Integrating Language Models and Symbolic Planning for Grounded Dialogue

Anonymous EMNLP submission

Abstract

Large language models excel at understanding and generating text and code; however, they face limitations in adhering to task-specific constraints and handling novel grounding. These limitations are exacerbated in grounded task-oriented dialogue, where constraint and grounding errors lead to compounding errors. We address these shortcomings by composing large language models into a classical modular dialogue system, consisting of a reader, planner, and writer. The reader leverages language models to convert partner utterances into executable code, calling functions that perform grounding. The translated code's output is stored to track dialogue state, while a symbolic planner determines the next appropriate response. We evaluate our system's performance on the demanding ONECOMMON dialogue task, where dialogue is grounded on abstract images of scattered dots. In this challenging setting, our system outperforms humans with a success rate of X% vs 65.8%.

1 Introduction

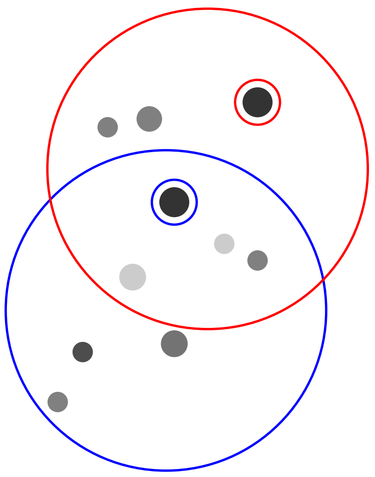
Large language models are strong dialogue systems, capable of obtaining reliable and accurate results in zero or few-shot settings (Madotto et al., 2020; Thoppilan et al., 2022; Peng et al., 2022). However, grounded dialogue often requires multimodal grounding in visual context. Recent work composes large language models with separately pretrained vision models, hoping that vision models can generalize to novel image domains (Li et al., 2023). This exposes grounded dialogue systems to a multiplicative source of error: if just one of either the language or vision are out of domain for any component of a dialogue system, overall system accuracy will suffer.

One possible solution to multimodal compounding error is modular models. Recent work has investigated the use of code generating language models for composing vision and language models in a modular fashion, processing each modality separately (Liang et al., 2022; Dídac et al., 2023). The modularity of these systems also enables task-specific biases to be built into systems. For example, Liang et al. (2022) craft a Python library that serves as a tool a code generation model can call in order to manipulate a robot arm.

Modular grounding with code-generating systems has largely been studied in the single turn setting, rather than grounded dialogue. Grounded dialogue offers new challenges: dialogue systems must deal with linguistic phenomena such as coreference and ellipsis, as well as planning and response generation. We present a modular grounded dialogue system, RTPW, that utilizes code generation for grounding, while incorporating components of dialogue system pipelines for handling the unique challenges afforded by dialogue.

RTPW follows a classical dialogue system decomposition: read, plan, and write (Young et al., 2013). The system reads partner utterances and parses them to code, using the execution results to track dialogue state and plan what to say next. At the core of the system is the state representation: We integrate both a probabilistic belief state (Young, 2006) as well as a dataflow approach (Andreas et al., 2020) in order to both perform symbolic planning and reference resolution. Our system does not require supervision beyond few-shot prompting. (Justin: More? probably not clear, too much jargon)

We evaluate RTPW on ONECOMMON, a challenging grounded dialogue task (Udagawa and Aizawa, 2019). The visual grounding in ONECOMMON is particularly challenging: Players are required to discuss a set of abstract dots, a setting not commonly found in pretraining datasets for vision models. The dots themselves are often difficult to describe, with referring expressions made even more ambiguous by ONECOMMON's partial observability.



A: I have one large black dot by itself. Do you have it?
P: Yes, I do have that.
A: Let's pick that one.
P: <select> red
A: <select> blue

Figure 1: An example of perspective-dependent ambiguity from a dialogue agent and human partner playing OneCommon, taken from the evaluation of Fried et al. (2021). The players have different but overlapping circular views of a shared board, which contains dots of different shades and sizes. The agent and partner must collaborate through dialogue in order to find and select a shared dot. This dialogue ends in failure, as the agent and partner did not account for perspective-ambiguity and prematurely selected different dots.

In human evaluation, RTPW outperforms human players. Demonstrates method for minimally supervised adaptation of dialogue systems to new grounded tasks.

2 Related work

Modular dialogue systems SHRDLU. POMDP Young (Young, 2006). SMCalFlow (Andreas et al., 2020). Decoupling strategy and generation (He et al., 2018). (Madotto et al., 2020). Difference: Combine pieces of each: simple executable state, flexibility afforded by code-generating language models, probabilistic belief in grounded dialogue task.

Tool-based language models Code-as-policies (Liang et al., 2022), ViperGPT (Dídac et al., 2023), PAL, toolformer, VisProg. VisualChatGPT, chatgpt with plugins. Difference: Explicit symbolic planning, integration with classical dialogue system.

Planning Lots of recent LM planning papers. (Gandhi et al., 2023) Task-oriented dialogue often uses multi-turn model-based planning. Here planning is performed in the space of language via rollouts or tree-search (Lewis et al., 2017; Yarats and Lewis, 2017; Shridhar and Hsu, 2018; Jang et al., 2020). While we use a single-turn planning heuristic, our focus on improving partner models is complementary to multi-turn planning and can be combined in future work.

