# CB2: Collaborative Natural Language Interaction Research Platform

**Jacob Sharf, Mustafa Omer Gul,** and **Yoav Artzi**
Department of Computer Science and Cornell Tech, Cornell University
jacobsharf@gmail.com {momergul, yoav}@cs.cornell.edu

**Abstract**

CB2 is a multi-agent platform to study collaborative natural language interaction in a grounded task-oriented scenario. It includes a 3D game environment, a backend server designed to serve trained models to human agents, and various tools and processes to enable scalable studies. We deploy CB2 at https://cb2.ai as a system demonstration with a learned instruction following model.

## 1 Introduction

Collaborative grounded natural language interactions involve multiple agents, either human or machine, working together to complete tasks while coordinating using natural language. A key obstacle in studying such scenarios is building the research interaction platform, a significant design and engineering undertaking. This requires building and designing the interaction environment, the task the agents collaborate on, an interface for both machine learning models and human agents, and a process to onboard human agents. Each aspect dramatically influences the interaction and language elicited, and is critical to get right.

We introduce CB2, a platform for the study of collaborative grounded natural language interaction, and demonstrate its use through the deployment of a learned collaborative natural language agent. CB2 largely instantiates the CEREALBAR scenario (Suhr et al., 2019),[1] but is implemented from scratch to emphasize research accessibility. CB2 is a customizable, scalable, and complete research platform, including server and clients for multi-agent human-machine interactions, tools for real-time data management, and processes to onboard crowdsourcing workers.

The CB2 scenario poses learning and reasoning challenges, as well as opportunities. Comprehending and producing instructions in CB2 requires addressing the symbol grounding problem (Harnad, 1990), which is studied extensively in the instruction following (e.g., Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Misra et al., 2017; Fried et al., 2018) and generation (e.g., Mei et al., 2016; Wang et al., 2021) literature. However, the collaborative scenario remains relatively understudied. Collaboration is not simply an added complication, but dramatically alters both interaction and learning through joint presence and action. It allows the instructor to ad-hoc modify the tasks they delegate based on the follower behavior, potentially recovering from system failures. At the same time, this adaptation creates constant distribution shift, a significant generalization challenge. Learning is also drastically transformed through collaboration. The constant engagement of other agents (including humans), the ability to modify delegation strategies, and the shared task-based incentives bring about within-interaction signals that can be used for continual learning, reducing the dependency on annotated data and enabling model adaptation.

We deploy a demonstration of CB2 with a learned baseline instruction following agent (Section 7). Players can connect to CB2 and collaborate with our agent or other human agents at https://cb2.ai/.[2] The CB2 platform is available at https://github.com/lil-lab/cb2. A video demonstration of CB2 is available at https://youtu.be/tALpX_KKmIw.

## 2 Related Work

CB2 is a re-implementation and extension of CEREALBAR, a scalable platform to study natural language instruction collaboration (Suhr et al., 2019). CEREALBAR was used to study instruction following (Suhr et al., 2019; Suhr and Artzi, 2022), instruction generation (Kojima et al., 2021), and linguistic change (Effenberger et al., 2021).

[1] CB2 introduces several optional modifications to CEREALBAR aimed at richer language and tighter collaboration.

[2] Our deployment has received IRB exemption. All recorded data is anonymized.

CB2 is related to instruction following environments, such as SAIL (MacMahon et al., 2006), R2R (Anderson et al., 2018), RxR (Ku et al., 2020), and ALFRED (Shridhar et al., 2020). In contrast, CB2 is focused on embodied multi-agent collaborations, including with human agents.

Symbol grounding (Harnad, 1990), a core challenge in CB2, was studied extensively in the single-agent context of instruction following (e.g., Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Fried et al., 2018; Blukis et al., 2018) and generation (e.g., Daniele et al., 2016; Kojima et al., 2021; Wang et al., 2021). The CB2 scenario emphasizes multi-agent collaboration, an aspect that is significantly less studied with natural language instruction. The Cards corpus (Djalali et al., 2012; Potts, 2012) presents a related scenario, which has been used for linguistic analysis. A related problem is studied by the emergent communication literature (Lazaridou et al., 2017; Andreas et al., 2017; Lazaridou and Baroni, 2020), but with less focus on collaboration with human agents. Natural language collaboration between agents with asymmetric capabilities has also been studied with Minecraft-based scenarios (Narayan-Chen et al., 2019; Jayannavar et al., 2020; Kiseleva et al., 2022). CB2 differs from these in allowing both agents to effect changes on the environment, enabling ad-hoc modification and delegation of tasks.

