UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title

Learning Grounded Pragmatic Communication

Permalink

https://escholarship.org/uc/item/5vp9c5rc

Author

Fried, Daniel Patrick

Publication Date

2021

Peer reviewed|Thesis/dissertation

Learning Grounded Pragmatic Communication

by

Daniel Patrick Fried

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Dan Klein, Chair
Professor Trevor Darrell
Professor John DeNero
Professor Christopher Potts

Spring 2021

Learning Grounded Pragmatic Communication

Abstract

Learning Grounded Pragmatic Communication

by

Daniel Patrick Fried

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Dan Klein, Chair

This dissertation shows how language generation and interpretation across varied grounded domains can be improved through pragmatic inference: explicitly reasoning about the actions and intents of the people that the systems interact with. We train neural generation (*speaker*) and interpretation (*listener*) models which ground language into a world context, then layer a pragmatic inference procedure on top of these models. This pragmatic procedure predicts how human listeners will interpret text generated by the models and reasons counterfactually about why human speakers produced the text they did.

We begin by showing that explicit pragmatic inference aids in correctly generating and following natural language for complex, sequential grounded instruction tasks. Evaluation of language generation and interpretation shows that pragmatic inference improves state-of-the-art listener models (at correctly interpreting human instructions) and speaker models (at producing instructions correctly interpreted by humans).

Next, we describe extensions of this approach to vision-and-language navigation. We combine visually-grounded listener and speaker models, using the speaker model to synthesize new instructions for data augmentation in addition to evaluating candidate action sequences in pragmatic inference. Both models are supported by a panoramic action space that reflects the granularity of human-generated instructions. Experiments show that all three components of this approach—speaker-driven data augmentation, pragmatic inference and the panoramic action space—dramatically improve the performance of a baseline instruction follower, more than doubling the success rate over the best existing approach on a standard benchmark.

Finally, we present a grounded neural dialogue model that successfully collaborates with people in a partially-observable reference game. We focus on a setting where two agents each observe an overlapping part of a world context and need to identify and agree on some object they share. Therefore, the agents should pool their information and communicate pragmatically to solve the task. Our dialogue agent accurately grounds referents from the

partner's utterances using a structured reference resolver, conditions on these referents using a recurrent memory, and uses a pragmatic generation procedure to ensure the partner can resolve the references the agent produces.

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

Graduate school has been a collaborative multi-agent process (with many agents!), and I'm thankful to a ton of people for their help and support along the way:

Thanks to Dan Klein for being an fantastic advisor: teaching me to choose problems, distill ideas, and convey the things that matter. His advice was so helpful for me, even on non-academic things, that at various points I depended on it almost too much. Dan always seemed to know a way to improve anything, but over time he taught me to do this on my own and—just as importantly—convinced me that I could. I'm also grateful to Dan for putting my own interests ahead of his, for encouraging me when I needed it, and for going above and beyond on many occasions. I'm very thankful to have been part of the research group that he built.

I've also been fortunate to have a great committee. Much of my work, both in and out of this thesis, exists because of Trevor Darrell—thanks for inspiring and helping to hone the research, for well-timed encouragement and advice, and for believing in the work, and in me, even at times when I didn't. Thanks to John DeNero for being a dependable source of feedback and wisdom, for teaching me to teach, and for insights and interventions at crucial moments. Thanks to Chris Potts for teaching me almost everything that I know about pragmatics, for prompting me to think about meaning, and for advice and help on the next stage. I've also benefited enormously from mentorship from Aida Nematzadeh, Steve Clark, Jason Baldridge, and Hoifung Poon—thanks for inspiration on how to be effective and creative and do good science while working in new areas, for giving advice on how to find balance, and for being generous with your time and mentorship.

Thanks to my other collaborators—I've learned an incredible amount from all of you. Thanks to Jacob Andreas for getting me excited about most of the research paths I'm on, helping me to see the bigger picture, and basically being a second advisor in my early years; Ronghang Hu for showing me how to focus intently on the important things; Anja Rohrbach for helping me think critically and giving crucial and generous feedback; Justin Chiu for a more rigorous understanding of machine learning and for insights that spanned areas; Volkan Cirik for creativity that I hope to one day emulate; Mitchell Stern for tenacity and encouraging me to do things the right way; Nikita Kitaev for being fearless about building things and exploring bold ideas; David Gaddy for teaching me about unsupervised learning and asking the right questions; Rudy Corona for helping me think about abstraction; Jessy Lin for helping me think about interaction, and Nick Tomlin for teaching me the rest of what I know about pragmatics. A special thanks to Sheng Shen and Risham Sidhu for taking the lead as we explored new ground and putting up with my attempts to advise. Thanks too to the other residents of the NLP Bay over the years, who really enriched my time in grad school: Greg Durrett, Jonathan Kummerfeld, Max Rabinovich, Nick Altieri, Samee Ibraheem, Katie Stasaski, Cathy Chen, Kevin Lin, Kevin Yang, Eric Wallace, Ruiqi Zhong and Steven Cao.

For the work in this thesis, I'm also grateful to Andrea Daniele for sharing the SAIL simulator and their system outputs, to Hongyuan Mei for help with the SAIL dataset, to

Chapter 1

Introduction

Language is often strategic: it is an intentional action that people use to produce effects on others and the world (Austin, 1962). People choose what to say by predicting how listeners will interpret it and try to understand speakers by thinking about why they might have said what they did. A long line of work in linguistics, following Paul Grice (1975), views communication as a cooperative game where speakers and listeners try to understand each other in order to succeed. Many of the *pragmatic*, or context-dependent, features of language can be viewed as emerging from this cooperative principle.

In this dissertation, we operationalize a Gricean view of pragmatics and show that it allows natural language systems to more successfully collaborate with human partners on a variety of tasks in challenging contexts. Our systems learn models of their partners and use these models to reason about cooperative goals in a game-theoretic framework. We focus on complex *grounded* perceptual and embodied tasks, where systems must associate language with features and actions in an environment. For language generation tasks, we train listener models that predict how people will interpret language and use these models to make generated language truthful and relevant. For language understanding tasks, we show that speaker models of people's goals can help systems carry out instructions more accurately under ambiguity and challenging perceptual grounding. Constructing and training separate speaker and listener modules, which then reason about each other, makes them more likely to communicate successfully with people and do the right thing in context.

Our approach to pragmatics belongs to a general category of frameworks in which the interaction between speakers and listeners is modeled using an agent-based probabilistic (Rosenberg and Cohen, 1964; Frank and Goodman, 2012; Goodman and Stuhlmüller, 2013; Wang et al., 2016b; Monroe et al., 2017) or decision-theoretic treatment (Parikh, 2001; Golland et al., 2010; Jäger, 2012; Franke, 2013; Jeon et al., 2020). These models give rich accounts of context dependence and emergent pragmatic behavior (Goodman and Frank, 2016), but present challenging inference problems even when the space of listener actions is small, e.g., involving reference games with a limited set of possible referents (Smith et al., 2013; Khani et al., 2018; Andreas and Klein, 2016; Cohn-Gordon et al., 2018; McDowell and Goodman, 2019). Our goals in this work are to show how this inference procedure

(1) can be carried out even in much richer spaces of actions than previously considered in work on computational pragmatics and (2) can be layered on top of state-of-the-art natural language processing systems to improve their performance when collaborating with people on real-world grounded tasks.

We first demonstrate a pragmatic inference procedure for interpreting and generating instructions, i.e., descriptions of sequential actions (Chapter 2). We layer this procedure on top of learned neural grounding models, finding that the procedure makes models more successful when interacting with people, and in some cases outperforms the people whom the models were trained to imitate. Next, we show how this procedure can be extended to visually-rich environments with limited grounding data: following navigation instructions inside real-world buildings (Chapter 3). Finally, we show applications to a grounded collaborative dialogue task, where models must both generate and interpret language while dealing with challenges of partial observability and referential ambiguity (Chapter 4). In all settings, we find that pragmatic inference makes models more successful at generating language that can be successfully understood by people, and better able to interpret challenging or ambiguous grounded language.

Chapter 2

Generating and Following Instructions

In this chapter, we present a technique for layering explicit pragmatic inference on top of models for complex, sequential instruction-following and instruction-generation tasks.[1] We investigate a range of current data sets for both tasks, showing that pragmatic behavior arises naturally from this inference procedure and gives rise to state-of-the-art results in a variety of domains.

Consider the example shown in Figure 2.1a, in which a speaker agent must describe a route to a target position in a hallway. A conventional learned instruction-generating model produces a truthful description of the route (*walk forward four times*). But the pragmatic speaker in this chapter, which is capable of reasoning about the listener, chooses to also include additional information (*the intersection with the bare concrete hall*), to reduce potential ambiguity and increase the odds that the listener reaches the correct destination.

This same reasoning procedure also allows a listener agent to overcome ambiguity in instructions by reasoning counterfactually about the speaker (Figure 2.1b). Given the command *walk along the blue carpet and you pass two objects*, a conventional learned instruction-following model is willing to consider all paths that pass two objects, and ultimately arrives at an unintended final position. But a pragmatic listener that reasons about the speaker can infer that the long path would have been more easily described as *go to the sofa*, and thus that the shorter path is probably intended. In these two examples, which are produced by the system we describe in this chapter, a unified reasoning process (choose the output sequence which is most preferred by an embedded model of the other agent) produces pragmatic behavior for both speakers and listeners.

The application of models with explicit pragmatic reasoning abilities has so far been largely restricted to simple reference games, in which the listener's only task is to select the right item from among a small set of candidate referents given a single short utterance from the speaker. But as the example shows, there are real-world instruction following and generation tasks with rich action spaces that might also benefit from pragmatic modeling. Moreover, approaches that learn to map directly between human-annotated instructions

---

[1]This chapter is adapted from *Unified Pragmatic Models for Generating and Following Instructions*, (Fried et al., 2018a). Source code is available at `http://github.com/dpfried/pragmatic-instructions`.

Behavior

Base Speaker — *walk forward four times*

Rational Speaker — *go forward four segments to the intersection with the bare concrete hall*

Instruction — *walk along the blue carpet and you pass two objects*

Base Listener
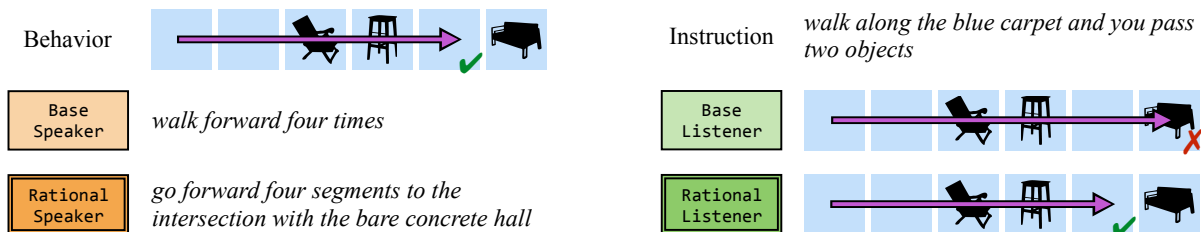
Rational Listener

(a) The rational speaker, which reasons about listener behavior, generates instructions which in this case are more robust to uncertainty about the listener's initial orientation, choosing to specify the destination (*the intersection with the bare concrete hall*).

(b) The base listener moves to an unintended position (even though it correctly *passes two objects*). The rational listener, which reasons about the speaker, infers that a route ending at the sofa would have been described differently, and stops earlier.

Figure 2.1: Real samples for the SAIL navigation environments, comparing *base* models, without explicit pragmatic inference, to the *rational* pragmatic inference procedure.

and action sequences are ultimately limited by the effectiveness of the humans themselves. The promise of pragmatic modeling is that we can use these same annotations to build a model with a different (and perhaps even better) mechanism for interpreting and generating instructions.

The primary contribution of this work is to show how existing models of pragmatic reasoning can be extended to support instruction following and generation for challenging, multi-step, interactive tasks. Our experimental evaluation focuses on four instruction-following domains which have been studied using both semantic parsers and attentional neural models. We investigate the interrelated tasks of instruction following and instruction generation, and show that incorporating an explicit model of pragmatics helps in both cases. Reasoning about the human listener allows a speaker model to produce instructions that are easier for humans to interpret correctly in all domains (with absolute gains in accuracy ranging from 12% to 46%). Similarly, reasoning about the human speaker improves the accuracy of the listener models in interpreting instructions in most domains (with gains in accuracy of up to 10%). In all cases, the resulting systems are competitive with, and in many cases exceed, results from past state-of-the-art systems for these tasks.

2.1 Problem Formulation

Consider the instruction following and instruction generation tasks shown in Figure 2.1, where an agent must produce or interpret instructions about a structured world context

(e.g. *walk along the blue carpet and you pass two objects*).

In the **instruction following** task, a listener agent begins in a world state (in Figure 2.1 an initial map location and orientation). The agent is then tasked with following a sequence of direction sentences $d_1 \ldots d_K$ produced by humans. At each time t the agent receives a percept y_t, which is a feature-based representation of the current world state, and chooses an action a_t (e.g. move forward, or turn). The agent succeeds if it is able to reach the correct final state described by the directions.

In the **instruction generation** task, the agent receives a sequence of actions $a_1, \cdots a_T$ along with the world state $y_1, \cdots y_T$ at each action, and must generate a sequence of direction sentences $d_1, \ldots d_K$ describing the actions. The agent succeeds if a human listener is able to correctly follow those directions to the intended final state.

We evaluate models for both tasks in four domains. The first domain is the SAIL corpus of virtual environments and navigational directions (MacMahon et al., 2006; Chen and Mooney, 2011), where an agent navigates through a two-dimensional grid of hallways with patterned walls and floors and a discrete set of objects (Figure 2.1 shows a portion of one of these hallways).

In the three SCONE domains (Long et al., 2016), the world contains a number of objects with various properties, such as colored beakers which an agent can combine, drain, and mix. Instructions describe how these objects should be manipulated. These domains were designed to elicit instructions with a variety of context-dependent language phenomena, including ellipsis and coreference (Long et al., 2016) which we might expect a model of pragmatics to help resolve (Potts, 2011).

2.2 Related Work

The approach in this chapter builds upon long lines of work in pragmatic modeling, instruction following, and instruction generation.

Pragmatics. Our approach to pragmatics belongs to a general category of Rational Speech Acts models (Frank and Goodman, 2012), in which the interaction between speakers and listeners is modeled as a probabilistic process with Bayesian actors (Goodman and Stuhlmüller, 2013). Alternative formulations (e.g. with best-response rather than probabilistic dynamics) are also possible (Golland et al., 2010). Inference in these models is challenging even when the space of listener actions is extremely simple (Smith et al., 2013), and one of our goals in the present work is to show how this inference problem can be solved even in much richer action spaces than previously considered in computational pragmatics. This family of pragmatic models captures a number of important linguistic phenomena, especially those involving conversational implicature (Monroe and Potts, 2015); we note that many other topics studied under the broad heading of "pragmatics," including presupposition and indexicality, require different machinery.

(a) (b)
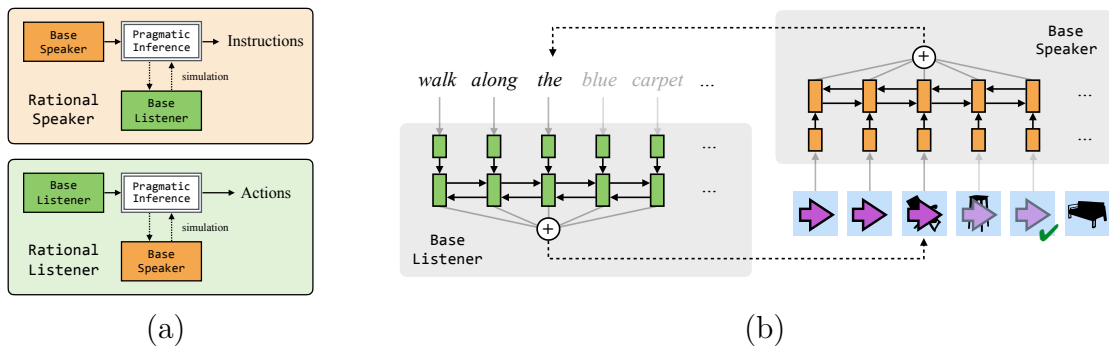
Figure 2.2: (a) Rational pragmatic models embed base listeners and speakers. Potential candidate sequences are drawn from one base model, and then the other scores each candidate to simulate whether it produces the desired pragmatic behavior. (b) The base listener and speaker are neural sequence-to-sequence models which are largely symmetric to each other. Each produces a representation of its input sequence (a description, for the listener; actions with associated environmental percepts, for the listener) using an LSTM encoder. The output sequence is generated by an LSTM decoder attending to the input.