Agent	Acc
RTPW	%
GPT4-MC	%
Fried et al. (2021)	%
BLIP	%

Table 1: Reference resolution

3 Background

This section provides the background on OneCommon (Udagawa and Aizawa, 2019), the grounded collaborative dialogue task of focus. OneCommon is a particularly difficult collaborative reference game, played between an agent and partner. The agent and partner's shared goal is to find one dot that is visible to both, one dot in common. This is complicated by the fact that the agent and partner have different but overlapping views, illustrated in Figure 1, leading to perspective-dependent ambiguity. For example: A dark dot to the agent may not be dark to the partner, depending on what they see. Games are constructed so that the agent and partner share between 4–6 dots, with fewer shared dots resulting in increased ambiguity and therefore game difficulty. We focus on the most challenging setting, where 4 dots are shared.

This setting is particularly difficult for pretrained vision and language models. We hypothesize that descriptions of combinations of abstract dots are not well-represented in pretraining datasets. We motivate our system with a small-scale preliminary study, where we test whether systems can resolve simple templated referring expressions in OneCommon. The results in Table 1i show that resolution is difficult for a state-of-the-art vision and language model, with a language-only prompting approach faring similarly. However, a task-specific but minimally supervised code generation approach is quite accurate.

4 Modular systems for reference games

We consider the grounded dialogue setting where an agent and their partner must communicate to achieve a joint goal. This section provides a high-level overview of our dialogue system, RTPW, with implementations given in Sections **??**-**??**.

Our goal is to build a grounded dialogue system that, given a dialogue context and dialogue history, produces the next utterance. We propose a dialogue system, RTPW, which follows a classical dialogue system pipeline decomposition: reading, tracking, planning, and writing (Young, 2006; Young et al., 2013; He et al., 2018) and also integrates aspects of a dataflow system (Andreas et al., 2020). The modular pipeline allows us to build in inductive biases that allow the targeted use of a general-purpose large language model in combination with high-precision task-specific components. (Justin: Sufficient coverage over understanding and generation to complete the task relatively often.) We provide an illustration of the system in Figure 2.

At each turn, partner utterances are parsed into the body of a Python function:

> User: *Do you see a black dot?*

```
def turn(state):
  dots = set()
  for dot in single_dots():
    if is_dark(dot):
      dots.add(dot)
  return dots
```

The purpose of the resulting Python function is to build up a representation of both the common ground up to a given turn as well as the current question under discussion. For example, backtracking and reference are both important characteristics of dialogue ():

> User: *Do you see a big black dot?*
> Agent: *Yes*
> User: *Is there a big light dot next to it?*

```
def turn(state):
  dots = state
  for old_dot in dots:
    for dot in single_dots():
      if is_light(dot) and close(old_dot, dot):
        dots.add([old_dot, dot])
  return dots
```

This user utterance refers to the output of previous computation, which is given as the `state` argument. Similar to Andreas et al. (2020), we utilize task-specific heuristics for obtaining `state`, which we describe in Section 5.1.

Dialogue state is tracked by composing the output of these functions and synthesizing those outputs into a probabilistic belief state over task-specific sufficient statistics. The belief state is updated via Bayes' rule, utilizing a model of partner utterances given dialogue history and state (Young, 2006). The belief state is then used to plan the agent's response. For example, if the agent is certain, it may respond to the above utterance by confirming the shared light dot and proceeding to end the game.

Given a plan, responses are generated with rule-based templates, with the details given in Section 5.4. The system capitalizes on a special characteristic of cooperative two-player games: we can control the trajectory of the conversation, allowing the system to only support a small subset of all possible dialogues. In practice, this could be extended to more flexible albeit slower response generation.

5 System implementation

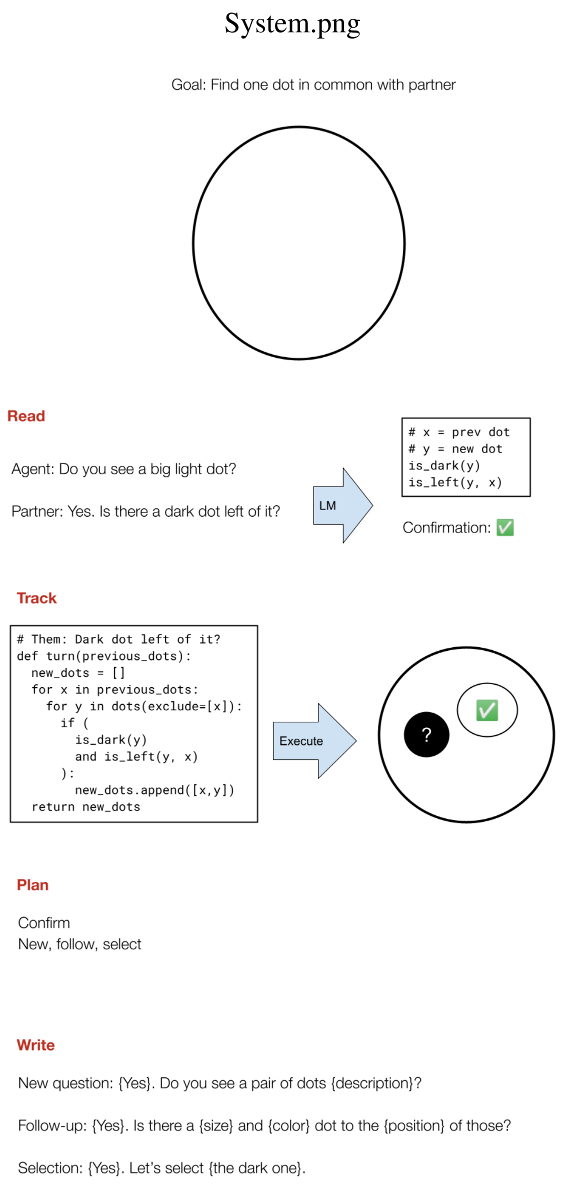
This section provides the implementation details of our system, specialized to ONECOMMON.