## 3 Interaction Scenario

CB2 largely implements the interaction scenario introduced by (Suhr et al., 2019) in the CerealBar environment with several modifications. The interaction takes place in a procedurally generated spatial environment and includes two agents that collaborate together to complete card collection tasks and coordinate using natural language. Figure 1a shows an instance of the environment.

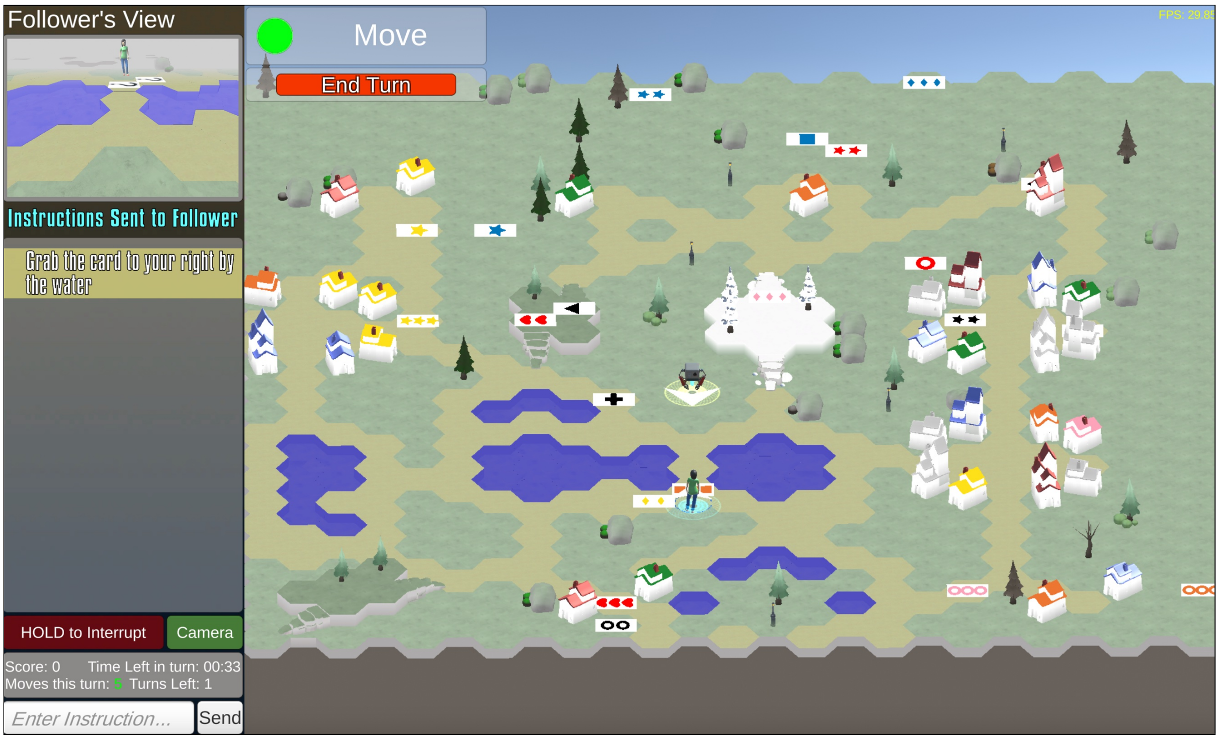
The environment is a procedurally generated 3D map made of a grid of hexagons (Figure 1a). It includes lakes, mountains (Figure 1c), paths, open spaces, and landmarks. A new environment is generated for each game. CB2 includes improved visuals and generation compared to CerealBar. For example, CB2 map generation includes semantic biases: houses are generated to cluster together and form towns (Figure 1b) and paths are generated to connect between meaningful areas in the map, such as towns and mountains. Landmark instances vary visually to elicit richer language. For example, houses are generated with different roof colors and number of floors (Figure 1b). The environment also includes randomly placed cards (Figure 1d). Each card shows 1–3 copies of one of a few possible shapes in one of a few possible colors.

