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 constraints and grounding are key to successful contextual understanding. We address these shortcomings by embedding 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, incorporating grounding libraries. The translated code is executed 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, which involves grounding discussions on abstract images of scattered dots. In this challenging setting, human success rate is 65.8%. Our system demonstrates significant advancements in language understanding, as evidenced by strong results in human play, self-play, and overall performance.

1 Introduction

Topic sentence: Grounded dialogue requires a lot of generalization. Both grounding and lexical variation.

Topic sentence: Large language models have shortcomings, in particular grounding. Recent work has shown that tool use can adapt ungrounded language models to grounded tasks with minimal supervision. Describe code as policies, vipergpt, chatgpt plugins, which bake in task-specific biases.

Topic sentence: But Tool-use has mostly been demonstrated in the single-turn setting, other than the commercial success of ChatGPT. The challenge of grounded dialogue is dialogue state tracking and reasoning while grounding conversation. This is exacerbated by state tracking with weird grounding. Tie in hallucination, cite Deepmind safety paper Shaking the Foundations. Possible solution is by giving demonstrations. But encoding itself may be hard.

Topic sentence: We show that the classical dialogue system decomposition is still useful with LLMs. In particular, modularity allows for building in strong biases dealing with the encoding problem through code generation but integrating that into a symbolic planner. Method overview: read, plan, write. Cast reading as code generation in order to integrate with symbolic. Task-Oriented Dialogue as Dataflow Synthesis (SmCalFlow)

Topic sentence: We focus on OneCommon, a challenging grounded dialogue task. Description of what's hard about OneCommon. Also argue that in dialogue, data collection might be even harder. Results.

2 Related work

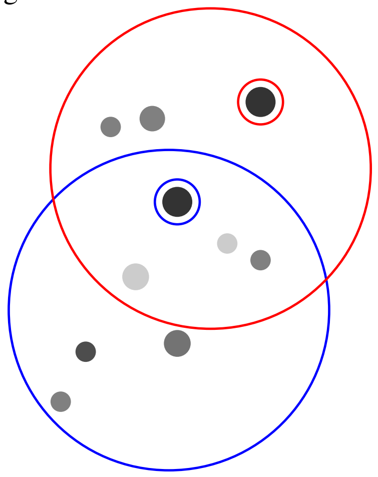
Modular dialogue systems SMCalFlow. Difference:

Tool-based language models Code-as-policies, ViperGPT, PAL, toolformer, VisProg. Difference:

Conversational tool-based language models VisualChatGPT, chatgpt with plugins. Difference:

Task-oriented dialogue 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.

grounding



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, shapes, 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.

3 Grounded dialogue with tool-based language models

We decompose dialogue into a sequence of three steps: reading, planning, and writing.

3.1 Reading

Language to python translation with execution.

Natural language understanding

Dialogue state tracking

3.2 Planning

Ask a new question, a follow-up question, or select.

Information gain heuristics

3.3 Writing

Realize the plan into language.

Simplicity and coreference

4 Prompt design

5 Experimental setup

We evaluate our method, REad-PlAn-Write (RePaW), on the OneCommon dataset (Udagawa and Aizawa, 2019). We compare approaches based on their success with human partners, as well as how they fare in selfplay. We offer additional ablations on prompt design, and share findings on practical aspects of prompt and code-based dialogue systems.

Systems We focus evaluation on the Partner Planner, which reasons about shared and unshared dots. We consider an ablated version, which does not model unshared dots.

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 model chooses plans based on the round-trip probability from an utterance back to the plan, which is an egocentric heuristic and also does not account for uncertainty over shared dots. 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.

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

	# shared			
Agent	4	5	6	Total
Partner Planner	26	12	5	43 (12)
-unshared	70	61	22	153 (22)
Fried et al. (2021)	28	17	4	49 (17)

Table 1: The number of incorrectly resolved plans in automatic feature-based evaluation on 519 static validation dialogues (2,341 turns). The total number of errors made in the first turn of a dialogue is shown in parentheses for each agent. A lower number of incorrect plans is better.

Agent	Success	Avg # turns
Partner planner	?	10
Partner Planner	85.2%	10.10
-unshared	77.0%	11.69
Fried et al. (2021)	62.4%	-
Human	65.8%	4.97

Table 2: The success rate of different agents in selfplay on the hardest setting of ONECOMMON, with 4 shared dots. The Partner Planner and ablated version communicate symbolically, while the Fried et al. (2021) baseline and human performance use language. A higher success rate is better. The human performance is from the ONECOMMON dataset (Udagawa and Aizawa, 2019).

feeding the feature representation of plans from the agent to the partner, who is a Partner Planner.

We generate plans given natural language dialogue history from a validation split of ONECOMMON following prior work (Fried et al., 2021).[1] 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.

6 Results

Human eval

Selfplay

[1]Prior work used 10-fold cross-validation. We use models from prior work trained on one fold, and evaluate our approach on the validation set for that fold. In particular, we use fold 1 from prior work.

Static reference resolution

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 1. 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 1.

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 2. 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.

7 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, Alane Suhr, Nicholas Tomlin.

References

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.

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

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.

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.

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.

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

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.

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[2] 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.

[2]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.

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 2. Note that a smaller number of shared dots increases the likelihood that plans get misresolved to unshared dots, increasing perspective-dependent ambiguity.

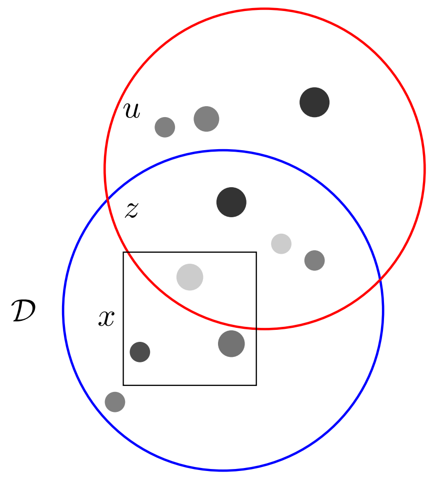
The agent communicates with the partner by producing an utterance plan, x, which it then describes in natural language. This plan is a subset of the dots in the agent view, $x \subseteq \mathcal{D}$, that the agent will ask the partner about. The partner gives a response y to the plan x, given their 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 2. Formally,

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



Do you have a triangle of one gray dot ...

$y = \text{No}$

Figure 2: 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.

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.[3]

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 $\mathcal{D}$, $O(2^{|\mathcal{D}|})$.

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:

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

Computing $p(y \mid x)$ has cost $O(2^{|\mathcal{D}|})$, while there are also $O(2^{|\mathcal{D}|})$ plans.[4] 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.

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

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

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 5.

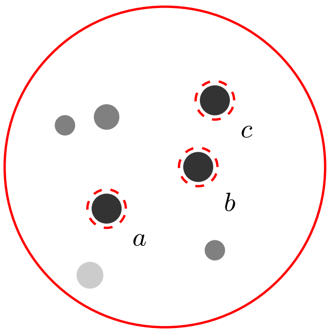
Selection To determine when to select a dot, the agent uses a threshold on the entropy $H[z]$, given by the hyperparameter τ. The agent them communicates which dot to select by describing the configuration of four dots with the highest marginal probability of being shared, as well as the dot within that 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 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 3.

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.



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

Figure 3: An example of feature-based resolution. The above feature representation for a pair of dots resolves to dot configurations $\{(a,b),(a,c),(b,c)\}$.

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):

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

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.