5.1 Read

We frame reading as mapping utterances to symbolic plans. A subset of these symbolic plans are then used to construct an executable Python function. Each plan consists of a dialogue act, set of mentioned dots, and confirmation. The dialogue act is a coarse description of the purpose of the utterance. We consider three dialogue acts: start, follow-up, select. with an additional bucket act no-op. Depending on the dialogue act, the set of mentioned dots may build upon previous mentions. The set of mentioned dots may be empty. A confirmation indicates whether the partner sees the agent's mentioned dots in the preceding turn. An utterance may neither confirm nor deny previous dot mentions.

We prompt a large language model to obtain plans from utterances. Similar to Code as Policies (Liang et al., 2022), we define a library of Python functions, expose them in the prompt as import statements, and provide in-context examples of utterances and generated code. Since the generated code is in Python, the model is able to flexibly combine our custom library with other Python code such as list comprehensions and standard library functions. We provide an example in Figure 2.

PROMPTS HERE



Agent: Do you see a big light dot?

Partner: Yes. Is there a dark dot left of it?

```
# x = prev dot
# y = new dot
is_dark(y)
is_left(y, x)
```

```
# Them: Dark dot left of it?
def turn(previous_dots):
  new_dots = []
  for x in previous_dots:
    for y in dots(exclude=[x]):
      if (
        is_dark(y)
        and is_left(y, x)
      ):
        new_dots.append([x,y])
  return new_dots
```

Confirm
New, follow, select

New question: {Yes}. Do you see a pair of dots {description}?

Follow-up: {Yes}. Is there a {size} and {color} dot to the {position} of those?

Selection: {Yes}. Let's select {the dark one}.

Figure 2: Dialogue system.

5.2 Track

The agent maintains a belief distribution over which of the 7 dots are shared. The agents update this belief if our last plan is either confirmed or denied, and if the agent can resolve the partner's dot mentions.

In both cases, we rely on Bayes' rule.

5.3 Plan

Symbolic planning: Ask a new question, follow-up question, or select.

Pick the action that has both the highest information gain but is also easy to understand. Don't forget to describe info gain. Effectively minimizing description length by prioritizing follow-up when possible. maximize Information gain - communicative cost.

In-depth questions are key to building common ground. In-depth questions without coreference are impossible to understand. Coreference without grounding is always incorrect. Symbolic planning on top of dialogue state tracking makes this possible.

5.4 Write

Realize the plan into language. We rely on three simple templates: 1) a start template, which initiates a line of inquiry, 2) a follow-up template, which builds on a previous question, and 3) a select template, which communicates which dot to select.

6 Experimental setup

We evaluate our method, TODO (RTPW), on the ONECOMMON dataset (Udagawa and Aizawa, 2019). The primary evaluation is agent success with human partners. We also report success in selfplay, where agents play with a copy of themselves. We offer additional ablations on prompt design, and share findings on practical aspects of prompt and code-based dialogue systems.

Systems ViperGPT baseline.

We compare our RTPW with a chain-of-thought baseline that does not rely on code execution or symbolic planning. As this baseline does not have access to code execution-derived grounding, we describe the visual context in words.

As a baseline, we compare to an agent from prior work, which does not account for perspective-dependent ambiguity (Fried et al., 2021). The baseline agent chooses 8 plans and 64 utterances for each plan, then chooses the plan and utterance pair with the highest probability of recovering the plan give the utterance using a reference resolution model applied only to the agent's own context.

We also experiment with generating the Python function directly, allowing the code generation model to resolve referring expressions on its own.

Human evaluation We also evaluate the Partner Planner on symbolic selfplay, where it plays the full ONECOMMON game with a copy of itself using symbolic communication. We evaluate only on the setting with the most perspective-dependent ambiguity, 4 shared dots. In symbolic selfplay, agents must perform planning, belief updates, and selection. The Partner Planner is able to exactly communicate confirmations and dot features (bucketed size, color, and relative positions). We compare the game success rate of the Partner Planner to success rate of the baseline by (Fried et al., 2021) and human performance on language selfplay. Language inherently has more noise than symbolic representations, meaning symbolic selfplay is an upper bound on performance with language.