The interaction involves two agents, a leader (Figure 1f) and a follower (Figure 1g), that collaborate together to complete tasks, but differ in their observations of the environments and abilities. Both the leader and the follower move in the environment, by moving between neighboring hexagons or by turning in place to change orientation. The agents select and deselect cards by moving over them (Figure 1e).

The goal of the agents is to select valid sets of cards. A valid set includes three cards, where each color, shape, and count are unique (Figure 1i). The agents select sets together. When the currently selected cards form a valid set, they disappear, the agents together receive one point, three new randomly selected cards appear in random positions, and the agents receive additional turns. The number of turns added diminishes with each set completion. Asymmetries between the two agents make collaboration critical for success.

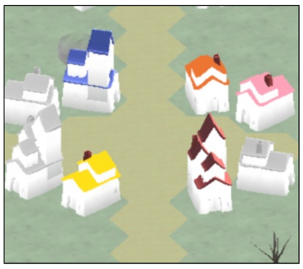
The leader sees a complete overhead view of the environment (Figure 1a), while the follower only sees what is ahead from a first-person view (Figure 1h). CB2 introduces two optional observability features not present in CerealBar. First, the patterns on unselected cards may be hidden from the follower, instead displaying a quesiton mark on all cards. Second, CB2 allows to control how far the follower sees ahead of them with a fog that is present only in the follower view. The observability gap means the leader is in charge of planning how the agents operate. If the follower acts independently of the leader plans, the interaction will be suboptimal, because follower actions are likely to conflict with leader actions and the partial view of the environment does not allow for optimal planning of goals and movement.

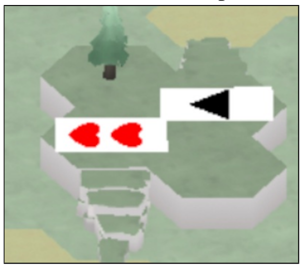
The agents move in turns, with a limited number of steps per turn. Each movement (forward, left, right, or backward) consumes a single step. Turns are time limited to keep the interaction moving and avoid long wait periods for the inactive agent. The exact time budget is customizable, but we generally provide significantly more time for the leader turns, so they can plan as needed. Turns alternate between the follower and leader. The follower has

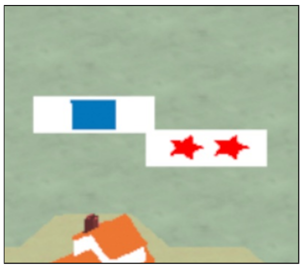

Follower's View
Instructions Sent to Follower
Move
End Turn
FPS: 29.85
HOLD to Interrupt
Camera
Score: 0 Time Left in turn: 00:33
Moves this turn: 5 Turns Left: 1
Enter Instruction...
Send

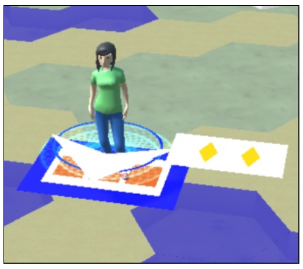
Grab the card to your right by the water

(a) An overhead view of a complete environment with the leader user interface.


(b) A cluster of houses.

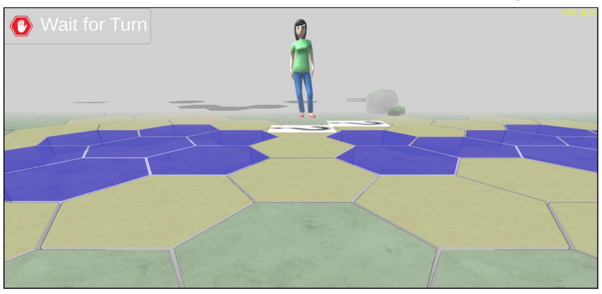

(c) A mountain with ramps.