Williams et al. (2015) use pragmatic reasoning with weighted inference rules to resolve ambiguity and generate clarification requests in a human-robot dialog task. Other recent work on pragmatic models focuses on the referring expression generation or "contrastive captioning" task introduced by Kazemzadeh et al. (2014). In this family are approaches that model the listener at training time (Mao et al., 2016), at evaluation time (Andreas and Klein, 2016; Monroe et al., 2017; Vedantam et al., 2017; Su et al., 2017) or both (Yu et al., 2017b; Luo and Shakhnarovich, 2017).

Other conditional sequence rescoring models that are structurally similar but motivated by concerns other than pragmatics include Li et al. (2016) and Yu et al. (2017a). Lewis et al. (2017) perform a similar inference procedure for a competitive negotiation task. The language learning model of Wang et al. (2016b) also features a structured output space and uses pragmatics to improve online predictions for a semantic parsing model. Our approach in this chapter performs both generation and interpretation, and investigates both structured and unstructured output representations.

Instruction Following. Work on instruction following tasks includes models that parse commands into structured representations processed by a rich execution model (Tellex et al., 2011; Chen, 2012; Artzi and Zettlemoyer, 2013; Guu et al., 2017) as well as models that map directly from instructions to a policy over primitive actions (Branavan et al., 2009), possibly mediated by an intermediate alignment or attention variable (Andreas and Klein, 2015; Mei et al., 2016). We use a model similar to Mei et al. (2016) as our base listener in this chapter, evaluating on the SAIL navigation task (MacMahon et al., 2006) as they did, as well as the SCONE context-dependent execution domains (Long et al., 2016).

Instruction Generation. Previous work has also investigated the instruction generation task, in particular for navigational directions. The GIVE shared tasks (Byron et al., 2009; Koller et al., 2010; Striegnitz et al., 2011) have produced a large number of interactive direction-giving systems, both rule-based and learned. The work most immediately related to the generation task in this chapter is that of Daniele et al. (2017), which also focuses on the SAIL dataset but requires substantial additional structured annotation for training, while both our base and pragmatic speaker models learn directly from strings and action sequences.

Older work has studied the properties of effective human strategies for generating navigational directions (Anderson et al., 1991). Instructions of this kind can be used to extract templates for generation (Look, 2008; Dale et al., 2005), while here we focus on the more challenging problem of learning to generate new instructions from scratch. Like our pragmatic speaker model, Goeddel and Olson (2012) also reason about listener behavior when generating navigational instructions, but rely on rule-based models for interpretation.

2.3 Pragmatic Inference Procedure

As a foundation for pragmatic inference, we assume that we have *base* listener and speaker models to map directions to actions and vice-versa. (Our notation for referring to models is adapted from Bergen et al. (2016).) The base listener, L_0, produces a probability distribution over sequences of actions, conditioned on a representation of the directions and environment as seen before each action: $P_{L_0}(a_{1:T}|d_{1:K}, y_{1:T})$. Similarly, the base speaker, S_0, defines a distribution over possible descriptions conditioned on a representation of the actions and environment: $P_{S_0}(d_{1:K}|a_{1:T}, y_{1:T})$.

Our pragmatic inference procedure requires these base models to produce candidate outputs from a given input (actions from descriptions, for the listener; descriptions from actions, for the speaker), and calculate the probability of a fixed output given an input, but is otherwise agnostic to the form of the models.

We use standard sequence-to-sequence models with attention for both the base listener and speaker (described in Section 2.4). Our models use segmented action sequences, with one segment (sub-sequence of actions) aligned with each description sentence d_j, for all $j \in \{1 \ldots K\}$. This segmentation is either given as part of the training and testing data (in the instruction following task for the SAIL domain, and in both tasks for the SCONE domain, where each sentence corresponds to a single action), or is predicted by a separate segmentation model (in the generation task for the SAIL domain), see Section 2.4.

Models

Using these base models as self-contained modules, we derive a *rational speaker* and *rational listener* that perform inference using embedded instances of these base models (Figure 2.2a). When describing an action sequence, a rational speaker S_1 chooses a description that has a

high chance of causing the listener modeled by L_0 to follow the given actions:

$$S_1(a_{1:T}) = \arg\max_{d_{1:K}} P_{L_0}(a_{1:T}|d_{1:K}, y_{1:T}) \tag{2.1}$$

(noting that, in all settings we explore here, the percepts $y_{1:T}$ are completely determined by the actions $a_{1:T}$). Conversely, a rational listener L_1 follows a description by choosing an action sequence which has high probability of having caused the speaker, modeled by S_0, to produce the description:

$$L_1(d_{1:K}) = \arg\max_{a_{1:T}} P_{S_0}(d_{1:K}|a_{1:T}, y_{1:T}) \tag{2.2}$$

These optimization problems are intractable to solve for general base listener and speaker agents, including the sequence-to-sequence models we use, as they involve choosing an input (from a combinatorially large space of possible sequences) to maximize the probability of a fixed output sequence. We instead follow a simple approximate inference procedure, detailed in Section 2.3.

We consider also incorporating the scores of the base model used to produce the candidates. For the case of the speaker, we define a *combined rational speaker*, denoted $S_0 \cdot S_1$, that selects the candidate that maximizes a weighted product of probabilities under both the base listener and the base speaker:

$$\arg\max_{d_{1:K}} P_{L_0}(a_{1:T}|d_{1:K}, y_{1:T})^\lambda$$
$$\times \ P_{S_0}(d_{1:K}|a_{1:T}, y_{1:T})^{1-\lambda} \tag{2.3}$$

for a fixed interpolation hyperparameter $\lambda \in [0, 1]$. There are several motivations for this combination with the base speaker score. First, as argued by Monroe et al. (2017), we would expect varying degrees of base and reasoned interpretation in human speech acts. Second, we want the descriptions produced by the model to be fluent descriptions of the actions. Since the base models are trained discriminatively, maximizing the probability of an output sequence for a fixed input sequence, their scoring behaviors for fixed outputs paired with inputs dissimilar to those seen in the training set may be poorly calibrated (for example when conditioning on ungrammatical descriptions). Incorporating the scores of the base model used to produce the candidates aims to prevent this behavior.

To define rational listeners, we use the symmetric formulation: first, draw candidate action sequences from L_0. For L_1, choose the actions that achieve the highest probability under S_0; and for the combination model $L_0 \cdot L_1$ choose the actions with the highest weighted combination of S_0 and L_0 (paralleling Equation 2.3).

Inference

As in past work (Smith et al., 2013; Andreas and Klein, 2016; Monroe et al., 2017), we approximate the optimization problems in Equations 2.1, 2.2, and 2.3: use the base models

to generate candidates, and rescore them to find ones that are likely to produce the desired behavior.

In the case of the rational speaker S_1, we use the base speaker S_0 to produce a set of n candidate descriptions $w_{1:K_1}^{(1)} \ldots w_{1:K_n}^{(n)}$ for the sequences $a_{1:T}, y_{1:T}$, using beam search. We then find the score of each description under P_{L_0} (using it as the input sequence for the observed output actions we want the rational speaker to describe), or a weighted combination of P_{L_0} and the original candidate score P_{S_0}, and choose the description $w_{1:K_j}^{(j)}$ with the largest score, approximately solving the maximizations in Equations 2.1 or 2.3, respectively. We perform a symmetric procedure for the rational listener: produce action sequence candidates from the base listener, and rescore them using the base speaker.[2]

As the rational speaker must produce long output sequences (with multiple sentences), we interleave the speaker and listener in inference, determining each output sentence sequentially. From a list of candidate direction sentences from the base speaker for the current subsequence of actions, we choose the top-scoring direction under the listener model (which may also condition on the directions which have been output previously), and then move on to the next subsequence of actions.[3]

2.4   Base Model Details

Given this framework, all that remains is to describe the base models L_0 and S_0. We implement these as sequence-to-sequence models that map directions to actions (for the listener) or actions to directions (for the speaker), additionally conditioning on the world state at each timestep.

Base listener

Our base listener model, L_0, predicts action sequences conditioned on an encoded representation of the directions and the current world state. In the SAIL domain, this is the model of Mei et al. (2016) (illustrated in green in Figure 2.2b for a single sentence and its associated actions), see "domain specifics" below.

Encoder.   Each direction sentence is encoded separately with a bidirectional LSTM (Hochreiter and Schmidhuber, 1997); the LSTM's hidden states are reset for each sentence. We obtain a representation h_k^e for the kth word in the current sentence by concatenating an embedding for the word with its forward and backward LSTM outputs.

---

[2]We use ensembles of models for the base listener and speaker. To obtain candidates that are high-scoring under the combination of models in the ensemble, we perform standard beam search using all models in lock-step. At every timestep of the beam search, each possible extension of an output sequence is scored using the product of the extension's conditional probabilities across all models in the ensemble.

[3]We also experimented with sampling from the base models to produce these candidate lists, as was done in previous work (Andreas and Klein, 2016; Monroe et al., 2017). In early experiments, however, we found better performance with beam search in the rational models for all tasks.

Decoder. We generate actions incrementally using an LSTM decoder with monotonic alignment between the direction sentences and subsequences of actions; at each timestep the decoder predicts the next action for the current sentence $w_{1:M}$ (including choosing to shift to the next sentence). The decoder takes as input at timestep t the current world state, y_t and a representation z_t of the current sentence, updates the decoder state h^d, and outputs a distribution over possible actions:

$$h_t^d = \text{LSTM}_d(h_{t-1}^d, [W_y y_t, z_t])$$
$$q_t = W_o(W_y y_t + W_h h_t^d + W_z z_t)$$
$$p(a_t \mid a_{1:t-1}, y_{1:t}, w_{1:M}) \propto \exp(q_t)$$

where all weight matrices W are learned parameters. The sentence representation z_t is produced using an attention mechanism (Bahdanau et al., 2015) over the representation vectors $h_1^e \dots h_M^e$ for words in the current sentence:

$$\alpha_{t,k} \propto \exp(v \cdot \tanh(W_d h_{t-1}^d + W_e h_k^e))$$
$$z_t = \sum_{k=1}^{M} \alpha_{t,k} h_k^e$$

where the attention weights $\alpha_{t,k}$ are normalized to sum to one across positions k in the input, and weight matrices W and vector v are learned.

SAIL Domain Specifics.   For SAIL, we use the alignments between sentences and route segments annotated by Chen and Mooney (2011), which were also used in previous work (Artzi and Zettlemoyer, 2013; Artzi et al., 2014; Mei et al., 2016). Following Mei et al. (2016), we reset the decoder's hidden state for each sentence.

SCONE Domain Specifics.   The SCONE dataset is constructed so that each sentence corresponds to a single (non-decomposed) action; this provides our segmentation of the action sequence. Each of the three SCONE domains have a larger space of possible outputs than SAIL, so we extend the decoder by: (i) decomposing each action into an action type and arguments for it, (ii) using separate attention mechanisms for types and arguments and (iii) using state-dependent action embeddings.

We factor action production in each of the three domains, separately predicting the action type and the arguments specific to that action type. Action types and arguments are listed in the first two columns of Table 2.1. For example, Alchemy's actions involve predicting the action type, a potential source beaker index i and target beaker index j, and potential amount to drain a. All factors of the action (the type and options for each argument) are predicted using separate attention mechanisms, which produce a vector q_f giving unnormalized scores for factor f (e.g. scoring each possible type, or each possible choice for the argument).

We also obtain state-specific embeddings of actions, to make it easier for the model to learn relevant features from the state embeddings (e.g. rather than needing to learn to

type	arguments	contextual embedding
Alchemy		
MIX	source i	contents of i
POUR	source i, target j	contents of i and j
DRAIN	amount a, source i	a, contents of i
Scene		
ENTER	color c, source i	people at $i-1$ and $i+1$
EXIT	source i	people at i, $i-1$, $i+1$
MOVE	source i, target j	people at i, $j-1$, $j+1$
SWITCH	source i, target j	people at i and j
TAKEHAT	source i, target j	people at i and j
Tangrams		
REMOVE	position i	—
SWAP	positions i and j	—
INSERT	position i, shape s	index of step when s was removed

Table 2.1: Action types, arguments, and elements of the world state or action history that are extracted to produce contextual action embeddings.

select the region of the state vector corresponding to the 5th beaker in the action MIX(5) in Alchemy, this action's contextual embedding encodes the current content of the 5th beaker). We incorporate these state-specific embeddings into computation of the action probabilities using a bilinear bonus score:

$$b(a) = q^\top W_{qa} a + w_a^\top a$$

where q is the concatenation of all q_f factor scoring vectors, and W_{qa} and w_a are a learned parameter matrix and vector, respectively. This bonus score $b(a)$ for each action is added to the unnormalized score for the corresponding action a (computed by summing the entries of the q_f vectors which correspond to the factored action components), and the normalized output distribution is then produced using a softmax over all valid actions.

Base speaker

While previous work (Daniele et al., 2017) has relied on more structured approaches, we construct our base speaker model S_0 using largely the same sequence-to-sequence machinery as above. S_0 (illustrated in orange in Figure 2.2b) encodes a sequence of actions and world states, and then uses a decoder to output a description.

Encoder.   We encode the sequence of vector embeddings for the actions a_t and world states y_t using a bidirectional LSTM. Similar to the base listener's encoder, we then obtain a

representation h_t^e for timestep t by concatenating a_t and y_t with the LSTM outputs at that position.

Decoder. As in the listener, we use an LSTM decoder with monotonic alignment between direction sentences and subsequences of actions, and attention over the subsequences of actions. The decoder takes as input at position k an embedding for the previously generated word w_{k-1} and a representation z_k of the current subsequence of actions and world states, and produces a distribution over words (including ending the description for the current subsequence and advancing to the next). The decoder's output distribution is produced by:

$$h_k^d = \text{LSTM}_d(h_{k-1}^d, [w_{k-1}, z_k])$$
$$q_k = W_h h_k^d + W_z z_k$$
$$p(w_k \mid w_{1:k-1}, a_{1:T}, y_{1:T}) \propto \exp(q_k)$$

where all weight matrices W are learned parameters.[4] As in the base listener, the input representation z_k is produced by attending to the vectors $h_1^e \ldots h_T^e$ encoding the input sequence (here, encoding the subsequence of actions and world states to be described):

$$\alpha_{k,t} \propto \exp(v \cdot \tanh(W_d h_{k-1}^d + W_e h_t^e))$$
$$z_k = \sum_{t=1}^{T} \alpha_{k,t} \, h_t^e$$

The decoder's LSTM state is reset at the beginning of each sentence.

SAIL Domain Specifics. Our speaker model operates on segmented action sequences. To provide a closer comparison to the generation system of Daniele et al. (2017), which did not use segmented routes, we implement a route segmenter which we train on the training data and use to predict segmentations for the test data. The route segmenter runs a bidirectional LSTM over the concatenated state and action embeddings (as in the speaker encoder), then uses a logistic output layer to classify whether the route should be split at each possible timestep.

We also collapse consecutive sequences of forward movement actions into single actions (e.g. MOVE4 representing four consecutive forward movements), which we found helped prevent counting errors (such as outputting *move forward three* when the correct route moved forward four steps).

SCONE Domain Specifics. In the SCONE domains, we (i) use the same context-dependent action embeddings used in the listener, and (ii) don't require an attention mechanism, since only a single action is used to produce a given sentence within the sequence of direction sentences.

---

[4]All parameters are distinct from those used in the base listener; the listener and speaker are trained separately.

model	domain	dropout rate	hidden dim	attention dim
L_0	SAIL	0.25	100	100
L_0	Alchemy	0.1	50	50
L_0	Scene	0.1	100	100
L_0	Tangrams	0.3	50	100
S_0	SAIL	0.25	100	100
S_0	Alchemy	0.3	100	–
S_0	Scene	0.3	100	–
S_0	Tangrams	0.3	50	–

Table 2.2: Hyperparameters for the base listener (L_0) and speaker (S_0) models. The SCONE speakers do not use an attention mechanism.

We use a one-hot representation of the arguments and contextual embedding (see Table 2.1) for each action a_t as input to the SCONE speaker encoder at time t (along with the representation e_t of the world state, as in SAIL). Since SCONE uses a monotonic, one-to-one alignment between actions and direction sentences, the decoder does not use a learned attention mechanism but fixes the contextual representation z_k to be the encoded vector at the action corresponding to the sentence currently being generated.

Training Details

The base listener and speaker models are trained independently to maximize the conditional likelihoods of the actions–directions pairs in the training sets.

We use ensembles for the base listener L_0 and base speaker S_0, where each ensemble consists of 10 models trained from separate random parameter initializations. This follows the experimental setup of Mei et al. (2016) for the SAIL base listener.

