and basic locations. ShapeAcc% gives percentage of cases where the given shape was correct. Additionally, SizeAcc% denotes, for the correct shapes, percentage of cases where it was of the correct size; Loc-spec denotes, for the correct shapes, how many had location specified; LocAcc% denotes location accuracy for such cases. We also tested rectangle and square for orientation (horizontal or vertical). Orient-spec denotes, for the correct shapes, the number of cases where orientation was specified; and OrientAcc% denotes the orientation accuracy for the same.

Instruction	Total #	Accuracy(%)
Overall	1368	50.9
Place...		
on top of	178	60.1
to the side of	154	66.2
touching	176	98.3
not touching	187	0.5
Place Overall	695	55.1
Remove...		
any block	234	88.5
block just placed	216	14.8
top block	44	2.3
bottom block	65	35.4
centre block	56	16.1
corner block	2	0
end block	56	73.2
Remove Overall	673	46.5

Table 8: Evaluation of Neural Builder (Jayannavar et al., 2020) on location descriptors for *place* and *remove* instructions.
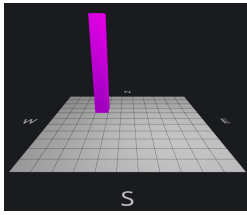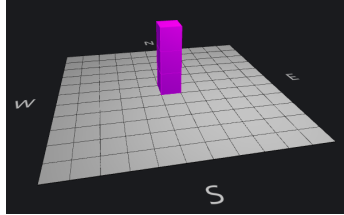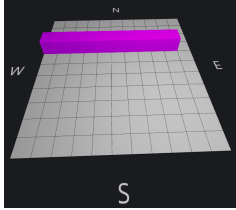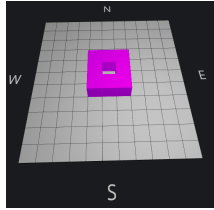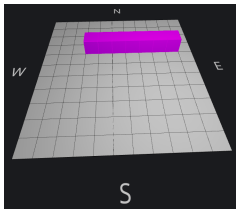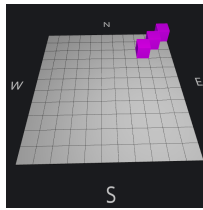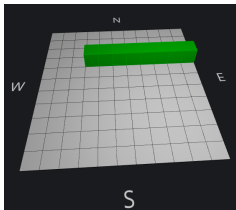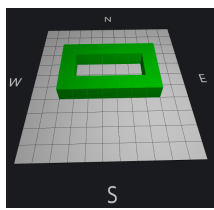
Lvl-1 Instruction	Neural Builder	Nebula
Build a purple tower of size 3 at the centre.		
Build a 3x3 purple square at the centre.		
Build a diagonal of 3 purple blocks at an edge.		
Build a 6x3 green rectangle at the centre.		

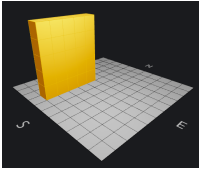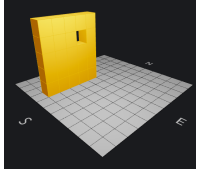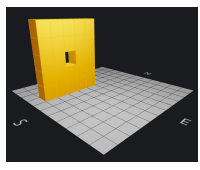Table 9: Comparison of Neural Builder (Jayannavar et al., 2020) and baseline Nebula on level-1 dataset.
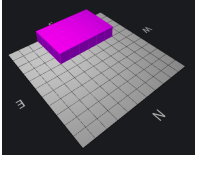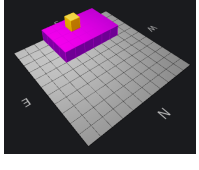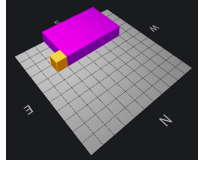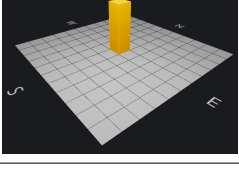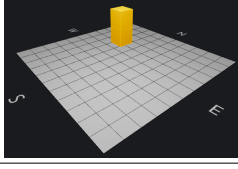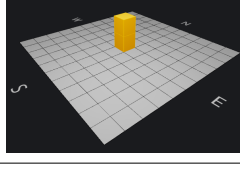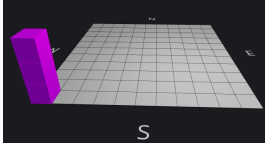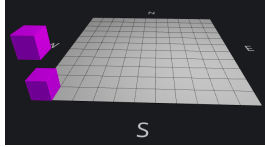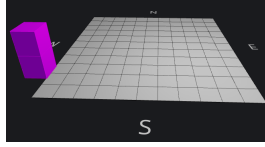
World State + Lvl-2 Instruction	Neural Builder	Nebula
OK now remove the centre block.		
OK now place a yellow block to the side of that.		
OK now remove the top block.		
OK now remove the bottom block.		

Table 10: Comparison of Neural Builder (Jayannavar et al., 2020) and baseline Nebula on level-2 dataset.