Neural *Abstructions*: Abstractions that Support Construction for Grounded Language Learning

Kaylee Burns *
Stanford University

Christopher D. Manning
Stanford University

Li Fei-Fei
Stanford University

Abstract

Although virtual agents are increasingly situated in environments where natural language is the most effective mode of interaction with humans, these exchanges are rarely used as an opportunity for learning. Leveraging language interactions effectively requires addressing limitations in the two most common approaches to language grounding: semantic parsers built on top of fixed object categories are precise but inflexible and end-to-end models are maximally expressive, but fickle and opaque. Our goal is to develop a system that balances the strengths of each approach so that users can teach agents new instructions that generalize broadly from a single example. We introduce the idea of neural *abstructions*: a set of constraints on the inference procedure of a label-conditioned generative model that can affect the meaning of the label in context. Starting from a core programming language that operates over abstructions, users can define increasingly complex mappings from natural language to actions. We show that with this method a user population is able to build a semantic parser for an open-ended house modification task in Minecraft. The semantic parser that results is both flexible and expressive: the percentage of utterances sourced from redefinitions increases steadily over the course of 191 total exchanges, achieving a final value of 28%. [1]

1 Introduction

As language learning agents become embodied in virtual and physical worlds alongside users, they are presented with the opportunity to curate rich data from humans for little to no cost. For example, when an agent misunderstands something, it can simply ask the human for input and guidance. Humans can explain unfamiliar concepts and describe procedures for accomplishing new tasks. Making these exchanges frictionless—and perhaps even beneficial to the user—incentivizes the human and agent to collaboratively construct rich mappings from natural language to actions or programs. However, the current machine learning toolkit needs stronger solutions for learning from complex instructions quickly and robustly. The goal of this work is to develop tools that allow users to define new instructions for a natural language system that can be adopted immediately and generally.

Wang et al. [65] first demonstrated how a user population can collaboratively create a more natural interface into a set of programs. The authors create a programming language for the task of building structures in voxel-based environment. Users then *naturalize* this programming language by providing pairings between natural language requests and programs. However, this model is very brittle as there is no room for users to specify distributions over object categories. For example, a reference to a tree

*Correspond to: kayburns@stanford.edu

[1] Link to code: https://github.com/kayburns/craftassist

1st CtrlGen: Controllable Generative Modeling in Language and Vision Workshop at NeurIPS 2021.

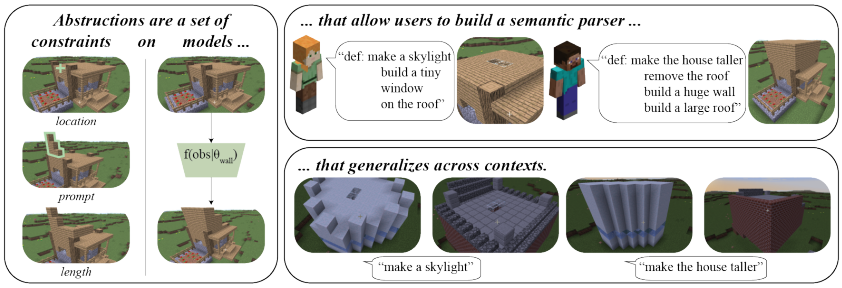

Figure 1: (left) Abstructions define a set of generative models and associated constraints so that they can be repurposed to represent arbitrarily complex reference objects. (right) Abstructions enable single-shot transfer of user-defined commands, enabling the emergence of a semantic parser that develops precision and flexibility in a target environment.

can only ever refer to one instantiation of "tree". Perhaps, we could enumerate all possible reference objects and develop generative or discriminative models to stand in for those references [32], but these approaches ignore a whole class of creative design problems where fixed object categories are not expressive enough to capture all potential user commands. Another alternative is to build end-to-end models that directly map language to a given modality with no modularity governing the interface between them [39, 37, 49], but these approaches suffer from the opacity and brittleness for which deep learning is infamous. Maintaining reliability helps users have a consistent and enjoyable experience with a language-equipped agent.

We introduce the concept of *abstructions*, a characterization of the generative models and user control inputs that enable the emergence of a flexible yet precise semantic parser from interactions with a population of users. Instead of beginning with a collection of fixed object categories, our parser binds language to a set of label-conditioned generative models. The parser is also made aware of constraints that affect the quality, performance, and meaning of these generative models, such as the initial location, build prompt, and length of generation. Adding these constraints to the generation process can change the meaning of a given model's output in context, as illustrated in Figure 1. Our blend of generative models and sets of constraints on them allows us to build a semantic parser that is sensitive to context while being able to adapt robustly to category knowledge given to us by users.

We evaluate the ability of a user population to develop a precise yet flexible semantic parser from abstructions in a creative building task in Minecraft using the CraftAssist [22] framework. Specifically, users are placed in a Minecraft session with our language agent and are instructed to make any desired modifications to a given house by talking directly with the agent. We show that over time, users rely increasingly on induced utterances and that the number of failed parses decreases. We demonstrate how newly defined commands can be applied to a wide variety of homes, immediately after being defined. Collectively, these results show that abstructions have the potential to create a language interface that blends precision and flexibility unlike past approaches to language grounding.

2 Related Work

Vision and Language. Static, supervised datasets have served as the foundation for bridging language to various modalities. A rich body of work focuses on the development of machine learning models that can successfully describe, reason about, and navigate within the visual world. In developing solutions [69, 18, 19, 3, 17, 46, 39, 57, 47, 50] for language and video descriptions [38, 54, 36] and visual question answering [6, 29, 15], researchers have identified critical limitations in the modern toolkit for multi-modal learning. Significant ablations don't result in significant performance drops [16, 34], vision is "ignored" in favor of language cues that are well correlated with prediction [51, 63], and models overfit to spurious correlations [13] that undermine generalization [1]. Many of these concerns have been addressed by curating balanced datasets [2, 31, 58, 21, 60], introducing auxiliary losses that counteract spurious correlations [48], and designing models with modularity in mind [5, 68, 40, 28]. But, disembodied from the environment in which the data was collected, agents are deprived of rich interactions to further structure their learning.