We optimize model parameters using ADAM (Kingma and Ba, 2015) with default hyperparameters and the initialization scheme of Glorot and Bengio (2010). All LSTMs have one layer. The LSTM cell in both the listener and the follower use coupled input and forget gates, and peephole connections to the cell state (Greff et al., 2017). We also apply the LSTM variational dropout scheme of Gal and Ghahramani (2016), using the same dropout rate for inputs, outputs, and recurrent connections. See Table 2.2 for hyperparameters. We perform early stopping using the evaluation metric (accuracy for the listener and BLEU score for the speaker) on the development set.

| | Single-sentence | | Multi-sentence | |
listener	Rel	Abs	Rel	Abs
past work	69.98 (MBW)	65.28 (AZ)	26.07 (MBW)	35.44 (ADP)
L_0	68.40	59.62	24.79	13.53
$L_0 \cdot L_1$	**71.64**	64.38	**34.05**	24.50
accuracy gain	+3.24	+4.76	+9.26	+10.97

Table 2.3: Instruction-following results on the SAIL dataset. The table shows cross-validation test accuracy for the base listener (L_0) and pragmatic listeners ($L_0 \cdot L_1$), along with the gain given by pragmatics. We report results for the single- and multi-sentence conditions, under the relative and absolute starting conditions,[5] comparing to the best-performing prior work by Artzi and Zettlemoyer (2013) (AZ), Artzi et al. (2014) (ADP), and Mei et al. (2016) (MBW). Bold numbers show new state-of-the-art results.

2.5 Experiments

We evaluate speaker and listener agents on both the instruction following and instruction generation tasks in the SAIL domain and three SCONE domains (Section 2.1). For all domains, we compare the rational listener and speaker against the base listener and speaker, as well as against past state-of-the-art results for each task and domain. Finally, we examine pragmatic inference from a model combination perspective, comparing the pragmatic reranking procedure to ensembles of a larger number of base speakers or listeners.

For all experiments, we use beam search both to generate candidate lists for the rational systems (Section 2.3) and to generate the base model's output. We fix the beam size n to be the same in both the base and rational systems, using $n = 20$ for the speakers and $n = 40$ for the listeners. We tune the weight λ in the combined rational agents ($L_0 \cdot L_1$ or $S_0 \cdot S_1$) to maximize accuracy (for listener models) or BLEU (for speaker models) on each domain's development data.

Instruction Following

We evaluate our listener models by their accuracy in carrying out human instructions: whether the systems were able to reach the final world state which the human was tasked with guiding them to.

SAIL. We follow standard cross-validation evaluation for the instruction following task on the SAIL dataset (Artzi and Zettlemoyer, 2013; Artzi et al., 2014; Mei et al., 2016).[5] Table 2.3

---

[5]Past work has differed in the handling of undetermined orientations in the routes, which occur in the first state for multi-sentence routes and the first segment of their corresponding single-sentence routes. For

listener	Alchemy	Scene	Tangrams
GPLL	52.9	46.2	37.3
L_0	69.7	70.9	69.6
$L_0 \cdot L_1$	72.0	72.7	69.6
accuracy gain	+2.3	+1.8	+0.0

Table 2.4: Instruction-following results in the SCONE domains. The table shows accuracy on the test set. For reference, we also show prior results from Guu et al. (2017) (GPLL), although our models use more supervision at training time.

shows improvements over the base listener L_0 when using the rational listener $L_0 \cdot L_1$ in the single- and multi-sentence settings. We also report the best accuracies from past work. We see that the largest relative gains come in the multi-sentence setting, where handling ambiguity is potentially more important to avoid compounding errors. The rational model improves on the published results of Mei et al. (2016), and while it is still below the systems of Artzi and Zettlemoyer (2013) and Artzi et al. (2014), which use additional supervision in the form of hand-annotated seed lexicons and logical domain representations, it approaches their results in the single-sentence setting.

SCONE. In the SCONE domains, past work has trained listener models with weak supervision (with no intermediate actions between start and end world states) on a subset of the full SCONE training data. We use the full training set, and to use a model and training procedure consistent with the SAIL setting, train listener and speaker models using the intermediate actions as supervision as well.[6] The evaluation method and test data are the same as in past work on SCONE: models are provided with an initial world state and a sequence of 5 instructions to carry out, and are evaluated on their accuracy in reaching the intended final world state.

Results are reported in Table 2.4. We see gains from the rational system $L_0 \cdot L_1$ in both the Alchemy and Scene domains. The pragmatic inference procedure allows correcting errors or overly-literal interpretations from the base listener. An example is shown in Figure 2.3. The base listener (left) interprets *then to orange's other side* incorrectly, while the rational listener discounts this interpretation (it could, for example, be better described by *to the left of blue*) and produces the action the descriptions were meant to describe (right). To

---

comparison to both types of past work, we train and evaluate listeners in two settings: *Abs*, which sets these undetermined starting orientations to be a fixed absolute orientation, and *Rel*, where an undetermined starting orientation is set to be a 90 degree rotation from the next state in the true route.

[6]Since the pragmatic inference procedure we use is agnostic to the models' training method, it could also be applied to the models of Guu et al. (2017); however we find that pragmatic inference can improve even upon our stronger base listener models.

```
a red guy appears on the far left
then to orange's other side
```

base listener, L_0 rational listener, $L_0 \cdot L_1$
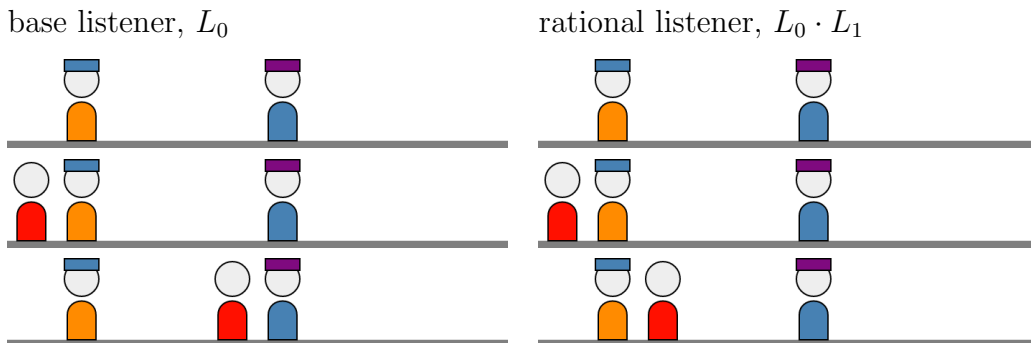
Figure 2.3: Action traces produced for a partial instruction sequence (two instructions out of five) in the Scene domain. The base listener moves the red figure to a position that is a marginal, but valid, interpretation of the directions. The rational listener correctly produces the action sequence the directions were intended to describe.

the extent that human annotators already account for pragmatic effects when generating instructions, examples like these suggest that our model's explicit reasoning is able to capture interpretation behavior that the base sequence-to-sequence listener model is unable to model.

Instruction Generation

As our primary evaluation for the instruction generation task, we had Mechanical Turk workers carry out directions produced by the speaker models (and by other humans) in a simulated version of each domain. For SAIL, we use the simulator released by Daniele et al. (2017) which was used in their human evaluation results, and we construct simulators for the three SCONE domains. In all settings, we take a sample of 50 action sequences from the domain's test set (using the same sample as Daniele et al. (2017) for SAIL), and have three separate Turk workers attempt to follow the systems' directions for the action sequence.

Table 2.5 gives the average accuracy of subjects in reaching the intended final world state across all sampled test instances, for each domain. The "human-generated" row reports subjects' accuracy at following the datasets' reference directions. The directions produced by the base speaker S_0 are often much harder to follow than those produced by humans (e.g. 29.3% of S_0's directions are correctly interpretable for Alchemy, vs. 83.3% of human directions). However, we see substantial gains from the rational speaker $S_0 \cdot S_1$ over S_0 in all cases (with absolute gains in accuracy ranging from 12.4% to 46.0%), and the average accuracy of humans at following the rational speaker's directions is substantially higher than for human-produced directions in the Tangrams domain. In the SAIL evaluation, we also

speaker	SAIL	Alchemy	Scene	Tangrams
DBW	70.9	—	—	—
S_0	62.8	29.3	31.3	60.0
$S_0 \cdot S_1$	**75.2**	**75.3**	**69.3**	**88.0**
accuracy gain	+12.4	+46.0	+38.0	+28.0
human-generated	73.2	83.3	78.0	66.0

Table 2.5: Instruction generation results. We report the accuracies of human evaluators at following the outputs of the speaker systems (as well as other humans) on 50-instance samples from the SAIL dataset and SCONE domains. DBW is the system of Daniele et al. (2017). Bold numbers are new state-of-the-art results.

speaker	SAIL	Alchemy	Scene	Tangrams
DBW	11.00	—	—	—
S_0	12.04	19.34	18.09	21.75
$S_0 \cdot S_1$	10.78	18.70	27.15	23.03
BLEU gain	-1.26	-0.64	+9.06	+1.28
accuracy gain (from Table 2.5)	+12.4	+46.0	+38.0	+28.0

Table 2.6: Gains in how easy the directions are to follow are not always associated with a gain in BLEU. This table shows corpus-level 4-gram BLEU comparing outputs of the speaker systems to human-produced directions on the SAIL dataset and SCONE domains, compared to gains in accuracy when asking humans to carry out a sample of the systems' directions (see Table 2.5).

include the directions produced by the system of Daniele et al. (2017) (DBW), and find that the rational speaker's directions are followable to comparable accuracy.

In Table 2.6, we also compare the directions produced by the systems to the reference instructions given by humans in the dataset, using 4-gram BLEU (Papineni et al., 2002).[7] Consistent with past work (Krahmer and Theune, 2010), we find that BLEU score is a poor indicator of whether the directions can be correctly followed.

---

[7]To compute BLEU in the SAIL experiments, as the speaker models may choose produce a different number of sentences for each route than in the true description, we obtain a single sequence of words from a multi-sentence description produced for a route by concatenating the sentences, separated by end-of-sentence tokens. We then calculate corpus-level 4-gram BLEU between all these sequences in the test set and the true multi-sentence descriptions (concatenated in the same way).

human	`take away the last item` `undo the last step`
S_0	`remove the last figure` `add it back`
$S_0 \cdot S_1$	`remove the last figure` `add it back in the 3rd position`
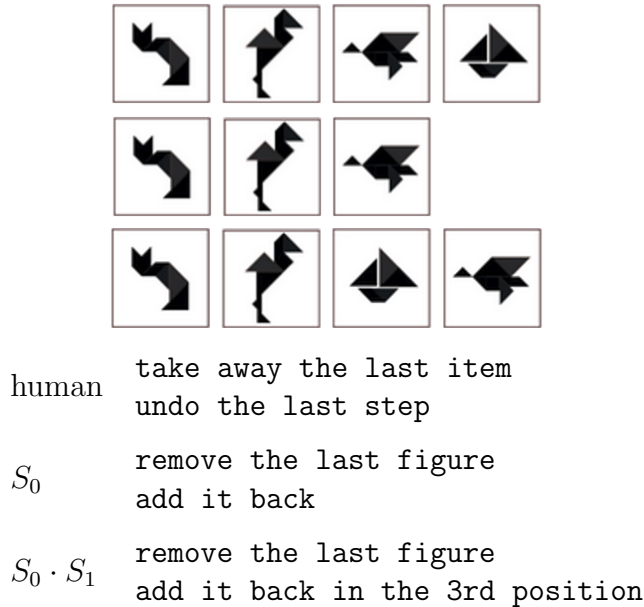
Figure 2.4: Descriptions produced for a partial action sequence in the Tangrams domain. Neither the human nor base speaker S_0 correctly specifies where to add the shape in the second step, while the rational speaker $S_0 \cdot S_1$ does.

Qualitatively, the rational inference procedure is most successful in fixing ambiguities in the base speaker model's descriptions. Figure 2.4 gives a typical example of this for the last few timesteps from a Tangrams instance. The base speaker correctly describes that the shape should be added back, but does not specify where to add it, which could lead a listener to add it in the same position it was deleted. The human speaker also makes this mistake in their description. This speaks to the difficulty of describing complex actions pragmatically even for humans in the Tangrams domain. The ability of the pragmatic speaker to produce directions that are easier to follow than humans' in this domain (Table 2.5) shows that the pragmatic model can generate something different (and in some cases better) than the training data.

Pragmatics as Model Combination

Finally, our rational models can be viewed as pragmatically-motivated model combinations, producing candidates using base listener or speaker models and reranking using a combination of scores from both. We want to verify that a rational listener using n ensembled base listeners and n base speakers outperforms a simple ensemble of $2n$ base listeners (and similarly for the rational speaker).

Fixing the total number of models to 20 in each listener experiment, we find that the

rational listener (using an ensemble of 10 base listener models and 10 base speaker models) still substantially outperforms the ensembled base listener (using 20 base listener models): accuracy gains are $68.5 \rightarrow 71.6\%$, $70.1 \rightarrow 72.0\%$, $71.9 \rightarrow 72.7\%$, and $69.1 \rightarrow 69.6\%$ for SAIL single-sentence Rel, Alchemy, Scene, and Tangrams, respectively.

For the speaker experiments, fixing the total number of models to 10 (since inference in the speaker models is more expensive than in the follower models), we find similar gains as well: the rational speaker improves human accuracy at following the generated instructions from $61.9 \rightarrow 73.4\%$, $30.7 \rightarrow 74.7\%$, $32.0 \rightarrow 66.0\%$, $58.7 \rightarrow 92.7\%$, for SAIL, Alchemy, Scene, and Tangrams, respectively.[8]

2.6   Discussion

We have demonstrated that a simple procedure for pragmatic inference, with a unified treatment for speakers and listeners, obtains improvements for instruction following as well as instruction generation in multiple settings. The inference procedure is capable of reasoning about sequential, interdependent actions in non-trivial world contexts. We find that pragmatics improves upon the performance of the base models for both tasks, in most cases substantially. While this is perhaps unsurprising for the generation task, which has been discussed from a pragmatic perspective in a variety of recent work in NLP, it is encouraging that pragmatic reasoning can also improve performance for a grounded listening task with sequential, structured output spaces.

---

[8]The accuracies for the base speakers are slightly different than in Table 2.5, despite being produced by the same systems, since we reran experiments to control as much as possible for time variation in the pool of Mechanical Turk workers.

Chapter 3

Vision-and-Language Navigation

In the vision-and-language navigation task (Anderson et al., 2018), an agent is placed in a realistic environment and provided a natural language instruction such as "*Go down the stairs, go slight left at the bottom and go through door, take an immediate left and enter the bathroom, stop just inside in front of the sink*". The agent must follow this instruction to navigate from its starting location to a goal location, as shown in Figure 3.1 (left). To accomplish this task the agent must learn to relate the language instructions to the visual environment. Moreover, it should be able to carry out new instructions in unseen environments.

Even simple navigation tasks require nontrivial *reasoning*: the agent must resolve ambiguous references to landmarks, perform a counterfactual evaluation of alternative routes, and identify incompletely specified destinations. While a number of approaches (Mei et al., 2016; Misra et al., 2017; Wang et al., 2018) have been proposed for the various navigation benchmarks, they generally employ a single model that learns to map directly from instructions to actions from a limited corpus of annotated trajectories.

In this chapter we treat the vision-and-language navigation task as a trajectory search problem, where the agent needs to find (based on the instruction) the best trajectory in the environment to navigate from the start location to the goal location.[1] Our model involves an instruction interpretation (*follower*) module, mapping instructions to action sequences; and an instruction generation (*speaker*) module, mapping action sequences to instructions (Figure 3.1), both implemented with standard sequence-to-sequence architectures. The speaker learns to give textual instructions for visual routes, while the follower learns to follow routes (predict navigation actions) for provided textual instructions. Though explicit probabilistic reasoning combining speaker and follower agents is a staple of the literature on computational pragmatics (Frank and Goodman, 2012), application of these models has largely been limited to extremely simple decision-making tasks like single forced choices.

We incorporate the speaker both at training time and at test time, where it works together with the learned instruction follower model to solve the navigation task (see Figure 3.2 for an

---

[1]This chapter is adapted from *Speaker-Follower Models for Vision and Language Navigation*, (Fried et al., 2018b). Our code and data are available at `http://ronghanghu.com/speaker_follower`.

Figure 3.1: The task of vision-and-language navigation is to perform a sequence of actions (navigate through the environment) according to human natural language instructions. Our approach consists of an instruction *follower* model (left) and a *speaker* model (right).

overview of our approach). At training time, we perform speaker-driven data augmentation where the speaker helps the follower by synthesizing additional route-instruction pairs to expand the limited training data. At test time, the follower improves its chances of success by looking ahead at possible future routes and pragmatically choosing the best route by scoring them according to the probability that the speaker would generate the correct instruction for each route. This procedure, using the external speaker model, improves upon planning using only the follower model. We construct both the speaker and the follower on top of a panoramic action space that efficiently encodes high-level behavior, moving directly between adjacent locations rather than making low-level visuomotor decisions like the number of degrees to rotate (see Figure 3.3).

To summarize our contributions: We propose a novel approach to vision-and-language navigation incorporating a visually grounded speaker–follower model, and introduce a panoramic representation to efficiently represent high-level actions. We evaluate this speaker–follower model on the Room-to-Room (R2R) dataset (Anderson et al., 2018), and show that each component in our model improves performance at the instruction following task. Our model obtains a final success rate of 53.5% on the unseen test environment, an absolute improvement of 30% over existing approaches.