Selfplay In order to show that resolving perspective-dependent ambiguity reduces errors, we perform automatic evaluation of plans by evaluating whether the agent plan is incorrectly resolved by the partner model. A plan is incorrectly resolved if the plan is not empty and the partner does not resolve the plan to any of the agent's intended referents. We evaluate this without language by directly feeding the feature representation of plans from the agent to the partner, who is a Partner Planner.

For each turn in the human-generated dialogue, we generate a plan from our model and label that plan as either a success or failure using the procedure above. We evaluate on 518 validation games, which have different numbers of shared dots: either 4, 5, or 6. Fewer shared dots results in more perspective-dependent ambiguity.

Hyperparameters For the Partner Planner, we set the response faithfulness probability $\theta = 0.95$. We determine the selection entropy threshold by running grid search over $\tau \in \{1, 1.5, 2\}$ on symbolic selfplay, and pick the value with the highest success rate.

The partner response classifier is a RoBERTa model (Liu et al., 2019), with a second stage of fine-tuning performed on 147 annotated dialogue turns. We annotate the text each turn as a confirmation, denial, or no response. The model is originally fine-tuned on sentiment (Heitmann et al., 2020).

For mention classification from text, we use a mention prediction network from past work (Fried et al., 2021). The mention prediction network explicitly models relationships between mentions using a conditional random field with neural potentials.

7 Results

Human evaluation

Selfplay

8 Analysis

needs to compare our system to the second best thing.

Agent	Success	Avg # turns
RTPW	%	
GPT4	%	
Human	%	
Fried et al. (2021)	62.4%	

Table 2: The success rate of different agents in selfplay on the hardest setting of ONECOMMON, with 4 shared dots. A higher success rate is better. The human performance is from the ONECOMMON dataset (Udagawa and Aizawa, 2019).

Agent	Success	Avg # turns
RTPW	%	
GPT4	%	
Human	%	
Fried et al. (2021)	62.4%	

Table 3: The success rate of different agents in selfplay on the hardest setting of ONECOMMON, with 4 shared dots. A higher success rate is better. The human performance is from the ONECOMMON dataset (Udagawa and Aizawa, 2019).

Prompt style	Acc	Time	Len
RTPW	%		
Single prompt	%		
Shorter	%		

Table 4: An analysis of different code generation prompt styles.

Codegen good at OOD Show template parsing results. We get 22-24/25, fried gets 4/25. Also does not conflate meaning with distribution of dots people mention. Weird because we are using large model vs LSTM? IF LSTM beats GPT, thats ok.

Reference resolution worry: if make claim that bad at refres, might be hard for reviewers to understand.How strong is the code generation system at understanding normal text? Static reference resolution results, comparison to Fried baseline. Conclusion: Not strong, but due to the 2-player nature of the game, able to steer the conversation. Makes intentional planning even more important (avoid weaknesses / ambiguity).

Prompt style analysis Length of prompts and number of runtime errors.

Use a different system s/gpt4/starcoder? s/gpt4/gpt3.5?

Follow-up vs just describe everything See if

Agent	Success	Avg # turns
RTPW	%	
GPT4	%	
Human	%	
Fried et al. (2021)	62.4%	
Human	65.8%	4.97

Table 5: The success rate of different agents in selfplay on the hardest setting of ONECOMMON, with 4 shared dots. A higher success rate is better. The human performance is from the ONECOMMON dataset (Udagawa and Aizawa, 2019).

follow-up heuristic vs info gain surprises user.

Static plan evaluation Directly modeling and marginalizing over unobserved partner perspective results in fewer errors than the egocentric heuristic from Fried et al. (2021), obtaining a 12% reduction in errors as shown in Table **??**. Additionally, the number of incorrectly resolved plans from Planner is much lower than an ablated planner that does not model unshared dots, a 72% reduction.

We hypothesize that the baseline model of Fried et al. (2021) makes fewer errors because it is able to repeat plans mentioned in the static dialogue. The Partner Planner does not often repeat plans, as there is little information gained. When restricted to the plan proposals from the first turn of a static human-demonstrated dialogue, where no plans can be copied, the Partner Planner outperforms the baseline by a larger margin, as shown in Table **??**.

We note that the absolute number of resolution errors is small relative to the total number of turns. We hypothesize that this is because of the nature of static evaluation. Static evaluation considers next step plan proposals given human dialogue, preventing agents from steering the dialogue themselves.

Symbolic selfplay The Partner Planner achieves strong performance in symbolic selfplay, as shown in Table 5. The ablated version, which does not model unshared dots, also performs well, but worse than the full Partner Planner. This demonstrates the utility of modeling unshared perspective.

Both the full and ablated Partner Planner outperform the baseline of Fried et al. (2021) and coached human performance from the training data of Udagawa and Aizawa (2019), demonstrating the utility of partner modeling. Much of this success comes from the ability to control the belief entropy selection heuristic, as shown by the performance of the ablated Partner Planner over human performance. The selection heuristic encourages the Partner Planner to be more patient and gather more information before selecting than most human participants, reflected in the average number of turns per game.

