Learning to execute or ask clarification questions

Anonymous ACL submission

Abstract

Collaborative tasks are ubiquitous activities where a form of communication is required in order to reach a joint goal. Collaborative building is one of such tasks. To this end, we wish to develop an intelligent builder agent in a simulated building environment (Minecraft) that can build whatever users wish to build by just talking to the agent. However, in order to achieve this goal, such agents need to be able to take the initiative by asking clarification questions when further information is needed. Existing work on Minecraft Corpus Dataset only learned to execute instructions neglecting the importance of asking for clarifications. In this paper, we extend the Minecraft Corpus Dataset by annotating all builder utterances into eight types, including clarification questions, and propose a new builder agent model capable of determining when to ask or execute instructions. Experimental results show that our model achieves state-of-the-art performance on the collaborative building task with a substantial improvement. We also provide baselines for the new tasks, *learning to ask* and the joint tasks, which consists in solving both collaborating building and learning to ask tasks jointly.

1 Introduction

Following instructions in natural language by intelligent agents to achieve a shared goal with the instructors in a pre-defined environment is a ubiquitous task in many scenarios, e.g., finding a target object in an environment (Nguyen and Daumé III, 2019; Roman et al., 2020), drawing a picture (Lachmy et al., 2021), or building a target structure (Narayan-Chen et al., 2019). A number of machine learning (ML) research projects about following instructions tasks have been initiated by making use of the video game Minecraft (Johnson et al., 2016; Shu et al., 2018; Narayan-Chen et al., 2019; Guss et al., 2019; Jayannavar et al., 2020). Building such agents requires to make progress in *grounded natural language understanding* – understanding complex instructions, for example, with spatial relations in natural language – *self-improvement* – studying how to flexibly learn from human interactions – *synergies of ML components* – exploring the integration of several ML and non-ML components to make them work together (Szlam et al., 2019).

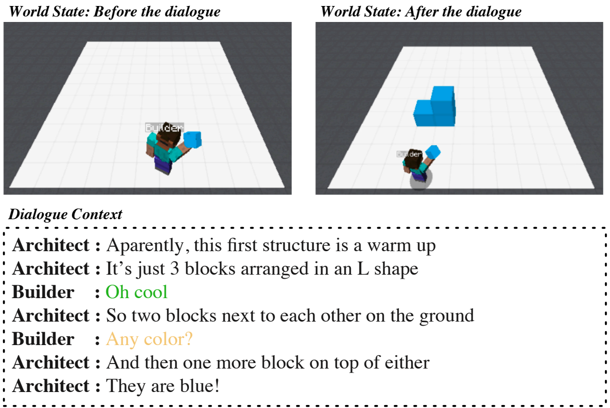

Figure 1: A simple example of builder task: The builder can observe the world state and dialogue context. For the sake of space, only a part of the dialogue history is displayed. The utterance in green displays understanding and the utterance in yellow asks a clarification question.

The recently introduced Minecraft Corpus dataset (Narayan-Chen et al., 2019) proposes a collaborative building task, in which an architect and a builder can communicate via a textual chat. Architects are provided with a target structure they want to have built, and the builders are the only ones who can control the Minecraft avatar in the virtual environment. The task consists in building 3D structures in a block world-like scenario collaboratively, as shown in the Figure 1. Earlier works in Minecraft collaborative building tasks (Jayannavar et al., 2020) attempted to build an automated builder agent with a large action space but failed to allow the builder to take the initiative in the conversation. However, an intelligent agent should not only understand and execute the instructor's requests but also be able to take initiatives, e.g., asking clarification questions, in case the instructions are ambiguous. In the task defined by this dataset, builders may encounter ambiguous situations that are hard to interpret by just relying on the world state information and instructions. For example, in Figure 1, we provide a simple case where the architect fails to provide sufficient information to the builder, such as the color of the blocks. In this situation, it is clearly difficult for the builder to know exactly which action should be taken. If, however, the builder is able to clarify the situation with the architect, this ambiguity can be resolved. Therefore, builders, besides following architects' instructions, should take the initiative in the conversation and ask questions when necessary.

To this end, in this paper we annotate all builder utterances in the Minecraft Corpus dataset by categorizing them in the dataset into eight dialogue utterance types as shown in Table 2, allowing the intelligent agents to learn when and what to ask given the world state and dialogue context. Particularly, a builder would ask *task-level questions* or *instruction-level questions* for further clarifications. Experimental results in the Sec. 5.2 show that determining when to ask clarification questions remains a challenging task. However, it is worth noting that the clarification questions in the Minecraft Corpus dataset are more complex and diverse than those in navigation tasks (Roman et al., 2020; Thomason et al., 2020; Zhu et al., 2021; Nguyen and Daumé III, 2019) whose questions are relatively simpler and mainly about where to go.