Structuring Other Modalities with Language. Language data can provide structure and insight for tasks that don't require it explicitly. Leveraging the compositionality of language by including it as an additional training input can improve classification performance in fine-grained [25] or few-shot [44] settings and can improve exploration when used to set goals intrinsically for reinforcement learning [14]. Language can also provide explanations for visual classification decisions, which gives practitioners insight into spurious correlations learned by visual models [26, 27, 43]. Most similar to our contribution is work that uses language feedback at training [20, 61, 56] or inference [52] time to guide and improve prediction results. However, instead of using language as an additional input at training or inference, we use natural language commands in combination with other user inputs to predict a set of constraints that affects the inference procedure of a generative model.

Learning Actions from Commands. The goal of our work is to learn a mapping from natural language instructions to actions. Some general frameworks for mapping instructions to actions include language-conditioned reinforcement learning [11, 8, 10], semantic parsers learned from supervision [23, 55], and a supervised mapping from instruction to action [42]. Tasks that focus on this problem include instruction guided navigation [4, 37, 42] and cooperative localization [24]. In contrast with these tasks, we focus on an open-ended creative design task like Kim et al. [33] where agents and humans can take actions collaboratively like Suhr et al. [59].

Language Learning from Interactions. Leveraging interactions with humans in the interest of improved learning outcomes has been studied in a variety of settings. Interactive dialogue can be used to bootstrap the capabilities of a semantic parser [7, 62], learn concepts from single examples [71], narrow-down classification decisions during inference [70] or training [67], or improve visual concept models in an online fashion [64]. Interactions need not be restricted to language: agents can also infer programs directly from examples [45]. Notably, Shah et al. [53] defines tasks for learning from human interactions in Minecraft. In our work, we use interactions to define new commands. Our goal is to naturalize a programming language through user-provided redefinitions, as described in Wang et al. [65]. We leverage a modernization of this approach presented in Karamcheti et al. [32]. However, unlike in these works, abstructions allow users to quickly define new object categories that are context sensitive, enabling naturalization in a creative editing task.

3 Learning to Ground Language with Abstructions

We study whether making abstructions, i.e., well-formed assumptions about label-conditioned generative models and their associated constraints, available to a semantic parsing framework can enable precision in specifying new commands without compromising the flexibility to work across contexts. To this end, we focus on an open-ended, creative house modification task in Minecraft. Users are able to specify any desired modifications to a given home through natural language requests to a virtual agent. When the virtual agent does not know or understand a given request, users have the opportunity to define new requests in terms of utterances the virtual agent already understands.

System Overview. At the start of interaction with users, the agent has access to a *core semantic-parsing framework* (Figure 2a), capable of building and destroying various house parts. We introduce *abstructions* (Figure 2b) into the `Build()` operation of the semantic parser by training label-conditioned generative models of next block placement and creating a set of user controls—number of blocks placed, build prompt, and location—that affect the quality and meaning of generated blocks. As users supply the agent with new instruction-program pairs, *online parser updates* (Figure 2c) are made through an alternative framework that relies on similarity search over sentence embeddings.

3.1 Motivating Examples

We motivate our design by describing two examples of user-defined instructions and how they are enabled by our system. These are also illustrated in Figure 1.

Instructions from Compositions. The most common form of user-defined instruction is a composition of known instructions. For example, "make the house taller" decomposes into the following steps: "remove the roof", "build a huge wall", and "build a large roof". Because we handle each

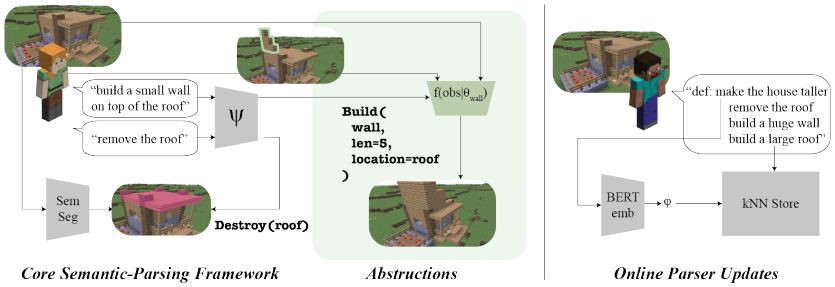

Figure 2: Overview of control flow when interacting with agent.

of these sub-instructions with generative models, as opposed to a single reference object, we can capture the ambiguity in each instruction and generalize better across homes. By contrast, if we were to compose two different objects defined under the framework presented by Wang et al. [65] or Karamcheti et al. [32], those objects could not adapt to a given scene because they have a fixed reference. Additionally, new object categories can also be created from compositions of commands. For example, to "build a second story" users may instruct the agent to "remove the roof, build a huge wall on top of the house, and build a roof on top of the house".

Instructions from Constraints. The benefit of abstructions is most evident in the second form of user-provided redefinitions. By placing constraints on the inference procedure of generative models, users can synthesize new object categories without requiring training a new model. Consider the command "build a skylight", which can be defined as "build a tiny window on top of the roof". The parser invokes the generative model for "window" to infer two block placements starting from a location on the roof. Another example is "build a rooftop patio", which could be constructed with the command "build a fence on top of the roof". Again, the constraints define the meaning of the primitive in context. In this case, the generative model for "fence" doesn't stand in for a fence or a subcategory of fence, but instead it is used for its likeness to a railing. Here, the program defines the meaning of the object and that meaning is passed directly as an instructional example to the agent.