(d) Cards in the environment.


(e) The leader selecting a card.

(f) The leader character.

(g) The follower character.



(h) The follower point of view.

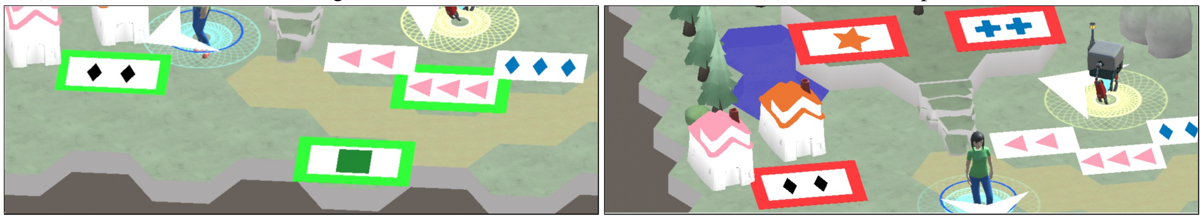

(i) Valid (left) and invalid (right) sets of selected cards.

Figure 1: Images of the game environment and UI. All images are taken from the same environment state.

significantly more steps than the leader per turn. This means the follower is able to move further in each turn, and potentially accomplish much more in each turn. This ability gap makes it critical for the leader to collaborate with the follower, rather than ignore the follower and attempt to accomplish tasks on their own, a suboptimal strategy.

The agents coordinate via uni-directional natural language instruction, the only form of coordination available. During a leader turn, in addition to moving in the environment, the leader can send text instructions to the follower. The follower executes the leader instructions and indicates when an instruction is complete. The leader can queue multiple instructions, but the follower only sees past instructions and the one they are currently executing. Because the follower does not see future instructions, alignment between the actions recorded and the instruction displayed is guaranteed. The leader can also cancel the instruction the follower is executing alongside all future instructions in the queue during the follower turn. This is intended to halt very bad executions, and reduce their overall cost, for example by having to correct drastic departures from the leader plan.

Instruction writing and sending by the leader, and marking them as complete by the follower do not consume steps. Leaders may write as many instructions as they wish during a single turn, and followers are not taxed if the tasks are given in multiple instructions that they need to mark as complete. Exempting the language channel from the budget of actions per turn aims to reduce the influence of the turn systems on the language produced. The combination of collaboration incentives (i.e., because of the capability differences between the agents) and the exclusivity of the language channel for communication makes effective natural language instruction essential for successful interactions.[3]

## 4 Framework Implementation

The CB2 framework has three main components: a Python server, a Unity client, and a Python headless client. The game logic is orchestrated from the server, allowing to customize the interaction without modifying Unity code. The Python client simplifies the interaction between learning processes and the system, for example during reinforcement learning. Figure 2 visualizes the architecture.

CB2's design emphasizes customizability, as much as possible, without modifying Unity code, a skill that is less common among researchers. This motivates placing the game logic on the Python server, a decision that dictates the client-server communication design. However, modifications that require updating the client user interface, such as adding bi-directional communication or translating the UI to other languages, do require modifying Unity coding.

### 4.1 Server

The server architecture is split into modules by logical function. We use asynchronous coroutines to reduce latency efficiently and keep the compute needs small. The platform is parameterized via a configuration file that is loaded by the server.

**Map Generation** Map generation is relatively expensive compared to other processes on the server, mainly because we may use multiple search iterations for routing paths between landmarks and to prevent the leader or follower from spawning in closed-off regions. We mitigate potential lag because of server load by preparing a pool of maps in advance, which we refill during idle periods.

**Player Lobbies** The server supports multiple lobbies concurrently. Separate lobbies provide different player pairing strategies, such as for human-human and human-model games. Players wait in a lobby until they are paired for a game. Each lobby maintains multiple queues for pairing players and assigning roles according to their experience or other information. For example, by default, we distinguish between expert and novice players, and prioritize pairing experts as leaders with novices as followers. Each lobby maintains active game rooms of different types, such as for standard games, tutorials, game replays, and custom scenarios. Each game room contains a game state machine and websocket connections to the clients.