3.1 Related Work

Natural Language Instruction Following. Systems that learn to carry out natural language instructions in an interactive environment include approaches based on intermediate structured and executable representations of language (Tellex et al., 2011; Chen, 2012; Artzi and Zettlemoyer, 2013; Long et al., 2016; Guu et al., 2017) and approaches that map directly from language and world state observations to actions (Branavan et al., 2009; Andreas and Klein, 2015; Mei et al., 2016; Misra et al., 2017). The embodied vision-and-language navigation task studied in this chapter (Anderson et al., 2018) differs from past situated instruction following tasks by introducing rich visual contexts. Past work (Wang et al., 2018) has applied

techniques from model-based and model-free reinforcement learning (Weber et al., 2017) to the vision-and-language navigation problem. Specifically, an environment model is used to predict a representation of the state resulting from an action, and planning is performed with respect to this environment model. Our work differs from this prior work by reasoning not just about state transitions, but also about the relationship between states and the language that describes them—specifically, using an external speaker model to predict how well a given sequence of states explains an instruction.

Pragmatic Language Understanding.   A long line of work in linguistics, natural language processing, and cognitive science has studied *pragmatics*: how linguistic meaning is affected by context and communicative goals (Grice, 1975). Our work here makes use of the Rational Speech Acts framework (Frank and Goodman, 2012; Goodman and Stuhlmüller, 2013), which models the interaction between speakers and listeners as a process where each agent reasons probabilistically about the other to maximize the chances of successful communicative outcomes. This framework has been applied to model human use of language (Frank et al., 2009), and to improve the performance of systems that generate (Andreas and Klein, 2016; Mao et al., 2016; Vedantam et al., 2017; Cohn-Gordon et al., 2018) and interpret (Yu et al., 2017b; Luo and Shakhnarovich, 2017; Vasudevan et al., 2018) referential language. In the previous chapter, we applied similar modeling tools to generation and interpretation of language about sequential decision-making. The present work makes use of a pragmatic instruction follower in the same spirit. Here, however, we integrate this with a more complex visual pipeline and use it not only at inference time but also at *training* time to improve the quality of a base listener model.

Semi- and Self-Supervision.   The semi-supervised approach we use is related to model bootstrapping techniques such as self-training (Scudder, 1965; McClosky et al., 2006) and co-training (Blum and Mitchell, 1998) at a high-level. Past work has used monolingual corpora to improve the performance of neural machine translation models structurally similar to the sequence-to-sequence models we use (Gulcehre et al., 2015; He et al., 2016; Sennrich et al., 2016). In a grounded navigation context, Hermann et al. (2017) use a word-prediction task as training time supervision for a reinforcement learning agent. The approach most relevant to our work is the SEQ4 model (Kočiský et al., 2016), which applies semi-supervision to a navigation task by sampling new environments and maps (in synthetic domains without vision), and training an autoencoder to reconstruct routes, using language as a latent variable. The approach used here is much simpler, as it does not require constructing a differentiable surrogate to the decoding objective.

Semi-supervised data augmentation has also been widely used in computer vision tasks. In Data Distillation (Radosavovic et al., 2017), additional annotation for object and keypoint detection is obtained by ensembling and refining a pretrained model's prediction on unannotated images. Self-play in adversarial groups of agents is common in multi-agent reinforcement learning (Silver et al., 2017; Sukhbaatar et al., 2017). In actor-critic approaches

(Sutton and Barto, 1998; Sutton et al., 2000) in reinforcement learning, a critic learns the value of a state and is used to provide supervision to the actor's policy during training. In this work, we use a speaker to synthesize additional navigation instructions on unlabeled new routes, and use this synthetic data from the speaker to train the follower.

Grounding Language in Vision. Existing work in visual grounding (Plummer et al., 2015; Mao et al., 2016; Hu et al., 2016b; Rohrbach et al., 2016; Nagaraja et al., 2016) has addressed the problem of *passively* perceiving a static image and mapping a referential expression to a bounding box (Plummer et al., 2015; Mao et al., 2016; Hu et al., 2016b) or a segmentation mask (Hu et al., 2016a; Liu et al., 2017; Yu et al., 2018), exploring various techniques including proposal generation (Chen et al., 2017) and relationship handling (Wang et al., 2016a; Nagaraja et al., 2016; Hu et al., 2017; Cirik et al., 2018). In our work, the vision-and-language navigation task requires the agent to *actively* interact with the environment to find a path to the goal following the natural language instruction. This can be seen as a grounding problem in linguistics where the language instruction is grounded into a trajectory in the environment but requires more reasoning and planning skills than referential expression grounding.

3.2 Instruction Following with Speaker-Follower Models

To address the task of following natural language instructions, we rely on two models: an instruction-*follower* model of the kind considered in previous work, and a *speaker* model—a learned instruction generator that models how humans describe routes in navigation tasks.

Specifically, we base our follower model on the sequence-to-sequence model (Anderson et al., 2018), computing a distribution $P_F(r \mid d)$ over routes r (state and action sequences) given route descriptions d. The follower encodes the sequence of words in the route description with an LSTM (Hochreiter and Schmidhuber, 1997), and outputs route actions sequentially, using an attention mechanism (Bahdanau et al., 2015) over the description. Our speaker model is symmetric, producing a distribution $P_S(d \mid r)$ by encoding the sequence of visual observations and actions in the route using an LSTM, and then outputting an instruction word-by-word with an LSTM decoder using attention over the encoded input route (Figure 3.1).

We combine these two base models into a speaker-follower system, where the speaker supports the follower both at training time and at test time. An overview of our approach is presented in Figure 3.2. First, we train a speaker model on the available ground-truth navigation routes and instructions, shown in Figure 3.2a. Before training the follower, the speaker produces synthetic navigation instructions for novel sampled routes in the training environments, which are then used as additional supervision for the follower, as described in Section 3.2 and shown in Figure 3.2b. At follower test time, the follower generates pos-
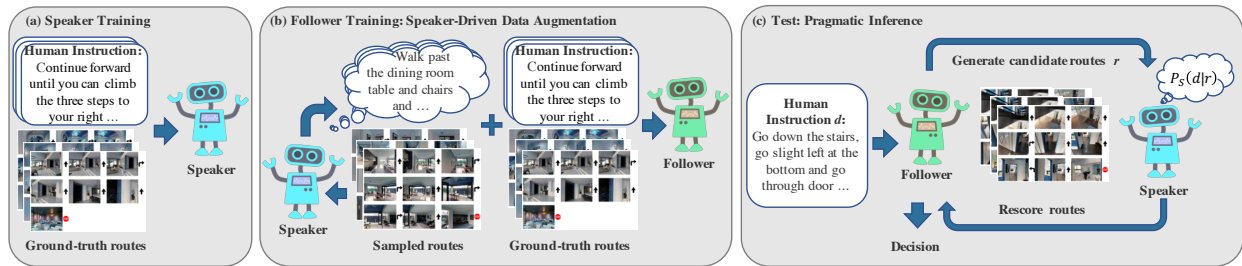
Figure 3.2: Our approach combines an instruction *follower* model and a *speaker* model. (a) The speaker model is trained on the ground-truth routes with human-generated descriptions; (b) it provides the follower with additional synthetic instruction data to bootstrap training; (c) it also helps the follower interpret ambiguous instructions and choose the best route during inference. See Section 3.2 for details.

sible routes as interpretations of a given instruction and starting context, and the speaker pragmatically ranks these, choosing one that provides a good explanation of the instruction in context (Section 3.2 and Figure 3.2c). Both follower and speaker are supported by the panoramic action space in Section 3.2 that reflects the high-level granularity of the navigational instructions (Figure 3.3).

Speaker-Driven Data Augmentation

The training data only covers a limited number of navigation instruction and route pairs, $\mathcal{D} = (d_1, r_1) \ldots (d_N, r_N)$. To allow the agent to generalize better to new routes, we use the speaker to generate synthetic instructions on sampled new routes in the training environments. To create a synthetic training set, we sample a collection of M routes $\hat{r}_1, \ldots, \hat{r}_M$ through the training environments, using the same shortest-path approach used to generate the routes in the original training set (Anderson et al., 2018). We then generate a human-like textual instruction $\hat{d}_k$ for each instruction $\hat{r}_k$ by performing greedy prediction in the speaker model to approximate $\hat{d}_k = \arg\max_d P_S(d \mid \hat{r}_k)$.

These M synthetic navigation routes and instructions $\mathcal{S} = (\hat{d}_1, \hat{r}_1), \ldots, (\hat{s}_M, \hat{r}_M)$ are combined with the original training data $\mathcal{D}$ into an augmented training set $\mathcal{S} \cup \mathcal{D}$ (Figure 3.2 (b)). During training, the follower is first trained on this augmented training set, and then further fine-tuned on the original training set $\mathcal{D}$. This speaker-driven data augmentation aims to overcome data scarcity issue, allowing the follower to learn how to navigate on new routes following the synthetic instructions.

Speaker-Driven Route Selection

We use the base speaker (P_S) and follower (P_F) models described above to define a *pragmatic follower* model. Drawing on the Rational Speech Acts framework (Frank and Goodman,

2012; Goodman and Stuhlmüller, 2013), a pragmatic follower model should choose a route r through the environment that has high probability of having caused the speaker model to produce the given description d: $\arg\max_r P_S(d \mid r)$ (corresponding to a rational Bayesian follower with a uniform prior over routes). Such a follower chooses a route that provides a good explanation of the observed description, allowing counterfactual reasoning about instructions, or using global context to correct errors in the follower's path, which we call *pragmatic inference.*

Given the sequence-to-sequence models that we use, exactly solving the maximization problem above is infeasible; and may not even be desirable, as these models are trained discriminatively and may be poorly calibrated for inputs dissimilar to those seen during training. Following previous work on pragmatic language generation and interpretation (Smith et al., 2013; Andreas and Klein, 2016; Monroe et al., 2017; Fried et al., 2018a), we use a rescoring procedure: produce candidate route interpretations for a given instruction using the base follower model, and then rescore these routes using the base speaker model (Figure 3.2c).

Our pragmatic follower produces a route for a given instruction by obtaining K candidate paths from the base follower using a search procedure described below, then chooses the highest scoring path under a combination of the follower and speaker model probabilities:

$$\arg\max_{r \in R(d)} P_S(d \mid r)^\lambda \cdot P_F(r \mid d)^{(1-\lambda)} \tag{3.1}$$

where λ is a hyper-parameter in the range $[0, 1]$ which we tune on validation data to maximize the accuracy of the follower.[2]

Candidate Route Generation. To generate candidate routes from the base follower model, we perform a search procedure where candidate routes are produced incrementally, action-by-action, and scored using the probabilities given by P_F. Standard beam search in sequence-to-sequence models (e.g., Sutskever et al., 2014) forces partial routes to compete based on the number of actions taken. We obtain better performance by instead using a *state-factored* search procedure, where partial output sequences compete at the level of states in the environment, where each state consists of the agent's location and discretized heading, keeping only the highest-scoring path found so far to each state. At a high-level, this search procedure resembles graph search with a closed list, but since action probabilities are non-stationary (potentially depend on the entire sequence of actions taken in the route), it is only approximate, and so we allow re-expanding states if a higher-scoring route to that state is found.

At each point in our state-factored search for searching and generating candidates in the follower model, we store the highest-probability route (as scored by the follower model) found

---

[2]In practice, we found best performance with values of λ close to 1, relying mostly on the score of the speaker to select routes. Using only the speaker score (which corresponds to the standard RSA pragmatic follower) did not substantially reduce performance compared to using a combination with the follower score, and both improved substantially upon using only the follower score (corresponding to the base follower).

so far to each state. States contain the follower's discrete location and heading (direction it is facing) in the environment, and whether the route has been completed (had the STOP action predicted). The highest-scoring route, which has not yet been expanded (had successors produced), is selected and expanded using each possible action from the state, producing routes to the neighboring states. For each of these routes r with final state s, if s has not yet been reached by the search, or if r is higher-scoring under the model than the current best path to s, r is stored as the best route to s. We continue the search procedure until K routes ending in distinct states have predicted the STOP action, or there are no remaining unexpanded routes. See Algorithm 1 for pseudocode.

Algorithm 1: State-factored search

---

class STATE**:**
    location   *# the agent's physical position in the environment*
    completed   *# whether the route has been completed (the* STOP *action has been taken)*
class ROUTE**:**
    states   *# list of* STATE*s in the route*
    score   *# route probability under the follower model,* P_F
    expanded   *# whether this route has been expanded in search*
function StateFactoredSearch(*start_state* : STATE*, K* : *int*)**:**
    partial = {}   *# a mapping from* STATE*s to the best* ROUTE *ending in that state found so far*
      (whether expanded or unexpanded)
    completed = {}   *# a similar mapping, but containing routes that have been completed*
    start_route = ROUTE([*start_state*]*, *1.0*, False*)
    partial[*start_state*] = *start_route*
    candidates = [*start_route*]
    while |*completed*| < *K* **and** |*candidates*| > 0 **:**
        # choose the highest-scoring unexpanded route to expand (route may be complete)
        route = $argmax_{r \in candidates}$*r.score*
        route.expanded = *True*
        # SUCCESSORS *generates* ROUTE*s by taking all possible actions, each of which extends this*
          route by one state, with the resulting total model score, and expanded set to False
        for *route′* **in** SUCCESSORS(*route*) **:**
            state′ = *route′.states.last*
            cache = *completed* **if** *route′.completed* **else** *partial*
            if *state′* **not in** *cache.keys* **or** *cache*[*state′*]*.score* < *route′.score* **:**
                cache[*state′*] = *route′*
        candidates = [*route* **in** *partial.values* **if not** *route.expanded*]
    return *completed*

Since route scores are products of conditional probabilities, route scores are non-increasing, and so this search procedure generates routes that do not pass through the same state twice—which we found to improve accuracy both for the base follower model and the pragmatic rescoring procedure, since instructions typically describe acyclic routes.

We generate up to $K = 40$ candidate routes for each instruction using this procedure, and rescore using Equation 3.1. In addition to enabling pragmatic inference, this state-factored search procedure improves the performance of the follower model on its own (taking the
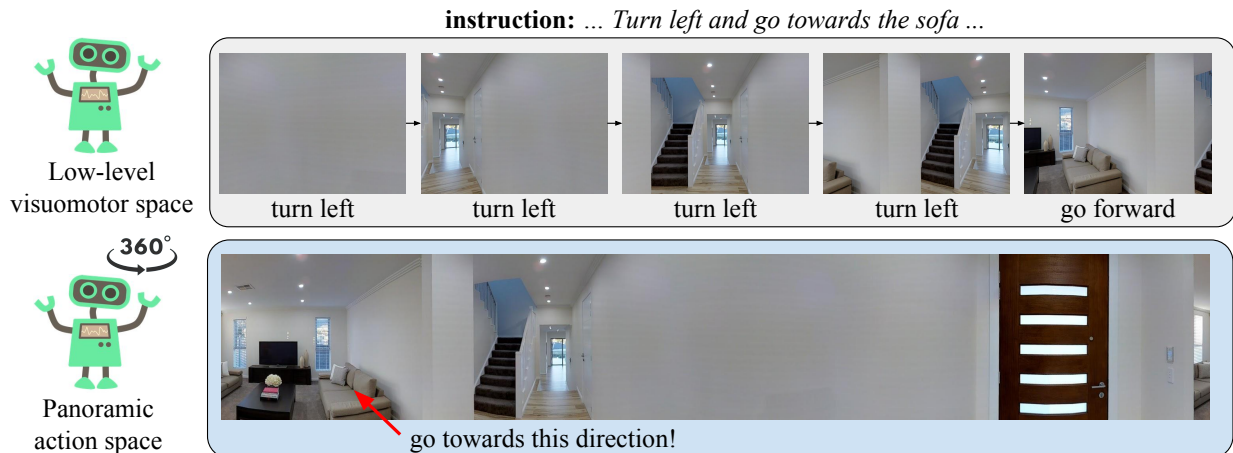
Figure 3.3: Compared with low-level visuomotor space, our panoramic action space (Sec. 3.2) allows the agents to have a complete perception of the scene, and to directly perform high-level actions.

candidate route with highest score under the follower model), when compared to standard greedy search (see Section 3.5 and Figure 3.5 for details).

Panoramic Action Space

The sequence-to-sequence agent in Anderson et al. (2018) uses low-level visuomotor control (such as turning left or right by 30 degrees), and only perceives frontal visual sensory input. Such fine-grained visuomotor control and restricted visual signal introduce challenges for instruction following. For example in Figure 3.3, to "turn left and go towards the sofa", the agent needs to perform a series of turning actions until it sees a sofa in the center of its view, and then perform a "go forward" action. This requires strong skills of planning and memorization of visual inputs. While a possible way to address this challenge is to learn a hierarchical policy such as in Das et al. (2018), in our work we directly allow the agent to reason about high-level actions, using a panoramic action space with panoramic representation, converted with built-in mapping from low-level visuomotor control.