9 Future Work

For future work, the current model will be extended with text generation in order to play ONECOMMON with humans. Preliminary experiments using utterance generation methods from prior work by Fried et al. (2021) found that the plans proposed by the Partner Planner were out-of-distribution for supervised methods, resulting in many errors in generation. Further experiments with a templated generation system found that the resulting templates were too difficult to understand for human partners. Future work will experiment with generation systems that account for utterance processing cost, balancing the amount of information conveyed and fluency.

Additionally, the scalability of the method will be tested on dialogue games with larger environments.

Acknowledgements

Ge Gao, Omer Gul, Woojeong Kim, Celine Lee, Jack Morris, Chenran Ning, Alane Suhr, Nicholas Tomlin, Jiawei Zhou.

Limitations

There is a tradeoff between expressivity and controllability for dialogue partner modeling. In this work, we emphasize controllability by explicitly modeling certain aspects of the partner perspective while not accounting for others, resulting in biased agents. The failure to account for particular partner aspects may affect fairness and equity with a diverse pool of partners.

References

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam

Thomson, Andrei Vorobev, Izabela Witoszko, Jason Andrew Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *CoRR*, abs/2009.11423.

P.R. Clarkson and A.J. Robinson. 1997. Language model adaptation using mixtures and an exponentially decaying cache. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 799–802 vol.2.

Surís Dídac, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*.

Daniel Fried, Justin T. Chiu, and Dan Klein. 2021. Reference-centric models for grounded collaborative dialogue. In *Proceedings of EMNLP*.

Kanishk Gandhi, Dorsa Sadigh, and Noah D. Goodman. 2023. Strategic reasoning with language models.

Dmitriy Genzel and Eugene Charniak. 2002. Entropy rate constancy in text. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 199–206, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Janosch Haber, Tim Baumgärtner, Ece Takmaz, Lieke Gelderloos, Elia Bruni, and Raquel Fernández. 2019. The photobook dataset: Building common ground through visually-grounded dialogue. *CoRR*, abs/1906.01530.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. *CoRR*, abs/1704.07130.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. *CoRR*, abs/1808.09637.

Mark Heitmann, Christian Siebert, Jochen Hartmann, and Christina Schamp. 2020. More than a feeling: Benchmarks for sentiment analysis accuracy. *Available at SSRN 3489963*.

T. Jaeger and Roger Levy. 2006. Speakers optimize information density through syntactic reduction. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

Youngsoo Jang, Jongmin Lee, and Kee-Eung Kim. 2020. Bayes-adaptive monte-carlo planning and learning for goal-oriented dialogues. 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 7994–8001. AAAI Press.

Mike Lewis, Denis Yarats, Yann N. Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning for negotiation dialogues. *CoRR*, abs/1706.05125.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pretraining with frozen image encoders and large language models.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2022. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*.

D. V. Lindley. 1956. On a Measure of the Information Provided by an Experiment. *The Annals of Mathematical Statistics*, 27(4):986 – 1005.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Andrea Madotto, Zihan Liu, Zhaojiang Lin, and Pascale Fung. 2020. Language models as few-shot learner for task-oriented dialogue systems. *CoRR*, abs/2008.06239.

Baolin Peng, Michel Galley, Pengcheng He, Chris Brockett, Lars Liden, Elnaz Nouri, Zhou Yu, Bill Dolan, and Jianfeng Gao. 2022. Godel: Large-scale pre-training for goal-directed dialog. arXiv.

Mohit Shridhar and David Hsu. 2018. Interactive visual grounding of referring expressions for human-robot interaction. *CoRR*, abs/1806.03831.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. 2022. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239.

Takuma Udagawa and Akiko Aizawa. 2019. A natural language corpus of common grounding under continuous and partially-observable context. *Proceedings* *of the AAAI Conference on Artificial Intelligence*, 33(01):7120–7127.

Yang Xu and David Reitter. 2018. Information density converges in dialogue: Towards an information-theoretic model. *Cognition*, 170:147–163.

Denis Yarats and Mike Lewis. 2017. Hierarchical text generation and planning for strategic dialogue. *CoRR*, abs/1712.05846.

Steve Young. 2006. Using pomdps for dialog management. In *2006 IEEE Spoken Language Technology Workshop*, pages 8–13.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

A Partner Modeling in Reference Games

Collaborative reference games pair an agent and a partner in order to agree on a shared object through natural language dialogue. At each turn, the agent or partner may decide to terminate the game and make a selection. Once either the agent or their partner terminates, the other player must also act (without observing the other's choice). If both agent and partner agree, both win; otherwise, both fail.

Our approach to reference games separates planning of utterances (choosing what to talk about) from surface realization (choosing how to say it). At each turn, our agent produces an utterance plan x by using a *partner model*, which simulates the partner's possible responses, y, given their hidden perspective, z. The agent uses the partner model to infer the partner's perspective and predict the partner's responses to the agent's plans. We first give an overview of the partner model and planning procedure in this section.

Partner model We model the partner's perspective as a latent variable, infer the value for this variable over the course of a game, and use it to plan generation. This contrasts with a typical egocentric heuristic, as used in Fried et al. (2021), which assumes the partner's perspective is identical to the agent's.