**Data Storage** Game events are recorded into an sqlite3 database, which allows for efficient interaction with game data. Each game is represented as a linear list of events, which can be replayed to recreate game state at a particular moment in time.

**Data Portal** The data portal provides an interface to view game records and statistics. The web data browser shows game-specific recordings, including turns, instructions, and individual player actions. Each game record also includes a link to launch a game replay using the web client. There is also a web page with live statistics, such as the mean and median scores, and a page to download an archive of all server data. The data portal also provides an

[3]Depending on the environment configuration, it is possible for one of the agents to operate alone if the cards forming a set are really close and the other agents does not move. This can allow 1–2 set completions. A higher score without collaboration via language coordination is extremely unlikely.

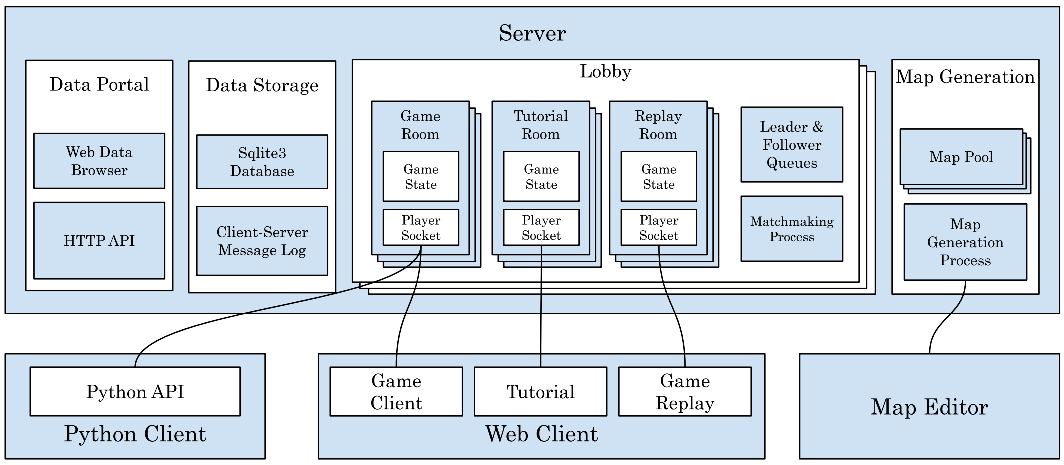


Figure 2: The CB2 system architecture.

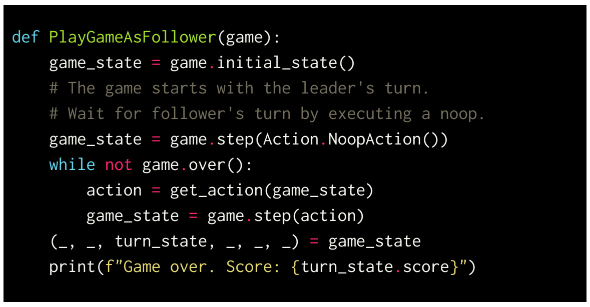
HTTP API for programmatic data access.

**Map and Scenario Editor** Maps are generated procedurally by default. CB2 also provides a map editor for researchers to place users in controlled scenarios. A real-time API allows attaching to an interaction and update the map in response to the game state, enabling dynamic interactions.

## 4.2 Web Client

The web client is developed using the Unity game engine, and is packaged as a WebAssembly binary. The client receives game states, actions, and instructions from the server. We design the client to be thin, putting as much of the game logic and configuration on the server as possible, so that changes to game logic can be made purely in Python. We designed the gameplay user interface (UI) to be accessible and easy to learn by incorporating feedback from players. All UI elements are clustered together, and have keyboard shortcuts. Figure 1a shows the leader interface during a leader turn.