3.2 Core Semantic-Parsing Framework

In the general case, we assume that we have access to a parser over a set of core instructions as well as a semantic segmentation and a sequential generation model. We build our virtual agent on top of the CraftAssist framework [22, 55]. This software provides the tooling for creating Minecraft sessions and the virtual agent, including the semantic parsing system and semantic segmentation module that make up the core parsing framework. These assumptions are highly practical, even in more natural settings. For example, a parser capable of decoding only the most straightforward core programs can be trained entirely through augmented data [41] or through a very small set of annotated utterances.

Core Parser. We use the parser introduced by Srinet et al. [55], which directly trains a BERT-based neural semantic parser on high-level Minecraft actions. Although the parser is capable of interpreting commands about a variety of high level tasks, we focus on `Destroy()` and `Build()` actions. The `Destroy()` action operates over a fixed collection of object categories, which are segmented from the scene as described in the next paragraph. For the `Build()` operation, the parser is also capable of interpreting qualitative or quantative descriptions of length and relative location.

Segmentation for `Destroy()` Operations. We use the semantic segmentation module provided by [22] to infer objects to destroy from the current house state. Unlike our `Build()` operation, we do not use user-specified constraints to steer the inference procedure of the `Destroy()` action, although this would be an interesting direction for future work.

3.3 Abstractions for `Build()` Operations

We replace the `Build()` action that is native to the CraftAssist framework with abstractions: a set of constraints on the inference procedure of a label-conditioned generative model that can affect the quality of generation and meaning of the primitive in context.

Although we describe a specific instantiation of abstractions, any generative model and set of constraints could be integrated into a semantic parser for the same purpose. In the general case, we assume that we have access to a parser, ψ, that infers constraints and labels, c, from utterances, u. The inferred label indexes into a library of fixed parameters, θ_c, which condition the generator. The user can also provide direct interventions to guide the generation process. Each of these components is visualized in Figure 2.

In our setting, we use a sequential block placement model as our generative model and a neural semantic parser trained on high-level Minecraft actions for ψ. However, any set of generative models that can support these constraints and that sufficiently cover the set of visual primitive concepts for a given task could be used as abstractions. Bau et al. [9] is one compelling example in the domain of image editing. Our interventions take the form of prompts: users may optionally supply the agent with a structure to seed the generation process. Finally, we only handle constraints on location and length.

Label-Conditioned Generative Models. To generate label-conditioned block placements, we adapt the VoxelCNN model presented in Chen et al. [12]. Given a 3D patch of a scene with block type information and a global view with occupancy information, VoxelCNN predicts next block type and placement. The original model was trained to generate complete houses on a dataset of 2,500 homes. We use voxel-level semantic segmentation labels to fine-tune VoxelCNN models for the labels: balcony, bed, bookcase, ceiling, column, deck, door, fence, floor, foundation, garden, grass, ground, ladder, lights, patio, pillar, porch, railing, roof, stair, torch, walkway, wall, window, and yard.

Our fine-tuning runs for an additional 4 epochs and for each class we select the model with the best performance on the validation set. We use the same train-val split as Chen et al. [12], but save 50% of the validation set as our test set. Averaging across categories, we achieve a top-10 accuracy of 66.0% and average 7.50 consecutively correct blocks. Performance by category as well as hyperparameters and compute resources can be found in Section A.1. For the classes window and bed, we overwrite block type prediction so that the agent gives predictable results. Window is hard-coded to predict glass blocks and bed is hard-coded to predict bed blocks.

For our setting, a sequential model is a very beneficial design choice. Users are able to provide interventions that strongly influence the outputs of the model so that concepts can be reused or remixed. Users can also prevent compounding errors by calling the same generative model multiple times. For example, if the direction of a wall is ambiguous, a user can instruct the `Build()` action to be called several times, allowing for them to intervene upon error. Our selection of constraints are motivated by these intervention opportunities.

Constraints and User Controls. We allow users to steer the inference procedure by providing constraints on location and length and intervening to supply a prompt. Location and length can be specified through natural language. If no location is specified, the agent asks users to specify a hint for the generation process. At this time, the user can provide location and prompt suggestions.

- *Location* is the coordinate at which the generative procedure will start. If a user specifies a location in their instruction, such as "on top of the roof", the coordinate location will be resolved heuristically using tools from the original CraftAssist framework. Otherwise, the starting coordinate is the direction of the user's cursor is projected onto the nearest house block.
- *Length* is the number of voxels sampled from the generative model. We map qualitative descriptions of size to block types: tiny is 2 blocks, small is 5 blocks, large is 50 blocks, huge is 100, and the default is 20 blocks. Note that this mapping is limited as qualitative descriptions of size don't adapt to category or to house size. It does however give users much more predictable results when specifying commands.
- *Prompt* is a block structure that users can optionally provide when asked for a hint. Depending on the primitive that is being invoked, this could affect the block type predicted or the

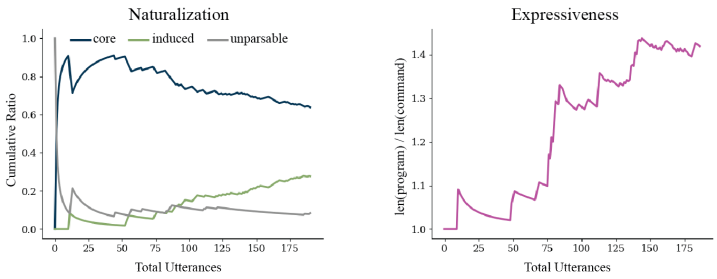

Figure 3: *Left*. Expressiveness is measured as the sum of core commands (i.e., build or destroy a reference object) and constraints each utterance refers to. *Right*. As users define more commands, users rely increasingly on induced utterances to achieve their build goals.

final structure shape. The system does not memorize or retain information about prompts, but this would be an interesting direction for future work.