Also, we propose a new automated builder agent that learns to map instructions to actions and decide when to ask questions. Our model utilizes three dialogue slots, the action type slot, the location slot, and the color slot. This solution has the benefit of making the learning easier with respect to those models that work using a large action space (Jayannavar et al., 2020). To solve the collaborative building task, both the dialogue context and the world state need to be considered. Therefore, to endow our model with the ability to better learn the representations between the world state and language, our model implements a cross-modality module, which is based on the cross attention mechanism. Experimental results on our extended Minecraft Corpus dataset show that our model achieves state-of-the-art performance with a substantial improvement for the *collaborative building task*. We also provide new baselines for *learning to ask* task and the combination of these two tasks: the collaborative building and learning to ask tasks.

2 Related Work and Background

Dialogue Tasks. As virtual personal assistants have now penetrated the consumer market, with products such as Siri and Alexa, the research community has produced several works on *task-oriented dialogue tasks* such as: hotel booking, restaurant booking, movie recommendation, etc. (Budzianowski et al., 2018; Li et al., 2018; Rastogi et al., 2020; Wei et al., 2018; Wu et al., 2019; Heck et al., 2020). These task-oriented dialogues have been modelled as slot filling tasks. These tasks consist of correctly identifying and extracting information (slots) useful to solve the task. However, most of these slot filling tasks (Coope et al., 2020; Heck et al., 2020) are considered as semantic tagging or parsing of natural language and do not normally consider visual information. Moreover, these tasks focus only on two of the many components needed by conversational systems: the Natural Language Understanding (NLU) and Dialogue State Tracking (DST) ones (Budzianowski et al., 2018; Williams et al., 2014). Beside these task-oriented dialogue tasks, the research community has also focused on *instruction following dialogue tasks*, such as: target completion tasks (de Vries et al., 2017), object finding tasks (Roman et al., 2020), and navigation tasks (Thomason et al., 2020; De Vries et al., 2018). Narayan-Chen et al. (2019) proposed the Minecraft Corpus dataset, where the task consists in a cooperative asymmetric task involving an architect and a builder that have to build a target structure collaboratively. Jayannavar et al. (2020) then built a builder model to follow the sequential instructions from the architect.

Multi-Modal. Almost all instruction following dialogue tasks need to consider both contextual information and actions as well as the state of the world (Suhr and Artzi, 2018; Suhr et al., 2019; Chen et al., 2019; Lachmy et al., 2021), which remains a key challenge for instruction following dialogue tasks. In particular, the Vision-and-Dialog Navigation (VDN) task (Chen et al., 2019; Thomason et al., 2020; Roman et al., 2020; Zhu et al., 2021) where the question-answering dialogue and visual contexts are leveraged to facilitate navigation, has attracted increasing research attention. Other tasks, such as blocking movements tasks (Misra et al., 2017) and object finding tasks (Janner et al., 2018), also require the modelling of both contextual information in natural language as well as the world state representation to be solved.

Spatial Reasoning. Many instruction following dialogue tasks contain texts with spatial-temporal concepts (Misra et al., 2017; Janner et al., 2018; Tan and Bansal, 2018; Chen et al., 2019; Yang et al., 2020). Therefore, another challenge of an embodied agent is to follow instructions based on learning spatio-temporal linguistic concepts in natural language. For instance, the Minecraft Corpus dataset (Narayan-Chen et al., 2019) contains utterances with spatial relations, e.g., “go to the middle and place an orange block two spaces to the left”. Interpreting and grounding abstraction stated in natural language, such as spatial relations, has not been systematically studied and remains still challenging. Lachmy et al. (2021) proposed the HEXAGONS dataset. This dataset needs players to follow instructions with spatial relations to recreate target images.

Learning by Asking Questions. Determining whether to ask clarification questions and what to ask is critical for instruction followers to complete the tasks. Several recent studies have focused on learning a dialogue agent with the ability to interact with users by both responding to questions and by asking questions to accomplish their task interactively (Li et al., 2017; de Vries et al., 2017; Misra et al., 2018; Roman et al., 2020). For instance, de Vries et al. (2017) introduced a game to locate an unknown object via asking questions about objects in a given image. A decision-maker is introduced to learn when to ask questions by implicitly reasoning about the uncertainty of the agent. Different from earlier works (Kitaev and Klein, 2017; Suhr et al., 2019), recent works on VDN tasks propose agents that learn to ask a question when the certainty of the next action is low (Nguyen and Daumé III, 2019; Thomason et al., 2020; Roman et al., 2020; Chi et al., 2020). Roman et al. (2020) proposed a two models-based agent with a navigator model and a questioner model. The former model was responsible for moving towards the goal object, while the latter model was used to ask questions. Zhu et al. (2021) proposed an agent that learned to adaptively decide whether and what to communicate with users in order to acquire instructive information to help the navigation. However, compared to questions in navigation tasks (Roman et al., 2020; Thomason et al., 2020; Zhu et al., 2021; Nguyen and Daumé III, 2019) where questions are relatively simple and mostly relevant to where to go, the clarification questions in our extended Minecraft collaborative building task are more complex and challenging due to their diversity.