The partner model predicts a distribution over the partner response y given the agent plan x under the latent shared perspective z, and decomposes as:

$$p(y \mid x) = \sum_z p(y \mid x, z) p(z).$$

Planning The agent uses the partner model to plan what to say next, by choosing the plan x that maximizes the expected information gain (Lindley, 1956) about the shared perspective z, defined as

$$\underset{x}{\operatorname{argmax}}\, H[z] - \mathbb{E}_{y|x}\left[H[z \mid x, y]\right],$$

where $H[z]$ is the entropy of the prior[1] and $H[z \mid x, y]$ the posterior, which requires marginalizing over z.

Belief update After observing the partner response y, the agent updates its belief $p(z)$ over the shared with Bayes' rule:

$$p(z \mid x, y) = p(y \mid x, z)p(z) / \sum_z p(y, z \mid x)$$

This is performed iteratively after each turn, and requires marginalizing over possible shared perspectives.

Selection After gathering information through planning and incorporating information through belief updates, the agent must decide when it has built enough common ground, collaboratively identifying a shared dot with its partner. We set a threshold on the belief entropy, $H[z]$, which determines when the agent should transition from information gathering to ending the game.

B Planning in OneCommon

We focus on applying partner modeling to OneCommon (Udagawa and Aizawa, 2019), which represents a class of collaborative reference games (He et al., 2017; Haber et al., 2019) where only a subset of each player's perspective is shared, resulting in perspective-dependent ambiguity.

In OneCommon, the agent's known perspective $\mathcal{D}$ consists of 7 dots in its view. Each dot has a set of features: size, color, and position in the 2D plane. All features are continuous. The main challenge of the game is that the partner perspective is also a view of 7 dots, Between 4–6 of those dots are shared with the agent perspective which we denote as the shared perspective z. Additionally there are a set of unshared dots u the fill out the partner perspective. An example is given in Figure 3. Note that a smaller number of shared dots increases the likelihood that plans get misresolved to unshared dots, increasing perspective-dependent ambiguity.

The agent communicates with the partner by producing an utterance plan, x, which it then describes in natural language. This plan is a subset of

[1]The prior entropy $H[z]$ in the definition of information gain is constant with respect to the plan x, and can be dropped from the objective.

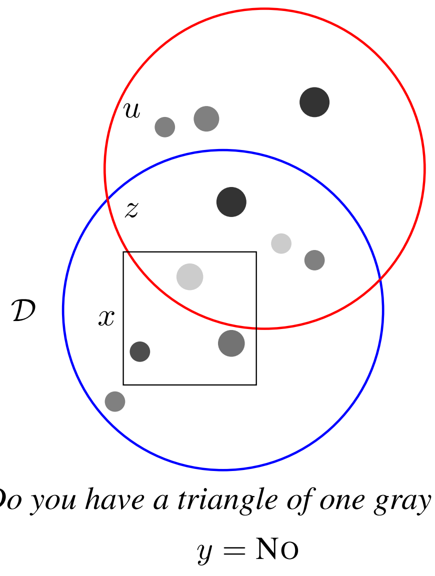


Figure 3: In ONECOMMON, the agent's perspective $\mathcal{D}$ is represented by the large blue circle, and the partner's unobserved perspective by the red. The shared dots z are in both perspectives, while the unshared dots u are only in the red circle. The agent plan x is given by the dots in the box, and also described in language. The partner response y is a binary confirmation.

the dots in the agent view, $x \subseteq \mathcal{D}$, that the agent will ask the partner about. The partner gives a response y to the plan x, given their perspective z. In ONECOMMON, the partner responds in natural language; however, the partner model only models the partner response as a confirmation $y \in \{\text{YES}, \text{NO}\}$, obtained by classifying natural language responses.

Exact planning is intractable because the objects in the partner perspective have continuous-valued features. In this section, we describe simplifying assumptions for the partner model and inference procedure that make planning tractable.

B.1 Partner model

We build a partner model by factoring the shared perspective z and partner response y as illustrated in Figure 3. Formally,

$$\begin{aligned} p(y \mid x) &= \sum_z p(y \mid x, z) p(z) \\ &= \sum_{z,u} p(y \mid x, z, u) p(z) p(u), \end{aligned}$$

where we introduce the latent variable u representing the unshared dots in the partner perspective.

The shared dot prior, $p(z)$, is a distribution over subsets of $\mathcal{D}$, indicating which dots in the agent perspective $\mathcal{D}$ are shared with the partner. The model $p(z)$ is initially uniform over dot subsets at the start of a game, but is updated given evidence from the partner response y at the end of each turn, $p(z \mid x, y)$. For notational simplicity we focus on the first turn.

The unshared dot prior, $p(u)$, is a distribution over the remaining partner dots. Since the dots in u are unobserved by the agent, we parameterize $p(u \mid s)$ using a uniform distribution over discretized features for each dot. We ignore spatial features for dots in u and discretize the other originally continuous features: size and color.[2]

The confirmation model, $p(y \mid x, z)$, checks whether a partner will confirm or deny the agent plan. The partner confirms if they are able to resolve the plan x to their perspective. Given a fully observed z and u, resolution of a plan x is performed by matching the features of x to z and u. There are no trained parameters in resolution, as it depends only on the features of dots in x, z, and u. See Appendix C for the details of feature-based resolution.