3.4 Parser Updates Online

Users are able to define new commands with the following syntax:

```
def:  new command; sub command 1; ...; sub command N
```

Below we denote the `new command` as u. Upon receipt of a user-provided definition, we retrieve embeddings for each token of u. Denoting m as the sentence length, we have:

$$\phi_{1:m} = \text{BERT}(u_{1:m}) \tag{1}$$

Our BERT embeddings are provided by the HuggingFace Transformers [66] library. To achieve a single sentence representation, ϕ, we apply an aggregation function across the token features:

$$\phi = \sigma(\phi_{1:m}) \tag{2}$$

We use averaging as our choice for σ. This method draws inspiration from Karamcheti et al. [32], but does not utilize additions like added aggregation layers or lifted utterances as in our setting, changing a reference object can substantially change the inferred program.

When new requests are received, the aggregated features of the request are cross referenced with a nearest neighbor store before a parse is attempted. We use Facebook AI Similarity Search [30] as our nearest neighbor store. For the scale of data we collect in our experiments, training a layer on top of features for improved embedding quality is ineffective, so this essentially acts as a dictionary lookup.

4 Experiments

We show that over time, users rely increasingly on induced utterances and that the average utterance becomes more expressive. We also evaluate the generalization performance of user-defined commands with user surveys and qualitative examples.

4.1 Experiment Design

Our experiment ran in two phases: one qualifying task that introduced users to the agent's capabilities and limitations, and a creative build period where users could define new instructions, shared across the entire user population. All of the data we present is sourced from the creative build period, which was designed as follows. In each session, users enter a Minecraft server with the agent three separate times and interact with a total of two different homes. In the first instance, the users have the opportunity to test out any modification requests or new commands on the first home. In the second instance, the users apply their desired modifications and new command definitions to the same home. All of these modifications are specified through dialogue exchanges with the agent. Users are also welcome to "clean up" the agent's work, but many users chose not to clean up the agent's block placements. By creating two separate sessions with the same house, we allow users to explore the

capabilities and weaknesses of the bot. In the third session, we instruct users to replicate all of their modifications on a new, second home so that we can evaluate how well their re-definitions transfer.

During the creative build period, we explicitly told users that our goal was to teach the agent new, expressive instructions. We also provided videos describing tips to get good results out of the generative models and ideas about types of commands to define. The full set of instructions given to users in both the qualifying task and creative build period as well as further design details are provided in the appendix.

We hosted our experiments on Amazon Mechanical Turk and handled HIT management through the EasyTurk wrapper [35]. Compute details for the AMT experiments are provided in Section A.2. Of the 11 users who attempted the qualifying task, 8 passed and 4 chose to continue with the creative task. All users were paid twelve dollars per hour, with bonuses reaching fifteen dollars per hour. Prior to launching the the Minecraft server, users were notified that the data from their interactions will be collected and were instructed not to share any personally identifying information. We also removed their Minecraft username from the data we provide.

4.2 Naturalization

To evaluate whether naturalization still takes place when we introduce abstructions, we classify every dialogue exchange that results in a `Build()` or `Destroy()` action (i.e., not conversational exchanges like "hello") as 'core', 'induced', or 'unparsable'. Core utterances are actions that the agent could complete successfully with the core parser alone. Induced utterances are commands that were defined by the user population. Unparsable utterances are commands that the agent could not complete successfully. Examples include syntax or reference objects with which the agent is unfamiliar. The agent can also fail to parse successfully because of computational issues unrelated to its abilities, such as the segmentation model not inferring objects from the scene quickly enough. These are still counted as unparsable. We only consider dialogue exchanges from the second and third sessions of the creative build task. Note, we explicitly ask the users to test their redefinitions in a new environment. However, we see a similar result for dialogue exchanges in the second session only. Please see Section A.3 of the appendix for these results. We also treat new command definitions at the time that they're provided as the sequence of commands that define it. For example, we count "def: make the house taller; remove the roof; build a huge wall; build a large roof" as three core utterances, as opposed to one induced utterance.

Figure 3 shows the cumulative ratio of parsable, unparsable, and induced utterances over the course of all user dialogue exchanges. Similar to Wang et al. [65], we see a steady increase in the proportion of utterances that come from user-induced commands. However, we do not see the proportion of induced utterances overtake core utterances. We suspect that this is because the core actions we provide cover more modifications of interest within our setting. Unlike in Wang et al. [65], the users do not need to start from single block placements. Over the entire course of naturalization, 27.7% of utterances were induced.

4.3 Expressiveness

We define expressiveness as the length of the "program", i.e., user-provided redefinitons, divided by the length of the utterance that maps to it. For a core utterance, the expressiveness is one because the "program" is simply the original command. The expressiveness of an induced command is the total number of words or tokens in the commands that are specified in the definition. For example, "build a skylight" has 3 words and is defined as 'build a tiny window on the roof", which has 7 words. So, the expressiveness of "build a skylight" is 2.33. As with naturalization, we compute expressiveness for the second and third sessions of the creative build task. We again treat new command definitions as the sequence of commands in the definition.