3 Dataset and Tasks

3.1 The Minecraft Dialogue Corpus

The Minecraft Dialogue Corpus (Narayan-Chen et al., 2019) is built upon a simulated block-world environment with dialogues between an architect and a builder. This consists of 509 human-human dialogues (15,926 utterances, 113,116 tokens) playing the role of an architect and a builder, and game logs for 150 target structures of varying complexity (min. 6 blocks, max. 68 blocks, avg. 23.5 blocks), For each target structure at least three dialogues are collected where each dialogue contains 30.7 utterances (22.5 architect utterances and 8.2 builder utterances) and 49.5 builder blocks movements on average.

The architect instructs about a target structure the builder to build it via a dialogue. Although the architect observes the builder operating in the world, only the builder can move blocks. The builder has access to an inventory of 120 blocks of six given colors that he or she can place or remove. The collaborative building task restricts the structures to a build region of size $11 \times 9 \times 11$, and contains 3709, 1331, and 1616 samples for training, validation, and test sets.

3.2 Builder Dialogue Annotation

Builders need to be able to decide their actions at any time point rather than only execute actions with the information about when to execute. Thus, we annotate all builders' utterances in the Minecraft Corpus dataset (Narayan-Chen et al., 2019) and categorize all 4,904 builder utterances into 8 utterance types. Each utterance falls into exactly one category. These categories are defined as follows: ***(1) Instruction-level Questions:*** used to request that the architect clarifies a given instruction or statement;***(2) Task-level Questions:*** used to re-

Table 1: The taxonomy of builders' utterances: We categorize them into eight types where instruction-level questions and task-level questions are both a sub-type of clarification questions. There are 4,904 builder utterances in total.

Catogory	Example	Amount	Percentage
Instruction-level Questions	1. What color? 2. Is it flat?	914	18.64%
Task-level Questions	1. What are we building? 2. What's next?	252	5.14%
Verification Questions	1. Like that or othwr way? 2. Is this the cross you wanted?	1021	20.82%
Greeting	1. Ready? 2. Hello!	808	16.48%
Suggestions	1. In the future we can call these donuts or something 2. No problem, if it's hard to describe we can just go step by step	59	1.23%
Display Understanding	1. No problem. 2. Knew what you meant	1296	26.43%
Status Update	1. I don't have enough green to continue. 2. I'll stay with this perspective	101	2.06%
Chit-Chat and others	1. I got my first job from Minecraft. 2. Oh, wwo, sorry!	453	9.24%

Table 2: Statistics of the extended Minecraft Dialogue Corpus: "Execution(Original)" represents that the builder should predict a sequence of building actions given the dialogue context and the world state in a sample; "Ask for clarifications" indicates that the builder should ask for more information in order to execute building actions; "Others" stands for remaining dialogue acts for the builder such as greetings, chit-chat, and display understanding.

	Train	Valid	Test
Execution (Original)	3709	1331	1616
Ask for clarifications	437	151	163
Others	837	267	366
Total	4983	1749	2145

quest the architect to give a description about the whole picture of the building task, e.g., asking for the next instruction or asking to describe how the target structure should look like; ***(3) Verification Questions:*** used to request to confirm that the previous action(s) were correct; ***(4) Greetings:*** used as a welcome message or to recognize the start of the mission and only occurs early in the dialogue; ***(5) Suggestions:*** used to provide suggestions; ***(6) Display Understanding:*** used to express whether a given instruction has been understood; ***(7) Status Update:*** used to describe the current status, e.g., tell the architect where they are, their current block stock status, or whether they have finished a given instruction. ***(8) Chit-chat and others:*** any other utterance not relevant to the completion of the task, including chit-chat, expressing gratification or apologies, etc.

Among these 8 utterance types, the *instruction-level questions* and the *task-level questions* are a sub-type of clarification questions used to further clarify instructions or the task itself when the information from the architect is not clear or ambiguous. Based on these annotations, we extend the original dataset (the first row in Table 2) with two other dialogue acts, ask for clarifications and others, as shown in the second and third row of Table 2. The 'Ask for clarifications' sample includes a dialogue context by a builder utterance labelled as *instruction-level questions* or *task-level questions*, while remaining dialogue contexts ended by builder utterances are considered as 'others'.