Beyond gameplay, the web client provides a tutorial to onboard players to the game by stepping them through a game interaction accompanied by prompts and tooltips. The tutorial flow is specified on the server, and can be modified easily. For example, rephrasing the tutorial instructions or translating them to other languages is relatively simple and can be achieved by updating the specifications in the server code. The web client also provides game replay, which is activated by adding URL parameters when the HTML page is loaded. The parameters are added automatically to links in the web data browser (Section 4.1).



```python
def PlayGameAsFollower(game):
    game_state = game.initial_state()
    # The game starts with the leader's turn.
    # Wait for follower's turn by executing a noop.
    game_state = game.step(Action.NoopAction())
    while not game.over():
        action = get_action(game_state)
        game_state = game.step(action)
    (_, _, turn_state, _, _, _) = game_state
    print(f"Game over. Score: {turn_state.score}")
```

Figure 3: Example code using the Python API.

## 4.3 Python Client

The programmatic Python client API supports fast lightweight interaction with the game. It is designed for machine learning processes that require interacting with the game environment, such as reinforcement learning (Sutton and Barto, 1998), and can be used to deploy agents interacting with human players or agent-agent interactions. Interaction through this API are similar to interactions with the Unity client, except that recording is optional to reduce overhead. When recorded, they can be replayed using the Unity client. We also provide an OpenAI Gym-style wrapper for the Python API. Figure 3 shows example code.

## 5 Example Task Formulations

CB2 is well suited to study a variety of tasks, with emphasis on learning and evaluation in collaborative interactions with human agents, such as:

**Instruction Following** The task of instruction following is to map a start state observation from the follower perspective and a leader instruction to a sequence of actions. After each action, the agent receives a new observation. Suhr et al. (2019)

studied this problem with CerealBar by learning from recorded human-human interactions, and Suhr and Artzi (2022) studied it within a continual learning from human feedback scenario. Both approaches were evaluated by deploying follower agents to interact with human leaders.

**Instruction Generation** The task of instruction generation is to generate a leader instruction for the follower to execute given an observation of the world state from the leader perspective. This requires planning the cards the two agents should select, divide the tasks, plan trajectories, and express the intended follower trajectory in a natural language instruction. Kojima et al. (2021) focused on the problem of mapping deterministically generated plans to natural language instructions, and proposed a continual learning approach for learning by observing human follower behavior.

**Emergent Communication** CB2 is particularly well suited to study emergent communication in multi-agent systems (Lazaridou and Baroni, 2020). The goal is to jointly learn separate models for the leader and follower. The two models generate actions to move in the world. The leader model additionally generates instructions, which the follower model is conditioned on. The learning can be driven by performance in the game. CB2 easily allows to integrate human agents into the learning and evaluation processes, bringing natural human language into the process. Alternating between interaction between agent-agent and agent-human interactions has the potential to address the language drift problem (Lee et al., 2019).

## 6 Crowdsourcing Process

CB2 poses several relatively demanding crowdsourcing tasks. Human-human interactions require pairing two workers for real-time play over extended time. We design a process to collect CB2 interactions via crowdsourcing, either for games where both roles are controlled by human players, or where one of the sides is controlled by a learned model. The task-focused design of CB2 naturally allows an effective incentive structure by tying game performance with compensation.

The key to our process is gradual training of workers. A new worker first starts with a tutorial and a qualifier quiz that covers the relatively simple role of the follower. The follower role requires following the leader instructions by controlling the character in the game. The worker is then qualified to the follower role only, and is paired by joining a dedicated follower-only queue in the lobby. Focusing on the follower role only simplifies the learning curve, and much of the learning required for the leader role takes place on the job, as the worker collaborates with more experienced leaders.

Once the worker displays sufficient level of performance for several games, they are invited to qualify as a leader by taking a leader tutorial and a quiz. The second tutorial is both longer and more complex than the follower tutorial, and includes both planning and instruction writing. Once the worker completes the tutorial and passes the quiz, they are qualified to the leader role, and can then participate in tasks as both leader or follower.

We design the lobby to pair workers based on experience. Because the leader role is significantly more critical to the effectiveness of the interaction and the quality of language data, we prioritize workers with better performance for it. We measure worker performance, keeping track of the mean game score in the most recent games. If two leader-qualified players are waiting in the lobby for matching, we will assign the leader role to the higher performing of the two.