Figure 3 shows that expressiveness increases over the course of the experiment, achieving an average expressiveness of 1.42 by the end of our naturalization experiment. Some users define commands with expressiveness below one. For example, one user defined the command "build an awning on house" as "build a roof", leading to an expressiveness score of .6. Presumably these redefinitions still have value to users or are otherwise more natural because of the prompts they are able to provide. Cases such as this could explain the dips in expressiveness across training. It also indicates that this

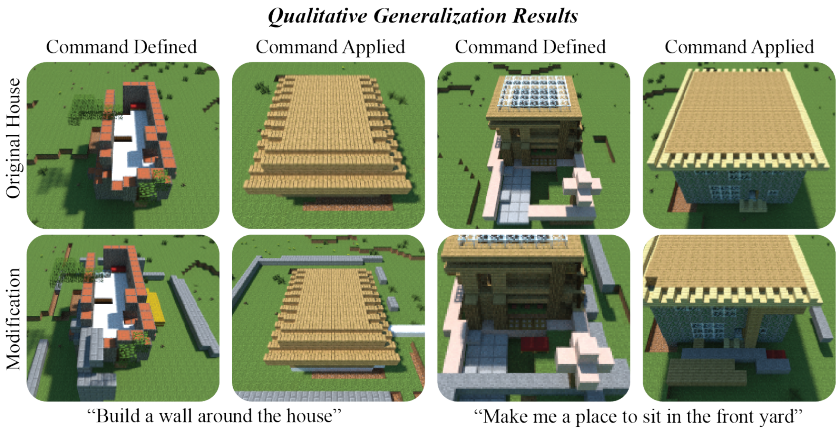

Figure 4: Examples of home modifications generated from user commands. The top row shows the original, unmodified home and the bottom row shows the result after the modification is applied. The first and third columns are examples where the user defined the command. The second and fourth column show the command applied to a new house.

make me a balcony: build a fence on top of roof
make me a second floor: build a wall; build a wall; build a wall; build a wall; build a roof
build a pool: build a wall; build a wall; build a wall; build a wall; build a floor
build a fenced yard in front of the house: build a large yard; build a fence
build a fenced walkway in front of the house: build a walkway; build a fence; build a fence
extend this house: remove wall; build a wall; build a wall; build a wall; build a wall; build a roof
build a rooftop pool: build a pool on top of roof
build bedroom with a view: make me a balcony; make me a place to sit down
build a raised bed: build tiny wall; build tiny wall to right; build a bed on top of wall
build a rooftop garden: build a roof; build a fence; build a garden
build me a second floor with a skylight: make me a second floor; build a skylight
build a porch: build a porch in front of the house
build a fenced patio in front of the house: build a patio; build a fence
build a large skylight: build a large window on roof
make me a rooftop lounge: make a fence on the roof; make me a place to sit down
destroy all: remove wall; remove house; remove floor; remove roof; remove window; remove foundation; remove bed; remove door
build a guard rail: build a tiny wall; build a tiny fence on top of tiny wall

Figure 5: Examples of user-provided redefinitions.

way of computing expressiveness, does not fully capture or describe the benefits that redefinitions provide to users.

4.4 Generalization across Diverse Homes

We use user surveys and qualitative examples of command transfers to understand how well abstructions enable redefinitions across the second and third session of the creative build task. Upon completing the task, 100% of users agreed that the command successfully transferred. Qualitative results revealed strong variance in the performance of generalized commands. The discrepancy between user ratings of generalization performance and visualizations of transfer performance are likely due to ambiguity in the definition of a successful transfer.

In Figure 4, we show how well two commands generalize across homes. The command "build a wall around the house" is defined as "build a wall; build a wall; build a wall; build a wall". The agent successfully builds walls around both homes. The command "make me a place to sit in the front yard" is defined as "make me a place to sit in the front yard: make a fence around the house; make me a place to sit down". Both of these are themselves induced commands. In this case, the agent makes a somewhat enclosed space and places two bed blocks, which are the small red blocks in front of the home, as a place to sit down.

4.5 Examples of Commands Defined

We visualize a selection of the 38 total commands supplied by users in Figure 5. These show that users significantly borrowed from other user's commands. These examples also illustrate some weaknesses of our design. Not all commands generalize to different forms of homes; "destroy all"

only works on homes with that exact list of objects. Users also relied heavily on the ability to specify location hints from their cursor, which is why many commands don't include qualitative descriptions of location.

5 Conclusion

In this work, we present the idea of neural *abstructions*: a set of abstractions around generative models and associated constraints that make it easier for users to develop precise commands that generalize across contexts. Our results show that abstructions have the potential to bring naturalization frameworks to a broader set of creative build tasks than was initially shown in Wang et al. [65]. Due to cost constraints, we could not run our experiments on a larger user population, but wish to in future work.

References

[1] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1955–1960, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1203. URL https://www.aclweb.org/anthology/D16-1203.

[2] Aishwarya Agrawal, Aniruddha Kembhavi, Dhruv Batra, and Devi Parikh. C-VQA: A compositional split of the visual question answering (VQA) v1.0 dataset. *ArXiv*, abs/1704.08243, 2017.

[3] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[4] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, I. Reid, Stephen Gould, and A. V. Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[5] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48, 2016.

[6] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015.

[7] Yoav Artzi and Luke Zettlemoyer. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 421–432, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D11-1039.

[8] Dzmitry Bahdanau, Felix Hill, J. Leike, Edward Hughes, Seyedarian Hosseini, P. Kohli, and Edward Grefenstette. Learning to understand goal specifications by modelling reward. In *ICLR*, 2019.

[9] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1907375117. URL https://www.pnas.org/content/early/2020/08/31/1907375117.

[10] Valts Blukis, Yannick Terme, Eyvind Niklasson, Ross A. Knepper, and Yoav Artzi. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *CoRL*, 2019.

[11] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*, 2017.

[12] Zhuoyuan Chen, Demi Guo, Tong Xiao, Saining Xie, Xinlei Chen, Haonan Yu, Jonathan Gray, Kavya Srinet, Haoqi Fan, Jerry Ma, Charles R Qi, Shubham Tulsiani, Arthur Szlam, and C. Lawrence Zitnick. Order-aware generative modeling using the 3d-craft dataset. In *ICCV*, 2019.

[13] Volkan Cirik, Louis-Philippe Morency, and Taylor Berg-Kirkpatrick. Visual referring expression recognition: What do systems actually learn? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 781–787, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2123. URL `https://www.aclweb.org/anthology/N18-2123`.