As shown in Figure 3.3, in our panoramic representation, the agent first "looks around" and perceives a 360-degree panoramic view of its surrounding scene from its current location, which is discretized into 36 view angles (12 headings × 3 elevations with 30 degree intervals – in our implementation). Each view angle i is represented by an encoding vector v_i. At each location, the agent can only move towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to make high-level decisions as to which navigable direction to go to next, with each navigable direction j represented by an encoding

vector u_j. The encoding vectors v_i and u_j of each view angle and navigable direction are obtained by concatenating an appearance feature (ConvNet feature extracted from the local image patch around that view angle or direction) and a 4-dimensional orientation feature $[\sin\psi; \cos\psi; \sin\theta; \cos\theta]$, where ψ and θ are the heading and elevation angles respectively. Also, we introduce a STOP action encoded by $u_0 = \overrightarrow{0}$. The agent can take this STOP action when it decides it has reached the goal location (to end the episode).

To make a decision on which direction to go, the agent first performs one-hop visual attention to look at all of the surrounding view angles, based on its memory vector h_{t-1}. The attention weight $\alpha_{t,i}$ of each view angle i is computed as $a_{t,i} = (W_1 h_{t-1})^T W_2 v_{t,i}$ and $\alpha_{t,i} = \exp(a_{t,i})/\sum_i \exp(a_{t,i})$.

The attended feature representation $v_{t,att} = \sum_i \alpha_{t,i} v_{t,i}$ from the panoramic scene is then used as visual-sensory input to the sequence-to-sequence model (replacing the 60-degree frontal appearance vector in Anderson et al. (2018)) to update the agent's memory. Then, a bilinear dot product is used to obtain the probability p_j of each navigable direction j: $y_j = (W_3 h_t)^T W_4 u_j$ and $p_j = \exp(y_j)/\sum_j \exp(y_j)$.

The agent then chooses a navigable direction u_j (with probability p_j) to go to the adjacent location along that direction (or u_0 to stop and end the episode). We use a built-in mapping that seamlessly translates our panoramic perception and action into visuomotor control such as turning and moving.

3.3 Experiments

Experimental Setup

Dataset. We use the Room-to-Room (R2R) vision-and-language navigation dataset (Anderson et al., 2018) for our experimental evaluation. In this task, the agent starts at a certain location in an environment and is provided with a human-generated navigation instruction, that describes a path to a goal location. The agent needs to follow the instruction by taking multiple discrete actions (e.g. turning, moving) to navigate to the goal location, and executing a "stop" action to end the episode. Note that differently from some robotic navigation settings (Pathak et al., 2018), here the agent is not provided with a goal image, but must identify from the textual description and environment whether it has reached the goal.

The dataset consists of 7,189 paths sampled from the Matterport3D (Chang et al., 2017) navigation graphs, where each path consists of 5 to 7 discrete viewpoints and the average physical path length is 10m. Each path has three instructions written by humans, giving 21.5k instructions in total, with an average of 29 words per instruction. The dataset is split into training, validation, and test sets. The validation set is split into two parts: *seen*, where routes are sampled from environments seen during training, and *unseen* with environments that are not seen during training. All the test set routes belong to new environments unseen in the training and validation sets.

Evaluation Metrics. Following previous work on the R2R task, our primary evaluation metrics are navigation error (NE), measuring the average distance between the end-location predicted by the follower agent and the true route's end-location, and success rate (SR), the percentage of predicted end-locations within 3m of the true location. As in previous work, we also report the oracle success rate (OSR), measuring success rate at the closest point to the goal that the follower has visited along the route, allowing the agent to overshoot the goal without being penalized.

Implementation Details. Following Anderson et al. (2018) and Wang et al. (2018), we produce visual feature vectors v using the output from the final convolutional layer of a ResNet (He et al., 2016) trained on the ImageNet (Russakovsky et al., 2015) classification dataset. These visual features are fixed, and the ResNet is not updated during training. To better generalize to novel words in the vocabulary, we also experiment with using GloVe embeddings (Pennington et al., 2014), to initialize the word-embedding vectors in the speaker and follower.

In the baseline without using synthetic instructions, we train follower and speaker models using the human-generated instructions for routes present in the training set. The training procedure for the follower model follows (Anderson et al., 2018) by training with *student-forcing* (sampling actions from the model during training, and supervising using a shortest-path action to reach the goal state). We use the training split in the R2R dataset to train our speaker model, using standard maximum likelihood training with a cross-entropy loss.

In speaker-driven data augmentation (Section 3.2), we augment the data used to train the follower model by sampling $178,000$ routes from the training environments. Instructions for these routes are generated using greedy inference in the speaker model (which is trained only on human-produced instructions). The follower model is trained using student-forcing on this augmented data for $50,000$ iterations, and then fine-tuned on the the original human-produced data for $20,000$ iterations. For all experiments using pragmatic inference, we use a speaker weight of $\lambda = 0.95$, which we found to produce the best results on both the seen and unseen validation environments.

3.4 Results and Analysis

We first examine the contribution from each of our model's components on the validation splits. Then, we compare the performance of our model with previous work on the unseen test split.

Component Contributions

We begin with a baseline (Row 1 of Table 3.1), which uses only a follower model with a non-panoramic action space at both training and test time, which is equivalent to the student-forcing model in Anderson et al. (2018).[3]

---

[3]Note that our results for this baseline are slightly higher on val-seen and slightly lower on val-unseen than those reported by Anderson et al. (2018), due to differences in implementation details and hyper-parameter

#	Data Augmentation	Pragmatic Inference	Panoramic Space	Validation-Seen			Validation-Unseen		
				NE ↓	SR ↑	OSR ↑	NE ↓	SR ↑	OSR ↑
1				6.08	40.3	51.6	7.90	19.9	26.1
2	✓			5.05	46.8	59.9	7.30	24.6	33.2
3		✓		5.23	51.5	60.8	6.62	34.5	43.1
4			✓	4.86	52.1	63.3	7.07	31.2	41.3
5	✓	✓		4.28	57.2	63.9	5.75	39.3	47.0
6	✓		✓	3.36	66.4	73.8	6.62	35.5	45.0
7		✓	✓	3.88	63.3	71.0	5.24	49.5	63.4
8	✓	✓	✓	**3.08**	**70.1**	**78.3**	**4.83**	**54.6**	**65.2**

Table 3.1: Ablations showing the effect of each component in our model. Rows 2-4 show the effects of adding a single component to the baseline system (Row 1); Rows 5-7 show the effects of removing a single component from the full system (Row 8). NE is navigation error (in meters); lower is better. SR and OSR are success rate and oracle success rate (%); higher is better. See Section 3.4 for details.

Speaker-Driven Data Augmentation. We first introduce the speaker at training time for data augmentation (Section 3.2). Comparing Row 1 (the baseline follower model trained only with the original training data) against Row 2 (training this model on augmented data) in Table 3.1, we see that adding the augmented data improves success rate (SR) from 40.3% to 46.8% on validation seen and from 19.9% to 24.6% on validation unseen, respectively. This higher relative gain on unseen environments shows that the follower can learn from the speaker-annotated routes to better generalize to new scenes.

Note that given the noise in our augmented data, we fine-tune our model on *the original training data* at the end of training as mentioned in Section 3.2. We find that increasing the amount of augmented data is helpful in general. For example, when using 25% of the augmented data, the success rate improves to 22.8% on validation unseen, while with all the augmented data the success rate reaches 24.6% on validation unseen, which is a good balance between performance and computation overhead.

Pragmatic Inference. We then incorporate the speaker at test time for pragmatic inference (Section 3.2), using the speaker to rescore the route candidates produced by the follower. Adding this technique brings a further improvement in success rate on both environments (compare Row 2, the data-augmented follower without pragmatic inference, to Row 5, adding pragmatic inference). This shows that when reasoning about navigational

choices.

directions, large improvements in accuracy can be obtained by scoring how well the route explains the direction using a speaker model. Importantly, when using only the follower model to score candidates produced in search, the success rate is 49.0% on val-seen and 30.5% on val-unseen, showing the importance of using the speaker model to choose among candidates (which increases success rates to 57.2% and 39.3%, respectively).

Panoramic Action Space. Finally, we replace the visuomotor control space with the panoramic representation (Section 3.2). Adding this to the previous system (compare Row 5 and Row 8) shows that the new representation leads to a substantially higher success rate, achieving 70.1% and 54.6% success rate on validation seen and validation unseen, respectively. This suggests that directly acting in a higher-level representation space makes it easier to accurately carry out instructions. Our final model (Row 8) has over twice the success rate of the baseline follower in the unseen environments.

Importance of All Components. Above we have shown the gain from each component, after being added incrementally. Moreover, comparing Rows 2-4 (adding each component independently to the base model) to the baseline (Row 1) shows that each component in isolation provides large improvements in success rates, and decreases the navigation error. Ablating each component (Rows 5-7) from the full model (Row 8) shows that each of them is important for the final performance.

Qualitative Results. The intuition behind the speaker model is that it should help the agent more accurately interpret instructions specifically in ambiguous situations. Figure 3.6 (at the end of this chapter) shows how the introduction of a speaker model helps the follower in pragmatic inference.

Comparison to Prior Work

We compare the performance of our final model to previous approaches on the R2R held-out splits, including the test split which contains 18 new environments that do not overlap with any training or validation splits, and are only seen once at test time.

The results are shown in Table 3.2. In the table, "Random" is randomly picking a direction and going towards that direction for 5 steps. "Student-forcing" is the best performing method in Anderson et al. (2018), using exploration during training of the sequence-to-sequence follower model. "RPA" (Wang et al., 2018) is a combination of model-based and model-free reinforcement learning (see also Section 3.1 for details). "ours" shows our performance using the route selected by our pragmatic inference procedure, while "ours (VLNC)" uses a modified inference procedure for submission to the Vision-and-Language Navigation Challenge (see Appendix A). Prior work has reported higher performance on the seen rather than unseen environments (Anderson et al., 2018; Wang et al., 2018), illustrating the issue of generalizing to new environments. Our method more than doubles the success rate of the

Method	Validation-Seen			Validation-Unseen			Test (unseen)			
	NE ↓	SR ↑	OSR ↑	NE ↓	SR ↑	OSR ↑	NE ↓	SR ↑	OSR ↑	TL ↓
Random	9.45	15.9	21.4	9.23	16.3	22.0	9.77	13.2	18.3	9.89
Student-forcing	6.01	38.6	52.9	7.81	21.8	28.4	7.85	20.4	26.6	8.13
RPA	5.56	42.9	52.6	7.65	24.6	31.8	7.53	25.3	32.5	9.15
ours	**3.08**	**70.1**	**78.3**	**4.83**	**54.6**	**65.2**	**4.87**	**53.5**	63.9	11.63
ours (VLNC)*	–	–	–	–	–	–	**4.87**	**53.5**	**96.0**	1257.38
Human	–	–	–	–	–	–	1.61	86.4	90.2	11.90

Table 3.2: Performance comparison of our method to previous work: the Student-forcing method of Anderson et al. (2018) and the RPA system of Wang et al. (2018). NE is navigation error (in meters); lower is better. SR and OSR are success rate and oracle success rate (%) respectively (higher is better). Trajectory length (TL) on the test set is reported for completeness. *: *When submitting to the Vision-and-Language Navigation Challenge, we modified our search procedure to maintain physical plausibility and to comply with the challenge guidelines. The resulting trajectory has higher oracle success rate while being very long. See Appendix A for details.*

state-of-the-art RPA approach, and on the test set achieves a final success rate of 53.5%. This represents a large reduction in the gap between machine and human performance on this task.

3.5   Analysis of Inference Parameters and Implementation Details

We further study how varying the speaker weight λ in pragmatic inference (Section 3.2) influences the final navigation performance. In Table 3.3, we compare our full model (with speaker weight $\lambda = 0.95$) in Row 1 against using only the follower model to score routes ($\lambda = 0$) in Row 2, a baseline that still includes search but does not include speaker scores. The large gap in success rate between $\lambda = 0.95$ and $\lambda = 0$ shows that including speaker scores is crucial to the performance, confirming the importance of pragmatic inference. Figure 3.4 shows the average number of actions and the navigation error on val unseen with different speaker weights λ, where $\lambda = 0.95$ gives the lowest navigation error.

We also analyze some implementation details in our model. Comparing row 1 versus row 3 in Table 3.3 shows that using state-factored search to produce candidates for pragmatic inference produces better results on val unseen than using (standard) beam search. Comparing row 1 versus row 4 in Table 3.3 indicates that using GloVe (Pennington et al., 2014) to initialize word embedding sightly improves success rate.

#		Validation-Seen			Validation-Unseen		
		NE $\downarrow$	SR $\uparrow$	OSR $\uparrow$	NE $\downarrow$	SR $\uparrow$	OSR $\uparrow$
1	our full model ($\lambda = 0.95$)	**3.08**	70.1	**78.3**	**4.83**	**54.6**	65.2
2	w/o speaker scoring ($\lambda = 0$)	3.17	68.4	74.8	5.94	43.7	53.1
3	w/o state-factoring in search	3.14	**70.6**	77.4	5.27	50.7	60.7
4	w/o GloVe embeddings	3.08	69.6	77.4	4.84	53.2	**66.7**

Table 3.3: Effects of speaker scoring and implementation details in our model. NE is navigation error (in meters); lower is better. SR and OSR are success rate and oracle success rate (%); higher is better. Comparison between the 1st and the 2nd row shows that incorporating speaker scoring is crucial to performance, showing the importance of pragmatic inference. Comparison between the 1st and the 3rd row indicates that state-factored search (Section 3.2) produces better results than standard beam search on val-unseen. The difference between the 1st and the 4th row shows that using GloVe embeddings (Pennington et al., 2014) gives slightly higher success rate.
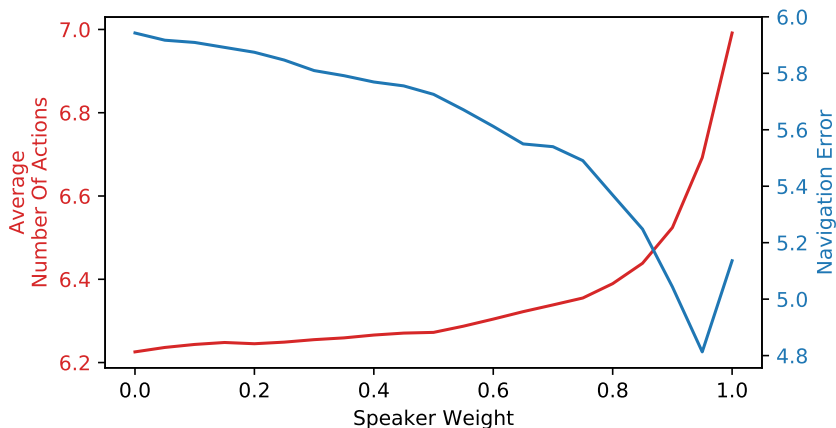
Figure 3.4: The average number of actions and navigation error with different speaker weights λ in pragmatic inference, evaluated on the val unseen split. Larger λ results in more number of actions on average, while $\lambda = 0.95$ gives the lowest navigation error.

In addition, we study how the number of candidate routes, K, used in pragmatic inference impacts the final navigation success rate. Figure 3.5 shows the success rate of our model on R2R val seen and val unseen splits using different numbers of candidate routes K for state-factored search. The results show that having more routes to choose between leads to better generalization to the unseen new environments (val unseen), but the gain from increasing the number of candidates tends to saturate quickly. In our final model in Table 3.2 and Table 3.3, we use $K = 40$. However, we emphasize that even with only five route candidates,
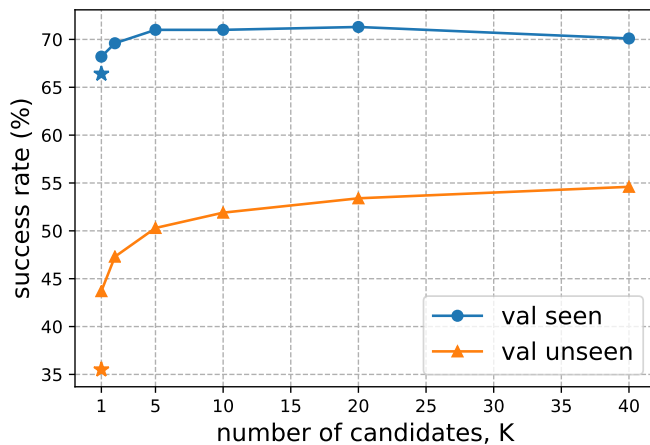
Figure 3.5: The success rate of our model on the val seen and val unseen splits, using different numbers K of route candidates (generated by state-factored search) for pragmatic inference. Stars show the performance of greedy inference (without search, and hence without pragmatics). While performance increases with number of candidates up through 40 on val unseen, the success rate tends to saturate. We note improvements both from the state-factored search procedure (comparing the stars to the circle and triangle points at $K = 1$) as well as from having more candidates to choose from in pragmatic inference (comparing larger values of K to smaller).
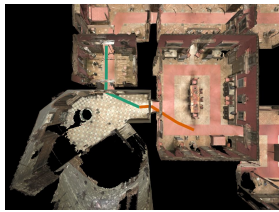
our model still achieves 50.3% success rate on val unseen, which improves substantially on both the 35.5% success rate from greedy decoding (i.e. the gold star at $K = 1$ in Figure 3.5), as well as the 43.5% success rate given by state-factored search with no pragmatic inference (i.e. the gold triangle at $K = 1$ in Figure 3.5).