3.3 Task Definition

Let $\mathcal{H}$ be the set of all dialogue contexts, $\mathcal{W}$ the set of all world states, and $\mathcal{A}$ the set of all building actions, including placing and removing a block and a special *stop* action, which terminates the task. The action execution will update the world state via a transition function $T : \mathcal{W} \times \mathcal{A} \rightarrow \mathcal{W}$. Given a dialogue context $h \in \mathcal{H}$, a grid-based world state $w_n \in \mathcal{W}$, and the action history $\{a_1, \ldots, a_n\}, a_1, \ldots, a_n \in \mathcal{A}$, the target is to predict the action type: execution (*placement*, *removal*, or *stop*), ask (for clarifications or others), or stop. When the prediction of the action type is execution, also a sequence of actions should be output $\{a_1, \ldots, a_n\}, a_1, \ldots, a_n \in \mathcal{A}$, such that $w_{i+1} = T(w_i, a_i), w_1, \ldots, w_n \in \mathcal{W}$, a_n is the *stop* action and w_n contains the target structure.

4 Method

In this section we introduce the proposed builder model, as shown in Figure 2. The model comprises

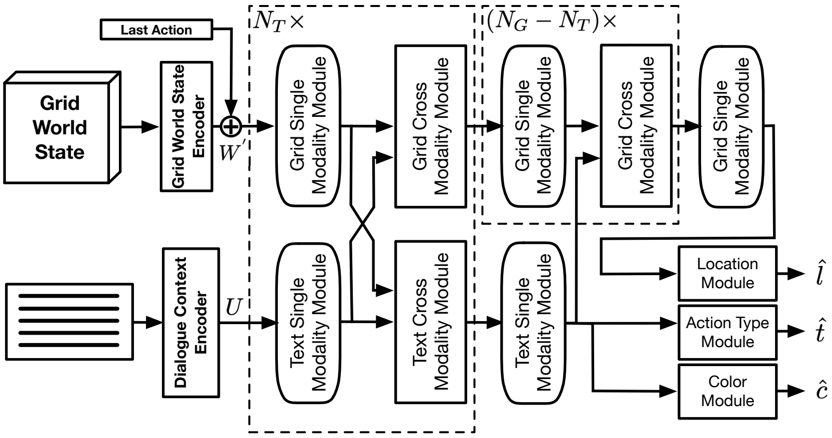

Figure 2: The model architecture. The $\oplus$ sign represents the concatenation operation. This illustration uses the plate notation. There are a total of N_T+1 text single modality modules, N_G+1 grid single modality modules, N_T text cross modality modules, and N_T grid cross modality modules. Arrows indicate the flow of information.

four major components: the *utterance encoder*, the *world state encoder*, the *fusion module*, and the *slot decoder*. The utterance encoder (in Sec. 4.1) and world state encoder (in Sec. 4.2) learn to represent the dialogue context and the world state. These encoded representations are then fed into the fusion module (in Sec. 4.3) that learns contextualized embeddings for the grid world and textual tokens through the single and cross modality modules. Finally, the learned world and text representations are mapped into the pre-defined slot-values in the slot decoder (in Sec. 4.4).

4.1 Dialogue Context Encoder

We add "architect" and "builder" annotations before each architect utterance A_t and each builder utterance B_t respectively. Then, the dialogue utterances are represented as

$$D_t = \text{“architect”}A_t \oplus \text{“builder”}B_t$$

at the turn t, where $\oplus$ is the operation of sequence concatenation.The entire dialogue context is defined as:

$$H = D_1 \oplus D_2 \oplus \cdots \oplus D_t \tag{1}$$

Given the dialogue context H, we truncate the tokens from the end of the dialogue context or pad them to a fixed length as inputs and then use the dialogue context encoder to encode utterance history into $U \in \mathbb{R}^{s\times d_w}$, where d_w is the dimension of the word embedding and s is the maximum number of tokens for a dialogue context. The dialogue context encoder can be word embeddings like Glove Pennington et al. (2014) or contextual word embeddings Devlin et al. (2019).

4.2 Grid World State Encoder

The world state is represented by a voxel-based grid. We first represent each grid state as a 7-dimensional one-hot vector that stands for empty status or one of 6 colors, yielding a 7×11×9×11 world state representation. Additionally, we truncate the action history to the last five ones, assign an integer weight in $1,\ldots,5$ and then include these weights as a separate input feature in each grid, resulting in a raw world state input of $W_0 \in \mathbb{R}^{8\times 11\times 9\times 11}$. We also represent the last action as an 11-dimensional vector a where the first two dimensions represent the placement or removal actions, the next six dimensions represent the color, and the last three dimensions represent the location of the last action.