The pay structure includes a base pay for connecting to the game, and an increasing bonus for each point. Both workers, the leader and follower, get the base pay and the additional bonus per point, tightly connecting compensation to their collaborative performance. Because the leader role is more complex, we provide an additional relative bonus to the worker in the leader role.

## 7 CB2 Demonstration Deployment

We demonstrate the functionality and potential of CB2 via deployment, including collecting a corpus of human-human interaction that we release, training a follower baseline model, and evaluating it in interaction with human leaders.

**Human Games Data** We follow the crowdsourcing process outlined in Section 6 to collect games between human leaders and followers. We collect 185 games containing 3,439 instructions. Table 1 provides data statistics.

**Model and Learning** We train an instruction following model with a behavior cloning objective using the collected human-human data. We filter out poor games to improve training data quality, applying heuristics such as removing games where over 20% of instructions are cancelled. Our

| Dataset | # Games | # Instructions | Mean Score | Vocabulary | Mean Instruction Length |
|---|---|---|---|---|---|
| Training Data | 185 | 3,439 | 6.42 ± 4.88 | 714 | 10.95 ± 5.29 |
| Human-Human Deployment | 187 | 3,404 | 6.69 ± 4.51 | 728 | 11.73 ± 6.09 |
| Human-Model Deployment | 188 | 2,869 | 3.15 ± 3.29 | 542 | 9.62 ± 5.28 |

Table 1: Data and interaction statistics for the human-human training data, and the two side-by-side deployments.

model architecture is based on the Decision Transformer (Chen et al., 2021; Putterman et al., 2022). Follower observations are embedded using HEXACONV (Hoogeboom et al., 2018) because of the hexagonal structure of the map. The observations are centered on the follower's position and rotated such that the follower is always facing the same direction. This baseline model conditions only on the current instruction for simplicity, similar to the model in Suhr et al. (2019). In contrast though, it does not assume full observability.

**Results** We deploy our baseline model as a system demonstration on Amazon Mechanical Turk. We evaluate it with 188 human-model interactions, conducted side-by-side in a randomized experiment with 187 human-human interactions. Human leaders are told that they can be matched with a human or a bot follower in the task description, but are not made aware of who they are interacting with in a specific interaction. Table 1 shows data and interaction statistics for our training data and final deployments. Overall, our models enable effective human-model collaboration in CB2, but at significantly lower performance than observed in human-human games. This is similar to the results of Suhr et al. (2019), although the numbers are not comparable because of the different environment.

Human leaders were able to infer relatively consistently the type of their partner in each interaction. This is indicated by differences in the human leader behavior when comparing human-human and human-model interactions. For instance, the vocabulary human leaders use in interactions with the model is smaller compared to when interacting with human followers and the instructions are shorter. Qualitatively, we observe that instructions in human-human interactions more often use exclamations (e.g., "oh," "shoot," and "oops") and informal speech, with abbreviations such as "btw" and "lol" or words such as "chill" and "kay." We also found that human leaders in human-human games tend to praise their partners, with words such as "awesome," "wonderful," "perfect" or "great" appearing uniquely in instructions from human-human games. The difference is also seen in game statistics. For instance, $16.54\%$ and $12.70\%$ of the times followers and leaders selected a card in human-model games, it was to deselect an already selected card, compared to $8.68\%$ and $8.78\%$ for human-human games. Our results illustrate the challenge posed by CB2, and the importance of the kind of deployment CB2 enables.

## 8 Conclusion

CB2 is a multi-agent research platform to study natural language instruction in collaborative, embodied environments. A core objective of CB2 is to enable scaleable studies where human agents interact with learned models, potentially over long periods of time. CB2 is designed to be easy to use and customize, with emphasis on accessibility for researchers with limited game development experience. It is designed from the ground up for machine learning, and includes a headless fast Python client API to support learning processes and to deploy learned models to interact with human users.

## Acknowledgements

This research was supported by NSF under grant No. 1750499, ARO W911NF21-1-0106. We thank Alane Suhr and Noriyuki Kojima for technical discussions and utility code, and the participating MTurk workers for their work and feedback.