3.6   Discussion

The language-and-vision navigation task presents a pair of challenging reasoning problems: in language, because agents must interpret instructions in a changing environmental context; and in vision, because of the tight coupling between local perception and long-term decision-making. The comparatively poor performance of the baseline sequence-to-sequence model for instruction following suggests that more powerful modeling tools are needed to meet these challenges. In this work, we have introduced such a tool, showing that a follower model for vision-and-language navigation is substantially improved by carefully structuring the action space and integrating an explicit model of a *speaker* that predicts how navigation routes are described. We believe that these results point toward further opportunities for improvements in instruction following by modeling the global structure of navigation behaviors and the pragmatic contexts in which they occur.

instruction:
Go through the door on the right and continue straight. Stop in the next room in front of the bed.

top-down overview of trajectories

(a) **orange**: trajectory without pragmatic inference

(b) **green**: trajectory with pragmatic inference

Step 1

Step 2

Step 3

Step 4

(a) navigation steps **without pragmatic inference**; red arrow: direction to go next

Step 1

Step 2

Step 3

Step 4

(b) navigation steps **with pragmatic inference**; red arrow: direction to go next

Figure 3.6: Navigation examples on unseen environments with and without pragmatic inference from the speaker model (*best visualized in color*). (a) The follower without pragmatic inference misinterpreted the instruction and went through a wrong door into a room with no bed. It then stopped at a table (which resembles a bed). (b) With the help of a speaker for pragmatic inference, the follower selected the correct route that enters the right door and stopped at the bed.

Chapter 4

Grounded Collaborative Dialogue

In grounded dialogue settings involving high degrees of ambiguity, correctly interpreting and informatively generating language can prove challenging. Consider the collaborative dialogue game shown in Figure 4.1. Each player has a separate, but overlapping, view on an underlying context. They need to communicate to determine and agree on one dot that they share, and both players win if they choose the same dot. To succeed, each participant must— implicitly or explicitly—ground their partner's descriptions to their own context, maintain a history of what's been described and what their partner is likely to have, and informatively convey parts of their own context.

In this chapter, we present a grounded pragmatic dialogue system which collaborates successfully with people on the task above. Figure 4.1 shows a real example game between our system and a human partner. Our approach is centered around a structured module for perceptually-grounded reference resolution. This reference resolution module plays two roles. First, the module is used to *interpret* the partner's utterances: explicitly predicting which referents (if any) in the agent's context the partner is referring to, for example *a smaller darker grey dot* and *the lighest grey and larger dot*. Second, the reference module is used for *pragmatic generation*: choosing utterances by reasoning about how the partner might interpret them in context. Our pragmatic generation procedure selects referents to describe as well as choosing how to describe them, for example focusing on *the light one* (Figure 4.1).

Most past work that has constructed systems for grounded collaborative dialogue has focused on settings that have asymmetric player roles (Kim et al., 2019; de Vries et al., 2018; Das et al., 2018, 2017), are fully-observable, or are grounded in symbolic attributes (He et al., 2017). In contrast, we focus on the ONECOMMON corpus and task (Udagawa and Aizawa, 2019), which is symmetric, partially-observable, and has relatively complex spatial and perceptual grounding. These traits necessitate complex dialogue strategies such as common grounding, coordination, clarification questions, and nuanced acknowledgment (Udagawa and Aizawa, 2019), leading to the task being challenging even for pairs of human partners.

Past work on ONECOMMON has focused on the subtask of reference resolution (Udagawa and Aizawa, 2020; Udagawa et al., 2020), and evaluated dialog systems only in automatic

Underlying Hidden State

Agent's View   Partner's View

A: I have two large light grey dots with a smaller darker dot
   below and to the right of it
P: I have one large lightest grey dot in my entire view
A: I have two large light grey dots
P: There is a smaller slightly darker grey dot next to the
   lightest grey and larger dot
A: Yes , let's pick the light one
P: The light grey and large dot?
A: Yes it 's the one . Let's pick that one
P: SELECT red
A: SELECT blue

Figure 4.1: An example dialogue produced by our system (A) with a human partner (P). The participants have different but overlapping views of a shared board, which contains dots of different shapes and sizes. The partners must collaborate through dialogue in order to find and select a dot that is visible to both.

evaluation: using static evaluation on human–human games and self-play evaluations that simulate human partners using another copy of the agent. Our system outperforms this past work on these evaluations. We further confirm these results by performing—for the first time on this task—human evaluations, where we find that our system obtains a 50% relative increase in success rate over a system from past work when paired with human partners.

4.1   Setting

We choose to focus on the OneCommon task (Udagawa and Aizawa, 2019) since it is a particularly challenging representative of a class of partially-observable collaborative reference dialog games (e.g., He et al. 2017; Haber et al. 2019). In this task, two players have different but overlapping views of a game board, which consists of dots of various positions, shades of gray and sizes. The players must coordinate to choose a single dot that both players can see, which is challenging because neither knows which dots the other can see.

Each player's world view, w, consists of a circular view on an underlying board containing randomly scattered dots, with continuously varying positions, shades, and sizes (Figure 4.1). Each player's view contains 7 dots, and the views of the players overlap so that there are between 4 and 6 dots which are in both players' views.

We focus on a turn-based version of the dialogue task. In a given turn t, a player may communicate with their partner by either sending an utterance u_t or selecting a dot s. In the event of selection, the partner is notified but cannot see which dot the player has selected. Once a player has selected a dot, they can no longer send messages. The dialogue ends once both players have selected a dot, and is successful if both selected the same one.

4.2   Model Structure

Our approach is a modular neural dialogue model which factors the agent's generation process into a series of successive subtasks, all centered on grounding language into referents in the world context. In this section, we describe our model structure, which defines a neural module for each subtask. We then describe our reference-centric pragmatic generation procedure in Section 4.3.

An overview of the relationship between modules in our model is shown in Figure 4.2. Each module can condition on neural encodings of the context (the world and past dialogue), as well as the outputs of other modules. We describe our system at a high-level here, then give task-specific implementation details about each component in Section 4.4.

Context Encodings

Our modules can condition on encodings of (i) the past utterances $u_{1:t}$ in the dialogue, represented as a memory vector H_t produced by a word-level recurrent encoder and (ii) the continuous dots in the world context w, produced by the entity encoding network of Udagawa and Aizawa (2020), which produces a vector $w(d)$ for each dot d encoding the dot's continuous attributes as well as its pairwise attribute relationships to all other dots in the context (Santoro et al., 2017). (i) and (ii) both follow Udagawa and Aizawa (2020). To explicitly encourage the model to retain and use information about the history of referents mentioned by both players, which affects the choice of future referents as well as the selection of dot at the end of the game, we also use (iii) a structured recurrent **referent memory**
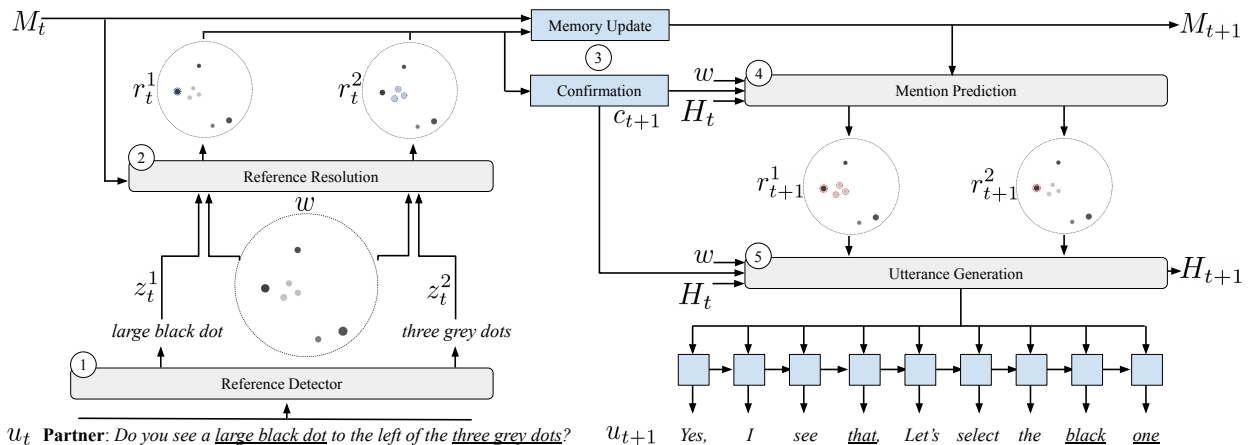
Figure 4.2: In a given turn, an agent first identifies referring expressions in their partner's utterance u_t using the reference detector (1). Each reference is then resolved with the reference resolution module (2), which uses encoded representations $z^{1:K_t}$ of the reference segments and the world context w. The referents are then used to update the referent memory M_t, and cross-referenced against the agent's own dots to confirm whether or not the agent can also see them (3). Given the referent memory M_t and confirmation variable c_{t+1}, the mention prediction module (4) produces a sequence of dot configurations $z_{t+1}^{1:K_{t+1}}$ to mention. Finally, the utterance generation module (5) uses the dialog history H_t, confirmation variable, and attended representations of the selected mentions and world context to generate a response u_{t+1}.

grounded in the context. This memory, inspired by He et al. (2017), has one representation for each dot d in the agent's view, $M_t(d)$, which is updated based on the referents predicted in turn t. See Section 4.4 for details.

Decomposing Turns into Subtasks

We assume turn $t+1$ in the dialogue has the following generative process (numbers correspond to Figure 4.2). Steps (1) and (2) identify and resolve referring expressions in the partner's last utterance u_t; steps (3) and (4) produce the agent's next utterance u_{t+1}.

1. First, a sequence of K_t (with $K_t \geq 0$) referring expressions are identified in u_t using the **reference detector** tagging model of Udagawa and Aizawa (2020)[1], and encodings $\mathbf{z}_t = z_t^{1:K_t}$ are obtained for them by pooling features from a recurrent utterance encoder.

2. Then, the referring expressions are grounded. From each referring expression's features z^k, we predict a referent r^k, which is the set of zero or more dots in the agent's own view

---

[1]Udagawa and Aizawa refer to this as a *markable detector* given their work's focus on referent annotation.

which are described by the referring expression. For example, the referring expression *three gray dots* corresponds to a single referent containing three dots. A **reference resolution** module $P_R(\mathbf{r}_t \mid \mathbf{z}_t, w, M)$, where $\mathbf{r}_t = r_t^{1:K_t}$, predicts a sequence of referents, one for each referring expression.

3. Given these referents, the agent updates the referent memory M_t using the predicted referents and constructs a discrete **confirmation variable** c_{t+1}, which indicates whether the agent can confirm in its next utterance that it has all the referents the partner is describing (e.g., *Yes, I see that*). c_{t+1} takes on one of three values: NA if no referring expressions were in the partner's utterance, YES if all of the partner's referring expressions have referents that are at least partially visible in the agent's view, and NO otherwise.

4. The agent chooses a sequence of referents to mention next using a **mention prediction** module $P_M(\mathbf{r}_{t+1} \mid c_{t+1}, M_{t+1}, H_t, w)$.

5. Finally, the next utterance u_{t+1} is produced using an **utterance generation** module $P_U(u_{t+1} \mid \mathbf{r}_{t+1}, c_{t+1}, H_t, w)$, also updating the word-level recurrent memory H_{t+1}.

At the end of the dialogue (turn T), the agent selects a dot s using a **choice selection** module $P_S(s \mid H_T, M_T, w)$ (not shown in Figure 4.2).[2]

Modules that predict referents (reference resolution, mention selection, and choice selection) are all implemented using a structured conditional random field (CRF) architecture (Section 4.4), with independent parameterizations for each module.

Our model bears some similarities to Udagawa and Aizawa (2020)'s neural dialogue model for this task: both models use a reference resolution module[3] and both models attend to similar encodings of the dots in the agent's world view ($w(d)$) when generating language. Crucially, however, our decomposition of generation into subtasks results in a factored, hierarchical generation procedure: our model identifies and then conditions on previously-mentioned referents from the partner's utterances,[4] maintains a structured referent memory updated at each utterance, and explicitly predicts which referents to mention in each of the agent's own utterances. In Section 4.3, we will see how factoring the generation procedure in this way allows us to use a pragmatic generation procedure, and in Section 4.5 we find that each of these components improves performance.

4.3 Pragmatic Generation

The modules as described above can be used to generate the next utterance u_{t+1} using the predictions of $P_M(\mathbf{r}_{t+1})$ and $P_U(u_{t+1}|\mathbf{r}_{t+1})$ (omitting other conditioning variables from the

---

[2]The choice selection module is invoked when the utterance generation model predicts a special <SELECT> token, following Udagawa and Aizawa (2020).

[3]Our model, however, uses a structured CRF while Udagawa and Aizawa's is unstructured.

[4]Udagawa and Aizawa used the reference resolution module only to define an auxiliary loss at training time.

notation for brevity; see Section 4.2 for the full conditioning contexts). This section describes an improvement, *pragmatic generation*, to this process. Utterances should be appropriate in the dialogue and world context (as modeled by P_M and P_U), but they should also be discriminative: allowing the listener to easily understand which referents the speaker is intending to describe. Our pragmatic generation approach, based on the Rational Speech Acts (RSA) framework (Frank and Goodman, 2012), uses the reference resolution module to predict whether the partner can identify the intended referents, $P_R(\mathbf{r}_{t+1}|u_{t+1})$. This encourages selecting referents that are easy for the partner to identify and describing them informatively in context.[5]

We use the following objective over referents $\mathbf{r}$ and utterances u for a given turn:

$$\underset{\mathbf{r},u}{\arg\max} \ L(\mathbf{r}, u)$$
$$L(\mathbf{r}, u) = P_M(\mathbf{r})^{w_M} \cdot P_U(u|\mathbf{r})^{w_S} \cdot P_R(\mathbf{r}|u)^{w_L}$$

where w_M, w_S, and w_L are hyperparameters.

This objective generalizes the typical RSA setup (as implemented by the weighted pragmatic inference objective of e.g., Andreas and Klein 2016 and Monroe et al. 2017), which chooses *how* to describe a given context (i.e., choosing an utterance u), to also choose *what* context to describe (i.e., choosing $\mathbf{r}$).

Given the combinatorially-large spaces of possible $\mathbf{r}$ and u, we rely on an early-stopping approximate search, which to our knowledge is novel for RSA: iterating through the highest probability structured referent sequences $\mathbf{r}$ under P_M, and for each $\mathbf{r}$ sampling N_u utterances u. If the maximum of these $(\mathbf{r}, u)$ pairs under L is better than a threshold τ, we return the pair. Otherwise, we continue on to the next $\mathbf{r}$. If more than N_r referent sequences have been evaluated, we return the best $(\mathbf{r}, u)$ pair found so far.

We give pseudocode for the pragmatic generation procedure in Algorithm 2. Figure 4.3 shows an example, showing 2 referents $\mathbf{r}$ (inputs to the `realize`) function on the left, and 3 utterances u sampled for each referent on the right. Fewer than N_r referent candidates may be evaluated (as in this case) if one $(\mathbf{r}, u)$ pair is found with $L(\mathbf{r}, u) \geq \tau$.

4.4 Module Implementations

As described so far, our system is applicable to a range of partially-observable grounded collaborative referring expression dialogue tasks (e.g., He et al. 2017; Haber et al. 2019). In this section, we describe implementations of our systems' modules, some of which are tailored to ONECOMMON.

---

[5]Note that the reference resolution model, which has access to the agent's own view and not the partner's, can only approximate whether the referents are identifiable by the partner; nevertheless we find that it is beneficial for pragmatic generation. Future work might explore also inferring and using the partner's view.

Algorithm 2: Our pragmatic generation procedure chooses a sequence of referents $\mathbf{r}$ to describe, and an utterance u to describe them, to optimize the objective $L(\mathbf{r}, u)$ (Sec. 4.3) using candidates from the models P_M and P_U and an early stopping search with threshold τ.