The structure of the world state encoder is similar to Jayannavar et al. (2020)'s, i.g., consisting of k 3D-convolutional layers (f_1) with kernel size 3, stride 1 and padding 1, followed by a ReLU activation function. Between every successive pair of these layers there is a 1×1×1 3D-convolutional layer (f_2) with stride 1 and no padding followed by ReLU:

$$W_i = \mathrm{ReLU}(f_2^i(\mathrm{ReLU}(f_1^i(W_{i-1})))), \tag{2}$$
$$W_k = \mathrm{ReLU}(f_1^i(W_{k-1})), \tag{3}$$

where $i=1,2,\ldots,k-1$. $W_k \in \mathbb{R}^{d_c\times 11\times 9\times 11}$ is the learned world grid-based representation where d_c is the dimension of each grid representation. Then we concatenate the last action representation $a \in \mathbb{R}^{11}$ to each grid vectors in W_k and reshape them into $W' \in \mathbb{R}^{d_c'\times 1089}$, where $d_c' = d_c + 11$.

4.3 Fusion Module

The fusion module comprises four major components: two *single modality modules* and two *cross-modality modules*. The former modules are based on self-attention layers and the latter on cross-attention layers. These take as input the world state representation and dialogue history representation. Between every successive pair of grid single-modality modules or text single-modality modules there is a cross modality module. We take N_G and N_T layers for the grid cross modality module and the text cross modality module. We first revisit the definition and notations about the attention mechanism (Bahdanau et al., 2015) and then introduce how they are integrated into our single modality modules and cross-modality modules.

Attention Mechanism. Given a query vector x and a sequence of context vectors $\{y_j\}_{j=1}^K$, the attention mechanism first computes the matching score s_j between the query vector x and each context vector y_j. Then, the attention weights are calculated by normalizing the matching score: $a_j = \frac{exp(s_j)}{\sum_{j=1}^K exp(s_j)}$. The output of an attention layer is the attention weighted sum of the context vectors: $Attention(x, y_j) = \sum_j a_j \cdot y_j$. Particularly, the attention mechanism is called self-attention when the query vector itself is in the context vectors $\{y_j\}$. We use the multi-head attention following Devlin et al. (2019); Tan and Bansal (2019).

Single-Modality Module. Each layer in a single-modality module contains a self-attention sub-layer and a feed-forward sub-layer, where the feed-forward sub-layer is further composed of a linear transformation layer, a dropout layer and a normalization layer. We take $N_G + 1$ and $N_T + 1$ layers for the grid single-modality modules and the text single-modality modules respectively, interspersed with cross-modality module as shown in Figure 2. Since new blocks can only be feasibly placed if one of their faces touches the ground or another block in the Minecraft world, we add masks to all infeasible grids in the grid single-modality modules. For a set of text vectors $\{u_i^n\}_{i=1}^s$ and a set of grid vectors $\{w_j^m\}_{j=1}^{1089}$ as inputs of n-th text single-modality layer and m-th grid single-modality layer, where $n \in \{1, \dots, N_T + 1\}$ and $m \in \{1, \dots, N_G + 1\}$, we first feed them into two self attention sub-layers:

$$u_i^n = \text{SelfAttn}_u^n(u_i^n, \{u_i^n\}), \quad (4)$$
$$w_j^m = \text{SelfAttn}_w^m(w_j^m, \{w_j^m\}, mask) \quad (5)$$

Lastly, the outputs of self attention modules, u_i^n and w_j^m, are followed by feed-forward sub-layers to obtain $\hat{u}_i{}^n$ and $\hat{w}_j{}^m$.

Cross-Modality Module. Each layer in the cross-modality module consists of one cross-attention sub-layer and one feed-forward sub-layer, where the feed-forward sub-layers follow the same setting as the single-modality module. Given the outputs of n-th text single-modality layer, $\{\hat{u}_i{}^n\}_{i=1}^s$, and the m-th grid single-modality layer, $\{\hat{w}_j{}^m\}_{j=1}^{1089}$, as the query and context vectors, we pass them through cross-attention sub-layers, respectively:

$$\hat{u}_i{}^{n+1} = \text{CrossAttn}_u^n(\hat{u}_i{}^n, \{\hat{w}_j{}^m\}), \quad (6)$$
$$\hat{w}_j{}^{m+1} = \text{CrossAttn}_w^m(\hat{w}_j{}^m, \{\hat{u}_i{}^n\}), \quad (7)$$

The cross-attention sub-layer is used to exchange the information and align the entities between the two modalities in order to learn joint cross-modality representations. Then the output of the cross-attention sub-layer is processed by one feed-forward sub-layer to obtain $\{u_i^{n+1}\}_{i=1}^s$ and $\{w_j^{m+1}\}_{j=1}^{1089}$, which will be passed to the following singe-modality modules.