In order to avoid jointly enumerating z and u, the model reasons separately about z and u by making the simplifying assumption that plans are fully in z or u. This means that the model will deny if part of x is in z, while the remainder is in u (and x is not fully contained in either z or u):

$$\begin{aligned} p(y = \text{NO} \mid x) &= \sum_{z,u} p(y = \text{NO} \mid x, z, u) p(z, u) \\ &= \sum_z p(y = \text{NO} \mid x, z) p(z) \\ &\quad \cdot \sum_u p(y = \text{NO} \mid x, u) p(u). \end{aligned}$$

Given the unsuccessful resolution of x to both z and u, the partner denies accurately with probability θ, a hyperparameter.

B.2 Inference

During inference, we need to compute $p(y \mid x)$ for all plans x, which can be done in two steps: First, we marginalize over the unshared dots u. This marginalization can be expressed in closed form. For the details, see Appendix D. Second, we marginalize over the possible set of shared dots z. The computational cost of marginalization is the size of the power set of $\mathcal{D}$, $O(2^{|\mathcal{D}|})$.

[2] We discretize size and color uniformly into 3 buckets based on their absolute range across ONECOMMON.

We utilize this distribution to compute the posterior on the shared perspective z, after observing a partner response to a plan,

$$p(z \mid x, y) = \frac{p(z, y \mid x)}{p(y \mid x)}.$$

This posterior then allows us to perform optimization over plans with respect to the expected information gain, as well as update our beliefs given the partner response.

Planning Planning optimizes the expected information gain with respect to the shared perspective z:

$$\underset{x}{\operatorname{argmin}} \mathbb{E}_{y|x} \left[H(z \mid x, y) \right].$$

Computing $p(y \mid x)$ has cost $O(2^{|\mathcal{D}|})$, while there are also $O(2^{|\mathcal{D}|})$ plans.[3] As a result, optimizing this objective takes $O(2^{2|\mathcal{D}|})$ computation, and is performed in less than one second on CPU.

Belief update The belief update directly uses the posterior distribution $p(z \mid x, y)$, as described in Section A.

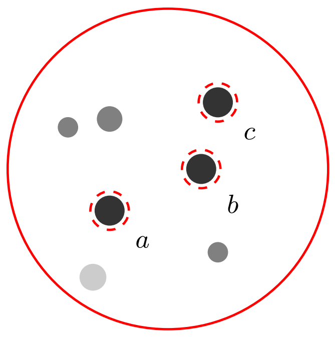
During gameplay in OneCommon, the agent either directly observes the symbolic response y or receives a description of y in natural language. In order to process the natural language dialogue, we use a classifier to extract y from natural language. Additionally, the partner can mention dots of their own, either symbolically or described in text. The agent incorporates partner mentions into its belief by treating them as a confirmed plan. We use another classifier to extract partner mentions from text. We give the details of both the response and mention classifiers in Section 6.

Selection To determine when to select a dot, the agent uses a threshold on the entropy $H[z]$, given by the hyperparameter τ. The agent them communicates which dot to select by describing the configuration of four dots with the highest marginal probability of being shared, as well as the dot within that configuration that is most likely to be shared. The agent then selects the described dot.

C Feature-based resolution

Feature-based resolution featurizes a plan x then compares the features to all subsets of dots in the partner domain $\mathcal{D}$. The set of features used for each plan x is given by the shape and size each dot in the plan, bucketed into 3 bins based on the range

[3] Plans x are subsets of $\mathcal{D}$ that the agent would like to ask the partner about.



Feature representation
Dot 1: Large, dark
Dot 2: Large, dark, below-left Dot 1

Figure 4: An example of feature-based resolution. The above feature representation for a pair of dots resolves to dot configurations $\{(a, b), (a, c), (b, c)\}$.

of each feature. The pairwise positions, limited to above-left, above-right, below-left, and below-right, are also contained in the feature set. We provide an example of feature-based resolution in Figure 4.

Given a plan x, feature-based resolution must compare all the features of the plan, of which there are $O(|x|^2)$, to all partial permutations of subsets size $|x|$ taken from $\mathcal{D}$, of which there are $O(|\mathcal{D}|^{|x|})$. This can be precomputed at the start of a dialogue.

When resolving x to unshared dots, we ignore spatial features.

D Resolution to unshared dots

The partner model, with the assumption that x cannot be split between z and u, is given by

$$\begin{aligned} & p(y = \text{NO} \mid x) \\ & = \sum_{z,u} p(y = \text{NO} \mid x, z, u) p(z) p(u) \\ & \approx \sum_{z,u} p(y = \text{NO} \mid x, z) p(z) p(y = \text{NO} \mid x, u) p(u) \\ & = \sum_{z} p(y = \text{NO} \mid x, z) p(z) \sum_{u} p(y = \text{NO} \mid x, u) p(u) \\ & = \sum_{z} p(y = \text{NO} \mid x, z) p(z) \sum_{u} p(y = \text{NO}, u \mid x). \end{aligned}$$

The probability a plan x resolves to the unshared dots u is

$$\sum_{u} p(y = \text{NO}, u \mid x) = 1 - \theta \binom{|u|}{|x|} \frac{B^{2 \cdot (|u| - |x|)}}{B^{2 \cdot |x|}},$$

where B is the feature bucket size, given $|u| \geq |x|$. This relies on the assumption that spatial features are ignored when resolving to unshared dots.