---

hyperparameters: N_r, N_u, τ
function generate($M, u_{1:t-1}, c_t, w$)**:**
    $(\hat{\mathbf{r}}, \hat{u}, \hat{s}) \leftarrow (\text{None}, \text{None}, -\infty)$
    for $\mathbf{r} \in \text{topk}_{N_r} P_M(\mathbf{r}|u_{1:t-1}, M, c, w)$ **:**
        $u, s \leftarrow$ realize($\mathbf{r}$)
        if $s > \hat{s}$ **:**
            $(\hat{\mathbf{r}}, \hat{u}, \hat{s}) \leftarrow (\mathbf{r}, u, s)$
            if $s \geq \tau$ **:**
                break
    return $\hat{\mathbf{r}}, \hat{u}$
function realize($\mathbf{r}$)**:**
    for $k \in 1 \ldots N_u$ **:**
        $u^{(k)} \sim P_U(\cdot \mid \mathbf{r})$
    $\hat{u} \leftarrow \arg\max_{u^{(k)}} L(\mathbf{r}, u^{(k)})$
    $\hat{s} \leftarrow \max_{u^{(k)}} L(\mathbf{r}, u^{(k)})$
    return $(\hat{u}, \hat{s})$

Reference Detection

We identify a sequence of referring expressions in the utterance using the **reference detector** of Udagawa and Aizawa (2020), a BiLSTM-CRF tagger (Huang et al., 2015). Then, following Udagawa and Aizawa (2020), we obtain features z^k for each of the K referring expressions in the utterance (for use in the reference resolution model) with a bidirectional recurrent encoder, using learned weights to pool the encodings at the referring expression's boundaries as well as the end of the utterance.[6]

Structured Reference Resolution

We use a structured reference resolution module to ground the referring expressions identified above: identifying dots in the agent's own view described by each expression. Grounding referents in this domain involves reasoning not only about attributes of individual dots but also spatially and comparatively within a single referring expression (e.g., *a line of three dots*) or across referring expressions (e.g., *a large grey dot* left of *a smaller dot*).

To predict a sequence of referents $\mathbf{r} = r^{1:K}$ from the K referring expression representations $z^{1:K}$ extracted above, we use a linear-chain CRF (Lafferty et al., 2001) with neural potentials to parameterize $P_R(r^{1:K}|z^{1:K}, w, M)$. This architecture generalizes the reference resolution

---

[6]The bidirectional encoder only has access to the utterances that have been produced so far, i.e., $u_{1:t}$ when the agent is generating utterance u_{t+1}.
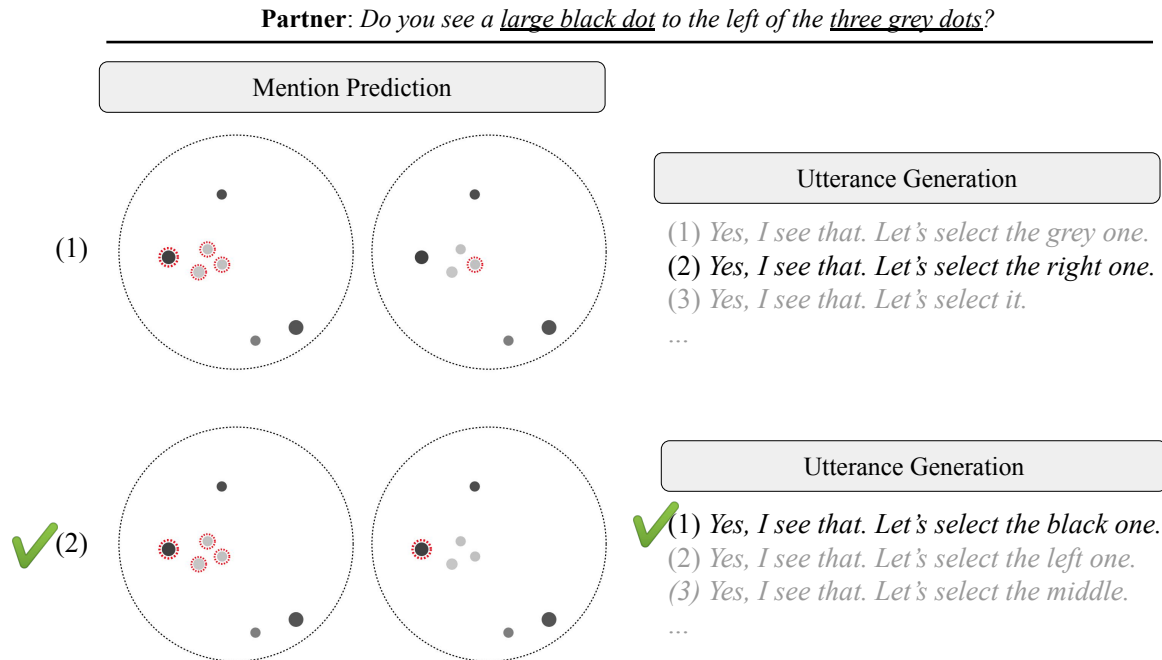
Figure 4.3: Agents optimize for a combination of fluency and informativity during pragmatic utterance generation (Section 4.3 and Algorithm 2). A set of candidate referents and utterance pairs is rescored using $L(\mathbf{r}, u)$, a weighted geometric mean of scores from the mention prediction, utterance, and reference resolution modules. The pair maximizing this score is chosen as a response.

and choice selection models of Udagawa and Aizawa (2020) and Udagawa et al. (2020) to model, in the output structure, relationships between dots, both inside and across referring expressions.

There are three different types of potentials, designed to model language-conditioned features of individual dots d in a referent r, ϕ; relationships within a referent, ψ, and transitions between successive referents, ω. Given these potentials, the distribution is parameterized as

$$P(r^{1:K}|z^{1:K}) \propto \exp\left(\sum_k f(r^k, z^k) + \psi(r^k, z^k) + \omega(r^{k:k+1}, z^{k:k+1})\right),$$

where $f(r, z) = \sum_{d \in r} \phi(d, z)$, and we've omitted the dependence of all terms on M and w for brevity. We share all module parameters across the two subtasks of resolving referents for the agent and for the partner.[7]

Individual Dots. Dot potentials ϕ model the correspondence between language features z^k and individual dots represented by encodings $w(d)$, as well as discourse salience using the

---

[7]Parameters are shared for efficiency; sharing had little effect on performance in preliminary experiments.

dot-level memory that tracks when the dot has been mentioned:[8]

$$\phi(d, z^k) = \text{MLP}_\phi([M(d), z^k, w(d)])$$

Dot Configurations. Configuration potentials $\psi(r^k, z^k)$ model the corresondence between language features and the set of all active dots in the agent's view for a referent r^k. These potentials further decompose into (1) pairwise potentials between active dots in the configuration, which relate the language embedding z^k to attribute differences between dots in the pair (including as relative position, size, and shade) and (2) a potential on the entire configuration, which relates the language embedding to an embedding for the count of active dots in the configuration.

Dot configuration potentials $\psi(r, z)$ are composed of two terms: $R(r, z)$ which decomposes into functions of pairwise relationships between the dots (whether active or not) in the context w and the text features z, and $A(r, z)$ which is a function of all active dots in the referent.

$$\psi(r, z) = R(r, z) + A(r, z)$$

The pairwise relationships are

$$R(r, z) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \alpha(r, z, i, j)$$

where N is the number of dots in view (7) and α is a scalar-valued neural function of the text features and whether the dots indexed by i and j are active in the referent r:

$$\alpha(r, z, i, j) = \begin{cases} p(z, i, j)_0, & r(i) \land r(j) \\ p(z, i, j)_1, & \neg r(i) \land \neg r(j) \\ p(z, i, j)_2, & \text{otherwise} \end{cases}$$

p is a 3-dimensional vector produced by an MLP:

$$p(z, i, j) = \text{MLP}_\psi([w(i) - w(j), z])$$

The active dot potential A is given by

$$A(r, z) = \text{MLP}_A([w(r), e(r)])$$

where $w(r)$ is the mean of the feature values for the active dots in r, $\frac{1}{|r_{active}|} \sum_{d \in r_{active}} w(d)$ and $e(r)$ is a learned 40-dimensional embedding for the discrete count of active dots in r.

---

[8]For the subtasks of reference resolution and choice selection, dot potentials are the same as the *attention module* used by Udagawa and Aizawa (2020), with the addition of the memory M.

Configuration Transitions. Transition potentials $\omega(r^k, r^{k+1}, z^k, z^{k+1})$ model the correspondence between language features and relationships between referring expressions, e.g., *to the left of* in <u>the black dot</u> to the left of <u>the triangle of gray dots</u>.

Configuration transitions $\omega(r^{k:k+1}, z^{k:k+1})$ are similar to the dot configuration potentials above but bridge the dots in referents k and $k+1$. They are the sum of two terms: $\omega(r^{k:k+1}, z^{k:k+1}) = S(r^{k:k+1}, z^{k:k+1}) + B(r^{k:k+1}, z^{k:k+1})$. S decomposes into pairwise relationships between dots across referents r^k and r^{k+1}, and B is a function of the feature centroids of the active dots in referents k and $k+1$.

$$S(r^{k:k+1}, z^{k:k+1}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \beta(r^{k:k+1}, z^{k:k+1}, i, j)$$

$$\beta(r^{k:k+1}, z^{k:k+1}, i, j) = \begin{cases} q_0, & r^k(i) \wedge r^{k+1}(j) \\ q_1, & \neg r^k(i) \wedge \neg r^{k+1}(j) \\ q_2, & \text{otherwise} \end{cases}$$

q (short for $q(z^{k:k+1}, i, j)$) is, like p in Dot Configurations, a 3-dimensional vector produced by an MLP:

$$q = \text{MLP}_{\omega}([w(i) - w(j), z^k - z^{k+1}])$$

B is defined analogously to A in Dot Configurations:

$$B(r^{k:k+1}, z^{k:k+1}) = \text{MLP}_B([w(r^k) - w(r^{k+1}), z^k - z^{k+1}])$$

with $w(r)$ again giving the mean of the feature values for the active dots in r. We fix $B(r^{k:k+1}, z^{k:k+1}) = 0$ if $|r^k_{active}| > 3$ or $|r^{k+1}_{active}| > 3$, which had little effect on model accuracy but improves memory efficiency.

Inference. We compute the normalizer for the CRF by enumerating the possible 2^7 assignments to each r^k to compute the ϕ, ψ, and ω potential terms, which can be performed efficiently on a GPU. To compute the normalizing constant, which sums over all combinations of assignments to these r^k, we use the standard linear-chain dynamic program. In training, we backpropagate through the enumeration and dynamic program steps to pass gradients to the parameters of the potential functions.

Confirmations

When applied to the partner's utterances, the reference resolution module gives a distribution over which referents the *partner* is likely to be referring to in the *agent's* own context. If the agent can identify the referents its partner is describing, it should be able to confirm them, both in the dots it talks about next (e.g., choosing to refer to one of the same dots

the partner identified) and in the text of the utterances (e.g., *yes, I see it*). The discrete-valued confirmation variable (defined in Section 4.2) models this, taking the value NA if no referring expressions were identified in the partner's utterance, YES if all of the $K > 0$ referring expressions have a non-empty referent (at least one dot predicted in the agent's context) and NO otherwise.

Referent Memory

The memory state is composed of one state vector $M_t(d)$ for each dot in the agent's own context. These dot states are updated using the referents identified in each utterance. This update is parameterized using a decoder cell, which is applied separately to each dot state:

$$M_{t+1}(d) = \text{RNN}_C(M_t(d), \iota(d, \mathbf{r}_t))$$

where ι is a function that extracts features from the predictive distribution over referents from the previous utterance, representing mentions of dot d in the referring expressions.

The function ι collapses predicted values for the dot d over K referents into a single representation for the dot, which we do in two ways: by max- and average- pooling predicted values for d across the K referents. We also obtain the prediction values in two ways: by taking the argmax structured prediction from P_R, and by taking the argmax predictions from each dot's marginal distribution. We found that using these "hard" argmax predicted values gave slightly better results in early experiments than using the "soft" probabilities from P_R. In combination, these give four feature values as the output of $\iota(d, \mathbf{r}_t)$.

Mention Selection

The mention selection subtask requires predicting a sequence of referents to mention in the agent's next utterance, $P_M(\mathbf{r}_{t+1} \mid u_{1:t}, M_{t+1}, c_{t+1}, w)$. To produce these referents, we use the same structured CRF architecture as the reference resolution module P_R. However, we use separate parameters from that module, and instead of the referring-expression inputs $\mathbf{z}$ use a sequence of vectors $x^{1:K_{t+1}}$ produced by a decoder cell RNN_M. The decoder conditions on the dialogue context representation H_t from the end of the last utterance, a learned vector embedding for the confirmation variable c_{t+1}, and a mean-pooled representation of the memory $m = \frac{1}{|d|} \sum_d M(d)$:

$$x^k = \text{RNN}_M(x^{k-1}, [H_t, c_{t+1}, m])$$

We obtain the number of referents K_{t+1} by predicting at each step k whether to halt from each x^k using a linear layer followed by a logistic function.

Choice Selection

To parameterize the choice selection module $P_S(s \mid u_{1:T}, M_T, w)$, we again reuse the CRF architecture, with independent parameters from reference resolution and mention selection

modules, replacing reference resolution's inputs $z^{1:K}$ with the dialogue context representation H_T from the end of the final utterance in the dialogue. Since only a single dot needs to be identified, we use only the CRF's individual dot potentials ϕ, removing ψ and ω. This is equivalent to the choice selection model (TSEL) used by Udagawa and Aizawa (2020) if the recurrent memory M_T is removed.

Utterance Generation

The utterance generation module $P_U(u_{t+1}|\mathbf{r}_{t+1}, c_{t+1}, H_t, w)$ is a sequence-to-sequence model. The module first uses a bidirectional LSTM to encode the sequence of K referents-to-mention $\mathbf{r}_{t+1} = r_{t+1}^{1:K}$ (predicted by the mention selection module), using the inputs at each position $k \in [1, K]$ a mean-pooled representation of the world context embeddings for the active dots in the referent: $\frac{1}{|r_{t+1}^k|} \sum_{d \in r_{t+1}^k} w(d)$, to produce a sequence of encoded vectors $y_t^{1:K}$.

Words in the utterance are then produced one at a time using a recurrent decoder that has a hidden state initialized with a function that combines $y_t^{1:K}$, the dialog context H_t, and a learned embedding for the discrete confirmation variable c_{t+1}. The decoder has two attention mechanisms over: (i) dot encodings $w(d)$, following Udagawa and Aizawa (2020), and (ii) the sequence of encoded referents $y_t^{1:K_{t+1}}$. We make gated updates to the decoder's initial state, updating it with (i) a linear projection of the forward and backward vectors for y_t^1 and y_t^K, representing the referent context and (ii) an embedding for the discrete confirmation variable c_{t+1}.

Implementation Details

For our reimplementation of the system of Udagawa and Aizawa (2020) in a shared codebase with our system, we replace their *tanh* non-linearities with `ReLU`s and use PyTorch's default initializations for all parameters. These improve performance across all evaluation conditions in comparison to the reported results.

For our system, we use separate word-level recurrent models, a Reader and a Writer, to summarize the dialogue history. The Reader is bidirectional over each utterance, and is used in the reference resolution and choice selection modules. The Writer is unidirectional, and is used in the mention selection and utterance generation modules. Model hyperparameters are given in Table 4.1.

For our full system and ablations, we train on each cross-validation fold for 12 epochs using the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 1×10^{-3}. We use a weighted combination of losses for the subtask objectives, and decay the learning rate when the loss plateaus on validation data.

We train models on a Quadro RTX 6000 GPU. Training takes around 1 day for models that use the structured CRF, and several hours without the structured CRF. Self-play evaluation takes around 1 hour.

Parameter Name	Value
Recurrent Unit Hyperparameters	
Reader GRU size	512
Writer GRU size	512
Mention decoder RNN_M size	512
Referent memory RNN_C size	64
Confirmation embedding c size	512
CRF Hyperparameters	
MLP_ϕ hidden layers	2
MLP_ϕ hidden size	256
MLP_ϕ dropout	0.5
MLP_ψ and MLP_ω hidden size	64
MLP_ψ and MLP_ω dropout	0.2
MLP_ψ and MLP_ω hidden layers	1
MLP_A and MLP_B hidden size	64
MLP_A and MLP_B dropout	0.2
MLP_A and MLP_B hidden layers	1
Generation Hyperparameters	
Sampling temperature in P_U	0.25
# Utterance candidates, N_u	100
# Referent candidates, N_r	20
Mention weight, w_M	0
Speaker weight, w_S	1×10^{-3}
Listener weight, w_L	$1 - w_S$
Early-stopping threshold, τ	0.8

Table 4.1: Model hyperparameters

4.5   Experiments

We compare our approach to past systems for the ONECOMMON dataset. While our primary evaluation is to evaluate systems on their success rate on the full dialogue game when paired with human partners (Section 4.5), we also compare our system to past work, and ablated versions of our full system, using the automatic evaluations of past work.

Models

We compare our full system (FULL) to ablated versions of it that successively remove: (i) the referent memory, ablating explict tracking of referents mentioned (F-MEM) and (ii) the

structured potentials ψ, γ in the reference resolution and mention selection modules (F-MEM -STRUC), removing explicit modeling of relationships within and across referents. We also compare to a reimplementation of the system of Udagawa and Aizawa (2020), which we found obtained better performance than their reported results in all evaluation conditions due to implementation improvements.