Finally, we obtain a set of word vectors, $\{\hat{u}_i{}^{N_T+1}\}_{i=1}^s$, and a set of grid vectors, $\{\hat{w}_j{}^{N_G+1}\}_{j=1}^{1089}$, that is, U^{N_T} and W^{N_G}. Since the value of N_G and N_T could be different, the modality with more layers would keep using the last single modality module's output of another modality as the input of its cross modality modules, as shown in the Figure 2.

4.4 Slot Decoder

The Slot Decoder contains three linear projection layers of trainable parameters, $M_L \in \mathbb{R}^{d_c'}$, $M_C \in \mathbb{R}^{6 \times d_w}$, $M_T \in \mathbb{R}^{d_a \times d_w}$ where d_a is the number of action types to predict. We compute the average of $U^{N_T} \in \mathbb{R}^{s \times d_w}$ alongside the s-dimension to obtain $u \in \mathbb{R}^{d_w}$. Then we compute location logits, color logits, and action type logits:

$$\hat{l} = \text{softmax}(M_L \cdot W^{N_G}), \quad (8)$$
$$\hat{c} = \text{softmax}(M_C \cdot u), \quad (9)$$
$$\hat{t} = \text{softmax}(M_T \cdot u), \quad (10)$$

where softmax functions are used to map the extracted information into $\hat{l} \in \mathbb{R}^{1089}$, $\hat{c} \in \mathbb{R}^6$, and $\hat{t} \in \mathbb{R}^{d_a}$.

5 Experiment, Results and Discussion

In this section, we first compare our model against the baseline for the collaborative building task where models only need to learn the instruction following task (in Sec. 5.1). Then, we train our model to learn when to ask and evaluate on our extended Minecraft Dialogue Corpus (in Sec. 5.2). Finally, we evaluate our model's ability on the combination of the two above-mentioned tasks (in the Sec. 5.3). All training details are reported in the Appendix. The software and data used to run these experiments are available at the following weblink: <anonymized>.

Table 3: Evaluation on the collaborative building task.

Model	Metric	Augmentation None	2x	4x	6x
BAP model	F1	19.7	19.5	21.2	20.8
	Recall	-	-	-	-
	Precision	-	-	-	-
Ours (GloVe)	F1	**35.0**	**36.5**	**37.8**	**39.4**
	Recall	28.3	30.1	31.4	33.4
	Precision	45.8	46.2	47.6	48.1
Ours (BERT)	F1	34.5	30.1	30.4	35.4
	Recall	26.7	23.6	23.4	27.9
	Precision	48.7	42.6	43.5	48.5

5.1 Collaborative Building Task

Settings. We first compare our model against the only baseline (Jayannavar et al., 2020), named BAP. Then, we conduct the experiments performing the same data augmentations as in BAP, where utterances are paraphrased, color substituted, and spatial transformation were used to augment the size and variety of the Minecraft Corpus Dataset. The basic train, valid, and test set contain 3709, 1331, and 1616 samples. All models are also evaluated with augmented training sizes [1]: 5,563 (indicated as 2x), 9,272 (4x), and 12,981 (6x) training samples. Additionally, we present the performance of two different dialogue context encoders: we use the pre-trained GloVe word embeddings with 300 dimensions (Pennington et al., 2014) as the initial word embeddings followed by a GRU(Chung et al., 2014) and contextual word embeddings using the pre-trained BERT base model (Devlin et al., 2019).

For the action type slot, we pre-define three potential values: *placement*, *removal*, and *stop*. The value of the location slot can be one of 1,089 candidate voxels and the value of the color slot can be one of six candidate colors. During training we minimize the sum of the cross entropy losses of the location slot, the color slot, and the action type slot. The F1 metric on the test set is used to evaluate model performance by comparing the model predictions against the action sequence performed by the human builder.

[1]We use the dataset released by the author of the baseline model, whose sizes are smaller than those reported in the paper.

Results. In Table 3 we present the results of our model and the baselines for the collaborative building task on the Minecraft Corpus Dataset. Experimental results show that our model outperforms the baseline model with a large margin. Meanwhile, results on the augmented dataset show that the advantage of the data augmentation is not obvious. The performance using contextualized word embeddings is poorer. This could be due to the size of the builder model with the BERT encoder which makes it more difficult to train.

5.2 Learning to Ask Questions

Settings. For the action type slot, we define three potential values: *execution*, *ask*, and *other*. 'Other' is used for all utterance types in Table 1 except for the instruction-level and task-level questions. In this experiment, the slots for location and color are not used. We test the pre-trained GloVe embeddings in the dialogue context encoder as described in Sec. 5.1. During the training, the cross entropy loss of the action type is minimized.

Table 4: Evaluation of the learning to ask task. Numbers in bold are the test accuracy for each action type.