[14] Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, F Peter Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. *NeurIPS 2020*, 2020.

[15] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[16] J. Devlin, Saurabh Gupta, Ross B. Girshick, Margaret Mitchell, and C. L. Zitnick. Exploring nearest neighbor approaches for image captioning. *ArXiv*, abs/1505.04467, 2015.

[17] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[18] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1044. URL `https://www.aclweb.org/anthology/D16-1044`.

[19] Lianli Gao, Zhao Guo, Hanwang Zhang, Xing Xu, and Heng Tao Shen. Video captioning with attention-based lstm and semantic consistency. *IEEE Transactions on Multimedia*, 19(9): 2045–2055, 2017. doi: 10.1109/TMM.2017.2729019.

[20] Prasoon Goyal, S. Niekum, and R. Mooney. Using natural language for reward shaping in reinforcement learning. *IJCAI*, abs/1903.02020, 2019.

[21] Yash Goyal, Tejas Khot, Aishwarya Agrawal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *International Journal of Computer Vision*, 127:398–414, 2018.

[22] Jonathan Gray, Kavya Srinet, Yacine Jernite, Haonan Yu, Zhuoyuan Chen, Demi Guo, Siddharth Goyal, C. Lawrence Zitnick, and Arthur Szlam. Craftassist: A framework for dialogue-enabled interactive agents, 2019.

[23] Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1051–1062, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1097. URL `https://www.aclweb.org/anthology/P17-1097`.

[24] Meera Hahn, Jacob Krantz, Dhruv Batra, Devi Parikh, James M. Rehg, Stefan Lee, and Peter Anderson. Where are you? localization from embodied dialog. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[25] Xiangteng He and Yuxin Peng. Fine-grained image classification via combining vision and language. In *CVPR*, pages 7332–7340, 07 2017. doi: 10.1109/CVPR.2017.775.

[26] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[27] Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Grounding visual explanations. In *ECCV*, 2018.

[28] Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. Explainable neural computation via stack neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[29] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[31] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

[32] Siddharth Karamcheti, Dorsa Sadigh, and Percy Liang. Learning adaptive language interfaces through decomposition. *arXiv preprint arXiv:2010.05190*, 2020.

[33] Jin-Hwa Kim, Nikita Kitaev, Xinlei Chen, Marcus Rohrbach, Byoung-Tak Zhang, Yuandong Tian, Dhruv Batra, and Devi Parikh. CoDraw: Collaborative drawing as a testbed for grounded goal-driven communication. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6495–6513, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1651. URL `https://www.aclweb.org/anthology/P19-1651`.

[34] Noriyuki Kojima, Hadar Averbuch-Elor, Alexander M. Rush, and Yoav Artzi. What is learned in visually grounded neural syntax acquisition. In *ACL*, 2020.

[35] Ranjay Krishna. Easyturk: A wrapper for custom amt tasks. `https://github.com/ranjaykrishna/easyturk`, 2019.

[36] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *International Conference on Computer Vision (ICCV)*, 2017.

[37] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2020.

[38] Tsung-Yi Lin, M. Maire, Serge J. Belongie, James Hays, P. Perona, D. Ramanan, Piotr Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[39] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019.

[40] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJgMlhRctm`.

[41] Alana Marzoev, S. Madden, M. Kaashoek, Michael J. Cafarella, and Jacob Andreas. Unnatural language processing: Bridging the gap between synthetic and natural language data. *ArXiv*, abs/2004.13645, 2020.

[42] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1287. URL https://www.aclweb.org/anthology/D18-1287.

[43] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *ArXiv*, abs/2006.14032, 2020.

[44] Jesse Mu, Percy Liang, and Noah Goodman. Shaping visual representations with language for few-shot classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[45] Maxwell Nye, Yewen Pu, Matthew Bowers, Jacob Andreas, J. Tenenbaum, and Armando Solar-Lezama. Representing partial programs with blended abstract semantics. *ArXiv*, abs/2012.12964, 2020.

[46] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[47] Alec Radford, Ilya Sutskever, Jong Wook Kim, Gretchen Krueger, and Sandhini Agarwal. Clip: Connecting text and images, 2021.

[48] S. Ramakrishnan, Aishwarya Agrawal, and Stefan Lee. Overcoming language priors in visual question answering with adversarial regularization. *ArXiv*, abs/1810.03649, 2018.

[49] A. Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *ArXiv*, abs/2102.12092, 2021.

[50] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[51] Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. Object hallucination in image captioning. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[52] Christian Rupprecht, Iro Laina, Nassir Navab, Gregory D. Harger, and Federico Tombari. Guide me: Interacting with deep networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[53] Rohin Shah, Cody Wild, Steven H. Wang, Neel Alex, Brandon Houghton, William Guss, Sharada Mohanty, Anssi Kanervisto, Stephanie Milani, Nicholay Topin, Pieter Abbeel, Stuart Russell, and Anca Dragan. NeurIPS 2021 competition proposal: The MineRL BASALT competition on learning from human feedback. *NeurIPS Competition Track*, 2021.

[54] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*, 2018.

[55] Kavya Srinet, Yacine Jernite, Jonathan Gray, and Arthur Szlam. CraftAssist instruction parsing: Semantic parsing for a voxel-world assistant. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4693–4714, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.427. URL https://www.aclweb.org/anthology/2020.acl-main.427.

[56] Shashank Srivastava, Igor Labutov, and Tom Mitchell. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1527–1536, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1161. URL https://www.aclweb.org/anthology/D17-1161.

[57] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SygXPaEYvH.

[58] Alane Suhr, M. Lewis, James Yeh, and Yoav Artzi. A corpus of natural language for visual reasoning. In *ACL*, 2017.