## Ethical Considerations

CB2 is a research environment. The focus on a relatively restricted 3D environment reduces the potential for ethical risks. Our use of CB2 has received exemption status by our institution's IRB office. We recommend that researchers using CB2 obtain IRB approval or exemption for their studies from their institution's IRB office, or an equivalent body. More broadly, systems that learn from interaction with users raise risks of adopting negative behavior patterns from their users. This is especially an issue in certain contexts, such as open ended conversational interfaces or chatbots. This is an important direction for future work. CB2 can be used to study such adversarial usage scenarios in a relatively safe way.

## References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

Jacob Andreas, Anca Dragan, and Dan Klein. 2017. Translating neuralese. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1.

Valts Blukis, Nataly Brukhim, Andrew Bennett, Ross A. Knepper, and Yoav Artzi. 2018. Following high-level navigation instructions on a simulated quadcopter with imitation learning. In *Proceedings of the Robotics: Science and Systems Conference*.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*.

Andrea F Daniele, Mohit Bansal, and Matthew R Walter. 2016. Natural language generation in the context of providing indoor route instructions. In *Proceedings Robotics: Science and Systems Workshop on Model Learning for Human-Robot Communication*.

Alex Djalali, Sven Lauer, and Christopher Potts. 2012. Corpus evidence for preference-driven interpretation. In *Logic, Language and Meaning*.

Anna Effenberger, Rhia Singh, Eva Yan, Alane Suhr, and Yoav Artzi. 2021. Analysis of language change in collaborative instruction following. In *Findings of the Association for Computational Linguistics: EMNLP*.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42.

Emiel Hoogeboom, Jorn W.T. Peters, Taco S. Cohen, and Max Welling. 2018. Hexaconv. In *International Conference on Learning Representations*.

Prashant Jayannavar, Anjali Narayan-Chen, and Julia Hockenmaier. 2020. Learning to execute instructions in a Minecraft dialogue. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Julia Kiseleva, Alexey Skrynnik, Artem Zholus, Shrestha Mohanty, Negar Arabzadeh, Marc-Alexandre Côté, Mohammad Aliannejadi, Milagro Teruel, Ziming Li, Mikhail Burtsev, et al. 2022. Iglu 2022: Interactive grounded language understanding in a collaborative environment at neurips 2022. *arXiv preprint arXiv:2205.13771*.

Noriyuki Kojima, Alane Suhr, and Yoav Artzi. 2021. Continual learning for grounded instruction generation by observing human following behavior. *Transactions of the Association for Computational Linguistics*, 9.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Angeliki Lazaridou and Marco Baroni. 2020. Emergent multi-agent communication in the deep learning era. *ArXiv*, abs/2006.02419.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2017. Multi-agent cooperation and the emergence of (natural) language. In *International Conference on Learning Representations*.

Jason Lee, Kyunghyun Cho, and Douwe Kiela. 2019. Countering language drift via visual grounding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Matthew MacMahon, Brian Stankiewics, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence*.

Hongyuan Mei, Mohit Bansal, and R. Matthew Walter. 2016. What to talk about and how? Selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier. 2019. Collaborative dialogue in Minecraft. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Christopher Potts. 2012. Goal-driven answers in the Cards dialogue corpus. In *Proceedings of the West Coast Conference on Formal Linguistics*.

Aaron L Putterman, Kevin Lu, Igor Mordatch, and Pieter Abbeel. 2022. Pretraining for language conditioned imitation with transformers. *Offline Reinforcement Learning Workshop at Neural Information Processing Systems*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Alane Suhr and Yoav Artzi. 2022. Continual learning for instruction following from realtime feedback. *arXiv preprint arXiv:2212.09710*.

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 Conference on Empirical Methods in Natural Language Processing*.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning: An introduction*. MIT press.

Su Wang, Ceslee Montgomery, Jordi Orbay, Vighnesh Birodkar, Aleksandra Faust, Izzeddin Gur, Natasha Jaques, Austin Waters, Jason Baldridge, and Peter Anderson. 2021. Less is more: Generating grounded navigation instructions from landmarks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.