Test Accuracy(%)		Prediction Execute	Ask	Other	Size
Oracle	Execution	**93.81**	4.33	1.86	1616
	Ask	22.09	**63.80**	14.11	163
	Others	35.79	36.89	**27.32**	366
Overall Test Acc		80.05			2145

Results. In Table 4, we present the results of our model. Although our model achieves around 80% overall test accuracy, the correct answers mainly come from the execution type while the model struggles with the ask and other types. These two types have in fact a joint test accuracy of 38.6%. Experimental results demonstrate that the difficulty of the learning to ask task and that there is still a large room for improvement.

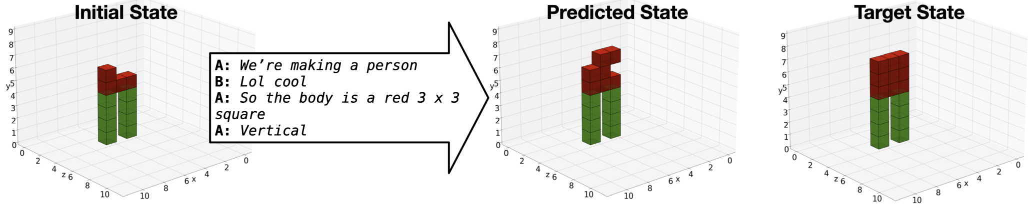

Figure 3: Case study of the collaborative building task in Sec. 5.1: *A* represents the architect and *B* the builder.

5.3 Joint Learning

Settings. For the action type slot, we pre-define five potential values: *placement*, *removal*, *stop*, *ask*, and *other*. The value of the location slot can be one of 1,089 candidate grid and the value of the color slot is one of 6 candidate colors. We still use pre-train GloVe embedding in the dialogue context encoder as described in Sec. 5.1. During the training we minimize the sum of the cross entropy losses of the location slot, the color slot, and the action type slot with weights equal to 0.1, 0.1, 0.8 for each slot type.

Table 5: Test accuracy of the joint task: Figure in bold of each row is the test accuracy for each action type.

Test Accuracy(%)		Prediction: Execution	Prediction: Ask	Prediction: Others
Oracle	Execution	**82.64**	8.64	8.72
	Ask	8.61	**60.93**	30.46
	Others	25.47	47.07	**31.46**
Overall Test Acc		72.26		

Table 6: The evaluation of the joint task.

	F1	Recall	Precison
Ours	28.4	20.9	43.9

Results. In Table 5, we present the results of our model's test accuracy for each action type. The model has an 82.6% test accuracy. However, if the execution of building actions is excluded, its joint test accuracy of ask and other action types is about 40.5%, indicating that deciding when to take the initiative remains challenging. In Table 6, we also report the results of recall rate, precision rate, and F1 score for the building task. Not surprisingly, the performance of our model drops slightly compared to those in Table 3, reflecting the difficulty of joint learning the collaborative building and the learning to ask tasks.

5.4 Case Study

Although our model can predict the actions more accurately than the baselines, for example our model can usually predict the color of the blocks correctly with about 60% test accuracy rate, it is still non-trivial for our model to predict the whole action sequence correctly. In Figure 3, the architect instructed the builder to build a 3x3 square and then our model generated only parts of the structure successfully.

The dataset noise makes the learning process more challenging: the builder action sequences are noisy due to, for example, the builder miss-clicking in the construction process (Narayan-Chen, 2020). Also, builder action sequences are often fragmented between utterances due to the frequent interruptions of the architect. In order to solve these issues a good model should be capable to learn better representations for higher-level abstractions in natural language like spatial relation concepts and be more robust to noisy actions. However, existing models including pre-trained ones (Devlin et al., 2019) fail to learn such representations for spatial reasoning, which translates into poor performance in these instruction following tasks.

6 Conclusion

In this paper, we extend the Minecraft Corpus dataset by labelling each builder utterances into eight types, in which two of them are relevant to asking clarification questions. This allows the builder models to learn to take the initiative in the instruction following tasks. Also, we have proposed a new model that achieves state-of-the-art performance on the Minecraft collaborative building task with a large improvement. Besides these contributions, we define the new learning to ask task used to learn to ask clarification questions, and new baseline models for this task and the joint one: the collaborative building and the learning to ask task.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. TOUCHDOWN: natural language navigation and spatial reasoning in visual street environments. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12538–12547. Computer Vision Foundation / IEEE.

Ta-Chung Chi, Minmin Shen, Mihail Eric, Seokhwan Kim, and Dilek Hakkani-Tür. 2020. Just ask: An interactive learning framework for vision and language navigation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2459–2466. AAAI Press.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Samuel Coope, Tyler Farghly, Daniela Gerz, Ivan Vulić, and Matthew Henderson. 2020. Span-ConveRT: Few-shot span extraction for dialog with pretrained conversational representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 107–121, Online. Association for Computational Linguistics.