E Planning objective

In addition to maximizing the information gain, we also add an entropy rate constraint to the planning objective, motivated by the entropy rate constancy and uniform information density hypotheses (Genzel and Charniak, 2002; Jaeger and Levy, 2006; Xu and Reitter, 2018).

In order to enforce a limit on the entropy rate, we limit the surprisal of the next mention proposals under a mention model $p(x_t|x_1, \ldots, x_{t-1})$ (Justin: how to include partner mentions and confirmations?). We could utilize the belief state $p(z = x_t)$ for this model (Justin: include other history?); however, this model is not sensitive to the ordering of plans in h. We model mentions with an linearly decaying cache model, (Clarkson and Robinson, 1997):

$$\begin{aligned} &p_{\text{cache}}(x_T \mid h) \\ &\propto \prod_d e^{\beta m([x_T]_d, h) \mathbf{1}([x_T]_d \in h)} e^{\alpha \mathbf{1}([x_T]_d \notin h)}, \end{aligned}$$

where

$$m([x_T]_d, h) = T - \operatorname*{argmax}_t t \mathbf{1}([x_T]_d \in h_t),$$

and β and α are hyperparameters that control the cost of unmentioned dots and recency bias respectively.

We also experiment with a supervised neural model (Justin: include other history?).

F Partner confusion model

In the previous section, we introduced processing cost (as a channel capacity constraint) in the planning objective to limit the amount of information conveyed by an agent. This is an anti-causal approach: Adding processing cost to the objective does not model the effects of a plan. In this section we give a causal model of partner confusion as a result of processing cost.

The generative process for a response, given in Section B, is given by: given a plan x, resolve that plan to the shared dots z or unshared dots u. If the plan resolves to one or both, the partner gives a confirmation $y = \textsc{Yes}$. Unfortunately, this partner model assumes superhuman levels of resolution.

In our human studies, we found that certain plans are difficult for humans to resolve. Plans that involve contextually vague color or size descriptors, or dots that are too far apart result in the denial of a plan that should have been confirmed. Additionally, we believe some human players were frustrated by the information-dense descriptions of inhuman model plans, leading to poor success rates. In this section we describe several changes to the partner model that better reflect the behaviour of a human partner.

There are three desiderata:

1. Spatial locality: The partner will deny a plan if the dots are too far apart
2. Channel capacity: The partner will deny a plan if there is too much new information
3. Information locality: The partner has limited memory and will treat repeated old information as new information

We incorporate these desiderata by building them into the partner model. We utilize a product of experts formulation, where we combine the original partner resolution model with a confusion model. The goal of the confusion model is to model the effort a human partner would be willing to put into resolving a plan. The product of experts formulation allows the confusion model to vote against confirmation, resulting in the response model predicting denial if the plan is theoretically resolvable, but too confusing for a human to actually understand.

Let h be the history of all previous plans, with $[h]_t = x_t$. The full partner response model is given by

$$p(y \mid x, z, u, h) \propto \underbrace{p_r(y \mid x, z, u)}_{\text{resolution}} \underbrace{p_c(y \mid x, z, h)}_{\text{confusion}}.$$

The confusion model itself is a product of experts, consisting of spatial and temporal confusion models:

$$p_c(y \mid x, z, h) \propto \underbrace{p(y \mid x, z)}_{\text{spatial}} \underbrace{p(y \mid x, h)}_{\text{temporal}}.$$

F.1 Spatial model

For the spatial model, we assume a plan is more likely to be denied if the dots in the plan are far apart. A first approach would be to either use the distance between the furthest pair of dots, the sum of the pairwise distances between dots, or the area of the convex hull of the dots. However, as the distances of dots may vary, we instead use the relative rankings of dot distances. This mimics one possible method of dot resolution, where the partner first finds an anchor dot then searches nearby to find the remaining dots in the plan. As a result, we have

$$p(y = \textsc{No} \mid x) = \min_{d \in x} \frac{\sum_{d' \in x} r(d, d')}{1 + \sum_{i=7-|x|+1}^{7} i},$$

where $r(d, d')$ gives the rank of dot d' to d in order of increasing distance given the agent's perspective. nswer both time, and take that into account with the partner model.

G Supervised partner model

H Selection objective

I Describing plans

The success of an agent that plans through a partner model is limited by the accuracy of its partner model. While the partner model predicts the partner's reaction to the agent's plan, a human partner receives a natural language description of the agent's plan. For the partner model to be accurate, the plan description must be precise.

Preliminary experiments indicate that a model from previous work (Fried et al., 2021) loses precision when describing plans involving more than a single dot. We hypothesize that this is due to data imbalance.