[59] Alane Suhr, Claudia Yan, Jack Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. Executing instructions in situated collaborative interactions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019.

[60] Alane Suhr, Stephanie Zhou, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. *ArXiv*, abs/1811.00491, 2019.

[61] Theodore Sumers, Mark Ho, Robert Hawkins, Karthik Narasimhan, and Thomas Griffiths. Learning rewards from linguistic feedback. In *AAAI*, 2021.

[62] Jesse Thomason, Shiqi Zhang, R. Mooney, and P. Stone. Learning to interpret natural language commands through human-robot dialog. In *IJCAI*, 2015.

[63] Jesse Thomason, Daniel Gordon, and Yonatan Bisk. Shifting the baseline: Single modality performance on visual navigation & QA. 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 1977–1983, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1197. URL https://www.aclweb.org/anthology/N19-1197.

[64] Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin Hart, Peter Stone, and Raymond J. Mooney. Improving grounded natural language understanding through human-robot dialog. In *International Conference on Robotics and Automation (ICRA)*, May 2019. doi: 10.1109/ICRA.2019.8794287.

[65] Sida I. Wang, Samuel Ginn, Percy Liang, and Christopher D. Manning. Naturalizing a programming language via interactive learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 929–938, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1086. URL https://www.aclweb.org/anthology/P17-1086.

[66] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[67] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Visual curiosity: Learning to ask questions to learn visual recognition. In *Conference on Robot Learning (CORL)*, 2018.

[68] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[69] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[70] Lili Yu, Howard Chen, Sida I. Wang, Tao Lei, and Yoav Artzi. Interactive classification by asking informative questions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, July 2020.

[71] H. Zhang, Haonan Yu, and W. Xu. Interactive language acquisition with one-shot visual concept learning through a conversational game. In *ACL*, 2018.

A Appendix

A.1 Generative Model Details

All models are trained on a TeslaK40c through an internal cluster. We compared models across the following hyperparameters: with and without block embeddings, on a feature dimension size of 16, 32, and 64, on placement histories of size 1 and 3, with learning rates of .1 and .01, and on batch sizes of 32 and 64. Our final model uses block embeddings and has a feature dimension size of 32, history length of 3, learning rate of .1, and batch size of 64.

Our model's block placement prediction performance across categories is depicted in Table 1. Accuracy at 10 denotes the accuracy of block placements within the first 10 predictions of the model. CCA Average is the average consecutively correct block placements averaged across home completion amounts of 10%, 25%, 50%, 75%, and 90%. These metrics are adopted from Chen et al. [12].

Table 1: Fine-tuned VoxelCNN performance across object categories

label	Acc@10	CCA Avg.
floor	0.83	11.28
roof	0.84	10.82
foundation	0.76	10.41
wall	0.78	11.02
walkway	0.66	9.56
ceiling	0.81	10.07
balcony	0.73	8.86
stairs	0.41	8.36
patio	0.79	8.10
porch	0.71	8.31
deck	0.71	8.31
pillar	0.65	7.86
window	0.84	8.08
lights	0.42	7.31
column	0.59	7.17
door	0.51	6.29
ground	0.64	7.25
torch	0.30	6.79
railing	0.79	5.56
fence	0.79	5.45
grass	0.62	4.62
bookcase	0.63	4.07
garden	0.51	3.43
yard	0.51	0.95

A.2 Naturalization experiment

Our data collection occurred in two stages: we hosted a qualifying task, during which users were instructed to follow a tutorial video to familiarize themselves with the agent, and the main experiment, which was an open ended house modification task. For each task, we walked users through instructions on Amazon Mechanical Turk (AMT) and then directed them to a website, which launched a Minecraft server for them to connect to. The AMT instructions and server instructions for the qualifier are in Figures 6 and 7. The AMT instructions and server instructions for the main experiment are in Figures 8 and 9.

We pre-populate our nearest-neighbor store with the commands defined in the qualifying task and in the tutorial videos. This includes "make the house taller", "build a skylight", and "make me a place to sit down". Each house is randomly sampled from the test split we use for training the label-conditioned generative models. We filter house candidates from the test split based on dimension in voxels, so that the size terms apply well across different homes. We also remove homes with

blocks that behave atypically, such as lava or water. This leaves us with 23 total homes from which to sample.

We hosted the Minecraft server and agent on ECS instances. A new task is run on each launch with a memory size of 8192 MiB and 4096 CPU units.

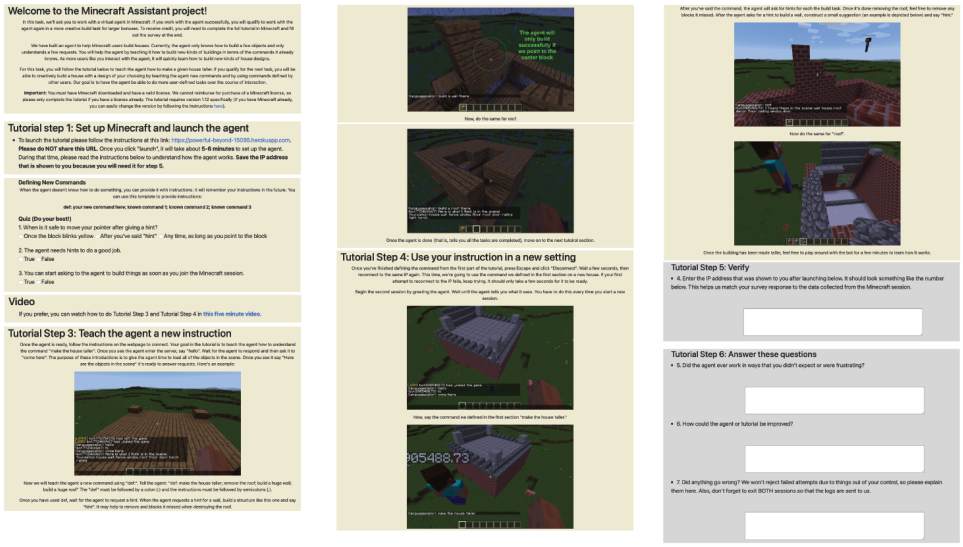

Figure 6: AMT qualification task description.