Harm De Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. 2018. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*.

Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4466–4475. IEEE Computer Society.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. 2019. Minerl: A large-scale dataset of minecraft demonstrations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2442–2448. ijcai.org.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. TripPy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44, 1st virtual meeting. Association for Computational Linguistics.

Michael Janner, Karthik Narasimhan, and Regina Barzilay. 2018. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, 6:49–61.

Prashant Jayannavar, Anjali Narayan-Chen, and Julia Hockenmaier. 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, Online. Association for Computational Linguistics.

Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. 2016. The malmo platform for artificial intelligence experimentation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4246–4247. IJCAI/AAAI Press.

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*.

Nikita Kitaev and Dan Klein. 2017. Where is misty? interpreting spatial descriptors by modeling regions in space. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 157–166, Copenhagen, Denmark. Association for Computational Linguistics.

Royi Lachmy, Valentina Pyatkin, and Reut Tsarfaty. 2021. Draw me a flower: Grounding formal abstract structures stated in informal natural language. *arXiv preprint arXiv:2106.14321*.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017.

Learning through dialogue interactions by asking questions. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9748–9758.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015, Copenhagen, Denmark. Association for Computational Linguistics.

Ishan Misra, Ross B. Girshick, Rob Fergus, Martial Hebert, Abhinav Gupta, and Laurens van der Maaten. 2018. Learning by asking questions. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 11–20. IEEE Computer Society.

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

Anjali Yuan Narayan-Chen. 2020. *Towards collaborative dialogue in Minecraft*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Khanh Nguyen and Hal Daumé III. 2019. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 684–695, Hong Kong, China. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Schema-guided dialogue state tracking task at dstc8. *arXiv preprint arXiv:2002.01359*.

Homero Roman Roman, Yonatan Bisk, Jesse Thomason, Asli Celikyilmaz, and Jianfeng Gao. 2020. Rmm: A recursive mental model for dialog navigation. *arXiv preprint arXiv:2005.00728*.

Tianmin Shu, Caiming Xiong, and Richard Socher. 2018. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Alane Suhr and Yoav Artzi. 2018. Situated mapping of sequential instructions to actions with single-step reward observation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2072–2082, Melbourne, Australia. Association for Computational Linguistics.

Alane Suhr, Claudia Yan, Jack Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. 2019. Executing instructions in situated collaborative interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2119–2130, Hong Kong, China. Association for Computational Linguistics.

Arthur Szlam, Jonathan Gray, Kavya Srinet, Yacine Jernite, Armand Joulin, Gabriel Synnaeve, Douwe Kiela, Haonan Yu, Zhuoyuan Chen, Siddharth Goyal, et al. 2019. Why build an assistant in minecraft? *arXiv preprint arXiv:1907.09273*.

Hao Tan and Mohit Bansal. 2018. Source-target inference models for spatial instruction understanding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5504–5511. AAAI Press.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. 2020. Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406. PMLR.

Wei Wei, Quoc Le, Andrew Dai, and Jia Li. 2018. Air-Dialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854, Brussels, Belgium. Association for Computational Linguistics.

Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. Association for Computational Linguistics.

Tsung-Yen Yang, Andrew Lan, and Karthik Narasimhan. 2020. Robust and interpretable grounding of spatial references with relation networks. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1908–1923, Online. Association for Computational Linguistics.

Yi Zhu, Yue Weng, Fengda Zhu, Xiaodan Liang, Qixiang Ye, Yutong Lu, and Jianbin Jiao. 2021. Self-motivated communication agent for real-world vision-dialog navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1594–1603.

A Appendix

Training Details. We examine our model on the extend Minecraft Dialogue Corpus dataset. Our model's hyper-parameters are fixed for all three experiments in the Sec 5 as follows. The number of 3D-conv layers k is 3, the dimension of each grid representation d_c is 300, the number of layers of the grid cross-modality modules N_G is 4, and the number of layers of the text cross-modality modules N_T is 2. The max length of the dialogue context s is selected as 100 and the dropout rates are all set to 0.2. The number of heads for the attention mechanism in the text singe and cross modality modules is set to 2, while the number of heads is set to 1 for the attention mechanism in the grid singe and cross modality modules. The cross entropy loss from the location slot is not counted if the ground truth label of the action type is not 'placement' or 'removal', and the cross entropy loss from the color slot is not counted if the ground truth label of the action type is not 'placement'. For the experiment in the Sec 5.2 and 5.3, we randomly sample from 'Ask' and 'Others' sets in the training set to make training samples of different action types ('Execution', 'Ask', and 'Others') in the training set balanced. We train our model with cross entropy loss functions of all slots and a batch size of 50, using Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-6, β_1 = 0.9 and β_2 = 0.99. We train our model with 50 epochs and select the model with the highest F1 score on the valid set.