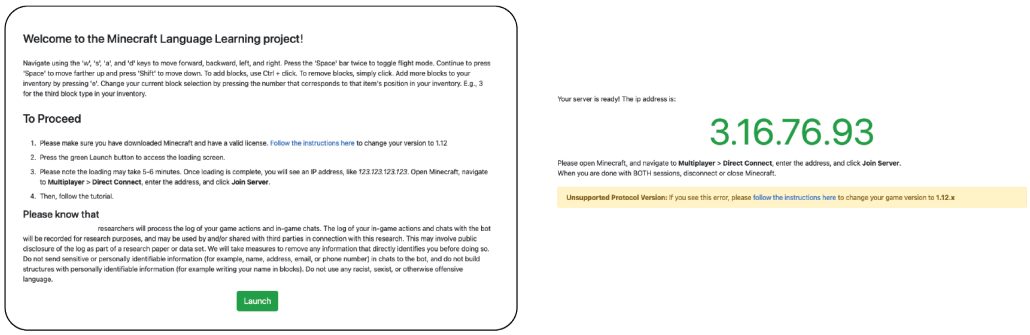

Figure 7: Server website for qualification task.

A.3 Naturalization and Expressiveness

To verify that the induced commands were not limited to the third session, where users are explicitly asked to repeat defined commands, we plot naturalization and expressiveness results from just the second session in Figure 10.

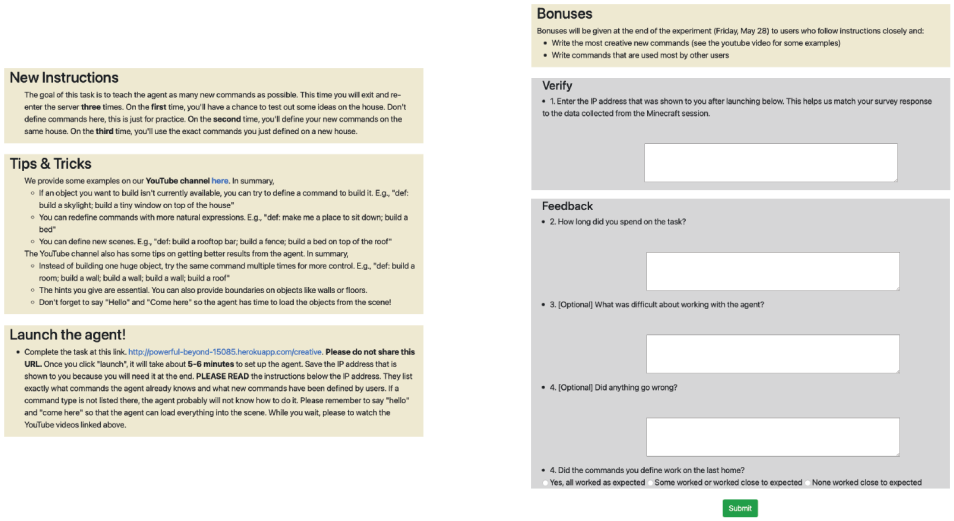

Figure 8: AMT main experiment task description.

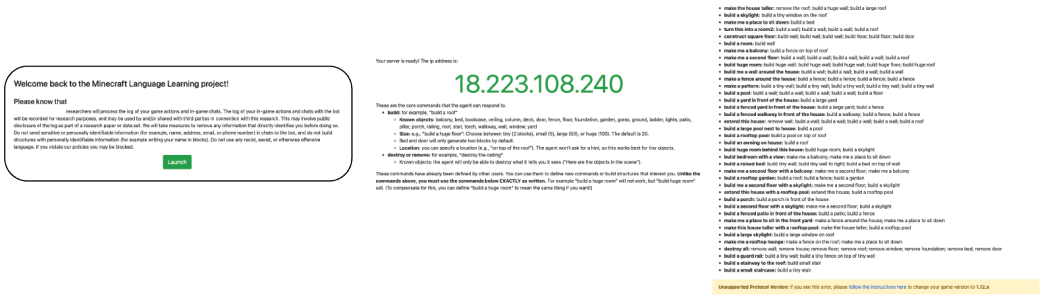

Figure 9: Server website for main experiment.

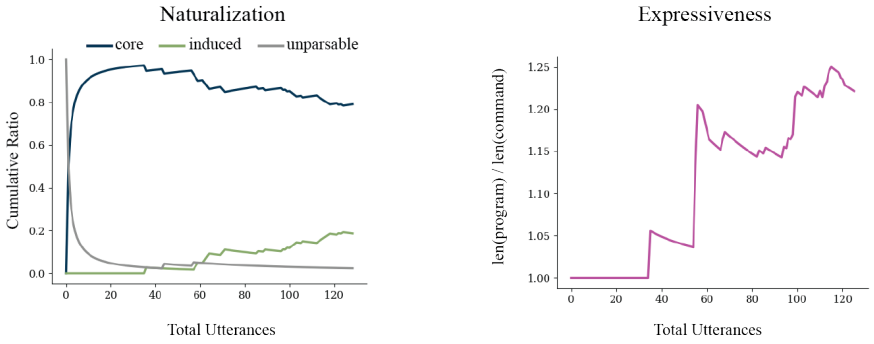

Figure 10: Naturalization and expressiveness results over only the second session, where users are not explicitly asked to repeat defined commands. When classifying utterance type, we use each sub-command in a command definition.