We obtain supervision for all components of the systems by training on the referent-annotated corpus of 5,191 successful human–human dialogues collected by Udagawa and Aizawa (2019; 2020). We train one copy of each model on each of the corpus's 10 cross-validation splits. We report means and standard deviations across the splits' models, except in human evaluations where we evaluate a single model.

Corpus Evaluation

Following Udagawa and Aizawa (2020), we evaluate models' accuracy at (1) predicting the dot chosen at the end of the game (Choice Acc.) using P_S and (2) resolving the referents in utterances from the human partner in the dialogue who had the agent's view (Ref Resolution dot-level accuracy Acc. and exact match accuracy Ex.) using P_R.

We see in Table 4.2 that our FULL model improves substantially on past work, including the work of Udagawa et al. (2020), who augment their referent resolution model with numeric features. Our structured reference resolver is able to learn these features in its potentials ψ (in addition to other structured relationships), and improves exact match from 44% to 76% compared to the ablated version of our system. Our recurrent memory helps in particular for the choice selection task, improving from 71% to 83% accuracy.

Model	Choice Acc.	Ref. Resolution Acc.	Ex.
U&A (2020)	69.3±2.0	86.4±0.4	35.0±2.0
U+ (2020)	–	86.0±0.3	54.9±0.8
F-MEM -STRUC	71.6±0.9	87.7±0.2	44.3±0.5
F-MEM	70.9±1.2	92.6±0.2	76.2±0.5
FULL	83.3±1.2	93.3±0.2	78.2±0.5
Human	90.8	96.3	86.9

Table 4.2: Accuracies for predicting the dot selected at the end of the game (Choice Acc.) and resolving referents from utterances produced in the agent's own perspective (dot-level accuracy Acc. and exact match Ex.) in 10-fold cross-validation on the corpus. Our FULL model outperforms all past work on the dataset: U&A (2020) is our reimplementation of Udagawa and Aizawa (2020), and U+ (2020) are taken from Udagawa et al. (2020). Human scores are annotator agreements (Udagawa and Aizawa, 2019).

| | Partner Refs. | | Next Refs |
Model	Acc.	Ex.	Ex.
F-MEM -STRUC	87.3±0.3	41.8±0.9	4.8±1.0
F-MEM	90.6±0.3	65.2±1.0	23.5±2.0
FULL	91.2±0.4	67.0±1.0	31.1±1.0

Table 4.3: Accuracies for resolving referents in the *partner*'s view (dot-level accuracy Acc. and exact match Ex.) and predicting the next referents to mention in the dialogue (Next Refs Ex.) in 10-fold cross-validation on the corpus of human–human dialogues. Our FULL benefits from its recurrent referent memory (outperforming F-MEM) and structured referent prediction module (outperforming F-MEM -STRUC).

We also compare the performance of our full and ablated systems on the tasks of resolving the partner's referring expressions and mention prediction. Table 4.3 gives performance accuracies for resolving referents in the *partner*'s view (dot-level accuracy Acc. and exact match Ex.) and predicting the next referents to mention in the dialogue (Next Refs Ex.) in 10-fold cross-validation on the corpus of human–human dialogues. We observe improvements from both the recurrent memory (comparing F-Mem to Full) and the structured referent prediction module (comparing F-Mem-Struc to Full) on both tasks.

Evaluation in Self-Play

To evaluate systems on the full dialogue task, we first use self-play, where a system is partnered with a copy of itself, following Udagawa and Aizawa (2020). We evaluate systems on 3000 world contexts, stratified into contexts with 4, 5, and 6 dots overlapping between the two agents' views, with 1000 contexts in each stratification.

Table 4.4 reports average task success (the fraction of times both agents chose the same dot at the end of the dialogue) averaged across the 10 copies of each model trained on the cross-validation splits. As in the corpus evaluation, we see substantial improvements to our system from the structured referent prediction and the recurrent reference memory. Our Full system, without pragmatic generation, improves over the system of Udagawa and Aizawa (2020) from 51% to 58% in the hardest setting, with a further improvement to 62% when adding our pragmatic generation procedure.

Human Evaluation

Finally, we perform human evaluation by comparing system performance when playing with workers from Amazon's Mechanical Turk (MTurk). To conduct evaluation, we used 100 world states from the #Shared=4 partition, and collected 718 complete dialogues by ran-

Model	#Shared=4	#Shared=5	#Shared=6
U&A (2020)	50.7±2.0	66.0±1.9	83.5±1.5
F-Mem -Struc	42.3±2.1	57.0±2.1	75.4±1.1
F-Mem	52.6±1.5	67.1±1.9	84.1±1.6
Full	58.5±2.7	71.6±2.9	86.8±1.8
Full+Prag	62.4±2.2	74.7±2.7	90.9±1.4
Human	65.8	77.0	87.0

Table 4.4: Task success rates in automatic self-play evaluations, by difficulty of context (the number of items shared in the players' views). Our Full model outperforms past work: U&A (2020) is our tuned reimplementation of Udagawa and Aizawa (2020). Human shows success rates of trained human annotators in collecting the dataset (Udagawa and Aizawa, 2019).

domly pairing worker with one of the following three: our best-performing model in self-play (Full+Prag), the model from Udagawa and Aizawa (2020), or another worker.

In order to ensure higher quality dialogues, and following Udagawa and Aizawa (2019), we filtered workers by qualifications, showed workers a game tutorial before playing, and prevented dots from being selected within the first minute of the game. We paid workers $0.30 per game, with a bonus of $0.15 if the dialogue was successful. See Appendix B for sample dialogues.

We compare systems based on the percentage of successful dialogues. The results, in Figure 4.4, corroborate the trends observed in self-play. Both the models of U&A (2020) and our Full+Prag perform worse against humans than against agent partners in the automatic self-play evaluation, illustrating the importance of performing human evaluations. However, the trend is preserved, and we see that the Full+Prag system substantially outperforms the U&A (2020) model, resulting in a 50% relative improvement in task success rate. This difference is statistically significant at the $p \leq 0.05$ level using a one-tailed t-test.

Success by Human Skill Level

In Section 4.5, we compared our systems to a human population of MTurk workers. However, human populations themselves vary greatly based on many factors, including the day and time workers are recruited, training and feedback given to workers, and worker retention. One difference between our worker population and the population that produced the dataset is training. When collecting the dataset, Udagawa and Aizawa (2019) performed manual and individualized coaching of their MTurk workers which made them more effective at the game: giving players personalized feedback on how to improve their game strategies, e.g.,

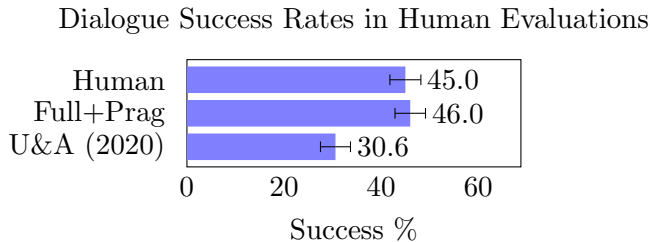Dialogue Success Rates in Human Evaluations

Figure 4.4: Success rates of systems on the full dialogue game task when paired with human partners. Error bars show standard errors. Our FULL+PRAG system achieves a 50% relative performance improvement over past work (U&A 2020).

"please ask more clarification questions."[9] Manual coaching produced a high-quality corpus by increasing players' skill and obtained a success rate of 66%; however coaching would make human evaluations difficult to replicate across works due to the labor, cost, and variability that coaching involves.

In this section, we run a sweep of system comparisons of the form of Section 4.5, but on increasingly select sub-populations of MTurk workers. Results are shown in Figure 4.5. The x-axis gives the minimum overall success rate for a worker's games to be retained, so that the far left of the graph shows all workers (corresponding to the numbers in Figure 4.4), the far right shows only those workers who won all of their games, and the black vertical line marks the player filtering needed to obtain a human-human success rate comparable to Udagawa and Aizawa (2019). Our FULL+PRAG system outperforms the model of Udagawa and Aizawa (2020) at all player skill levels.[10] This result shows that, while more accomplished workers' overall success rates can be much higher than the success rate of our general worker population, in all cases the ordering between the two systems remained the same.

4.6   Related Work

Goal-Oriented Dialog.   The modular approach that we use reflects the pipelined approach traditionally used in goal-oriented dialogue systems (Young et al., 2013). Recent work on neural systems has also used structured and memory-based approaches (Bordes and Weston, 2016; He et al., 2018) including tracking entities identified in text (Williams et al., 2017; He et al., 2017). We also find improvements from an entity-centric approach with a

---

[9]Udagawa and Aizawa also manually removed around 1% of dialogues where workers did not follow instructions. While we do not perform post-hoc manual filtering of the dialogues, in order to avoid introducing systematic bias that would favor or disfavor one of the systems we compare, an inspection of a subset of our collected dialogues indicates a similarly high fraction of our workers were making a good effort at the task.

[10]Until, by necessity, the point where filtering removes all workers who lost a game against any system. Differences between U&A'20 and FULL+PRAG are significant at the $p \leq 0.05$ level by a one-tailed t-test for minimum worker successes between 0 and 0.5.
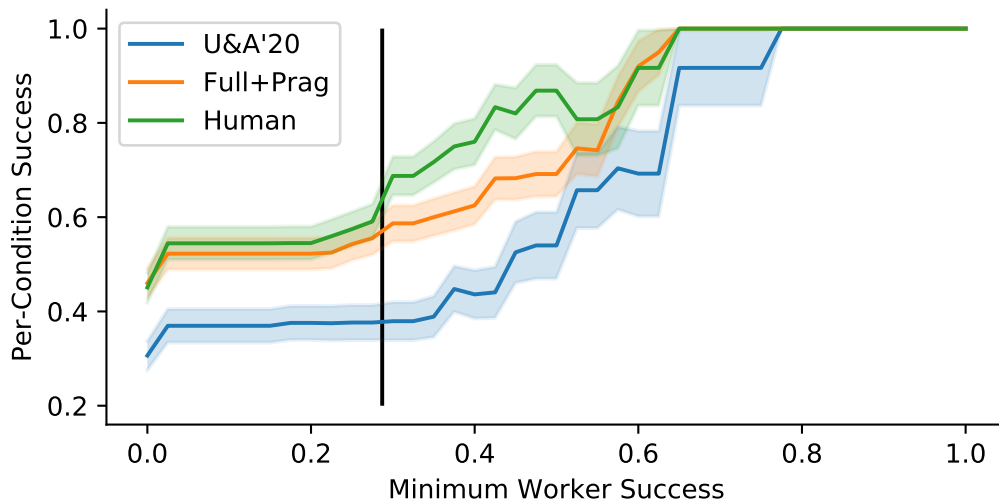
Figure 4.5: Success rates of human players against each system type, and other humans, with progressive filtering of humans by their overall success rate (across all conditions) along the x-axis. Shaded regions give standard errors. Our FULL+PRAG system outperforms the model from Udagawa and Aizawa (2020) at all levels.

structured memory, although our domain involves more challenging entity resolution and generation due to the spatial grounding.

Pragmatics. Our approach to pragmatics (Grice, 1975) builds on a long line of work in the RSA framework (Frank and Goodman, 2012), which models how speakers and listeners reason about each other to communicate successfully. The most similar applications to ours in past work on computational pragmatics have been to single-turn grounded reference tasks (rather than dialogue), with much smaller and unstructured spaces of referents than ours,[11] such as discriminative image captioning (Vedantam et al., 2017; Andreas and Klein, 2016; Cohn-Gordon et al., 2018) and referent identification (Monroe et al., 2017; McDowell and Goodman, 2019; White et al., 2020). Explicit speaker–listener models of pragmatics have also been used for dialog, but either in ungrounded settings (Kim et al., 2020) or with constrained language (Vogel et al., 2013; Khani et al., 2018).

Collaborative Games. The closest work on dialogue systems for collaborative grounded tasks has focused on tasks with different properties from ours, as discussed at the beginning of this chapter. A closely related task to the shared visual reference game we pursue here is the PhotoBook task (Haber et al., 2019), although a dialogue system has not been constructed for it. Other work on collaborative language games in grounded environments includes collection games (Potts, 2012; Suhr et al., 2019), navigation and interactive question answering games

---

[11] Our setting has 2^7 possible referents for each referring expression in the dialogue.

(Thomason et al., 2019; Nguyen and Daumé III, 2019), and construction tasks (Wang et al., 2017; Kim et al., 2019; Narayan-Chen et al., 2019).

4.7 Discussion

We presented a modular, reference-centric approach to a challenging partially-observable grounded collaborative dialogue task. Our approach is centered around a structured referent grounding module, which is used to interpret a partner's utterances as well as to enable a pragmatic generation procedure that encourages the agent's utterances to be able to be understood in context. Our full system substantially outperforms past work in human evaluations on the full dialogue task.

Chapter 5

Conclusion

This thesis demonstrated how language systems can benefit from treating communication as a cooperative multi-agent process. Our perspective is a classic one in pragmatics: we view language as an action that agents take to produce effects on other agents and thereby have effects in the world. We implemented this multi-agent perspective and showed that it benefits real-world language systems: we built explicit models of our systems' human partners and had the systems reason about these models in simulation.

Our findings here suggest that: (1) explicitly inferring the effects that language will have helps to ensure that generated language is contextually appropriate and correctly interpretable by human partners (Chapter 2 and Chapter 4) and (2) explicitly inferring the intended effects a human partner may have had in mind helps to deal with challenges of ambiguity and rich grounding contexts (Chapter 2 and Chapter 3).

The successes of explicit pragmatic inference in the tasks we've examined here suggest natural directions for future work. First, our models of human partners have been population-level models which are designed to model human behavior in aggregate. Explicit models could also be tailored and adapted to individual people to capture their language use, likely actions, and preferences. Second, the grounded tasks we've explored here have taken place in perfectly-simulable environments with precisely formulable objectives (e.g., task success), while many real-world tasks of interest involve stochastic and imperfectly-simulable environments with fuzzy success criteria. Finally, our work has focused most directly on cooperativity at the level of single language acts: while our environments are perceptually rich, our referents have complex structure, and our models incorporate utterance and action history, our inference procedures only consider a speaker conveying meaning to a listener in a single utterance. But for broader tasks, single utterances won't suffice: communication requires partners to undertake longer-term collaborations to build common ground (Clark and Wilkes-Gibbs, 1986; Traum, 1994; Clark, 1996) using both linguistic and non-linguistic acts (Clark and Brennan, 1991). Future work might tackle these challenges using collaborative multi-agent reasoning, allowing systems to better interact with human partners to ground meaning and achieve success in real-world environments.

Bibliography

Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, et al. 1991. The HCRC map task corpus. *Language and speech* 34(4):351–366.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for CCG semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 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 for Computational Linguistics (TACL)* 1(1):49–62.

John Langshaw Austin. 1962. *How to do things with words*, volume 88. Oxford University Press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Leon Bergen, Roger Levy, and Noah Goodman. 2016. Pragmatic reasoning through semantic inference. *Semantics and Pragmatics* 9.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR* abs/1605.07683.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Donna Byron, Alexander Koller, Kristina Striegnitz, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2009. Report on the first NLG challenge on generating instructions in virtual environments (GIVE). In *Proceedings of the 12th European Workshop on Natural Language Generation*.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)* .

David L. Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Kan Chen, Rama Kovvuri, and Ram Nevatia. 2017. Query-guided regression network with context policy for phrase grounding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Volkan Cirik, Taylor Berg-Kirkpatrick, and Louis-Phillippe Morency. 2018. Using syntax to ground referring expressions in natural images. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In L.B. Resnick, J.M. Levine, and S.D. Teasley, editors, *Perspectives on Socially Shared Cognition*, American Psychological Association, pages 127–149.

Herbert H. Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition* 22(1):1–39.

Reuben Cohn-Gordon, Noah Goodman, and Chris Potts. 2018. Pragmatically informative image captioning with character-level reference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Robert Dale, Sabine Geldof, and Jean-Philippe Prost. 2005. Using natural language generation in automatic route description. *Journal of Research and Practice in Information Technology* 37(1):89–105.

Andrea F. Daniele, Mohit Bansal, and Matthew R. Walter. 2017. Navigational instruction generation as inverse reinforcement learning with neural machine translation. *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)* .

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. 2018. Talk the Walk: Navigating New York City through grounded dialogue. *CoRR* abs/1807.03367.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science* 336(6084):998–998.

Michael C Frank, Noah D Goodman, Peter Lai, and Joshua B Tenenbaum. 2009. Informative communication in word production and word learning. In *Proceedings of the 31st annual conference of the cognitive science society*. pages 1228–1233.

Michael Franke. 2013. Game theoretic pragmatics. *Philosophy Compass* 8(3):269–284.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018a. Unified pragmatic models for generating and following instructions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

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

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. volume 9, pages 249–256.

Robert Goeddel and Edwin Olson. 2012. DART: A particle-based method for generating easy-to-follow directions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Noah D Goodman and Michael C Frank. 2016. Pragmatic language interpretation as probabilistic inference. *Trends in Cognitive Sciences* 20:818–829.

Noah D Goodman and Andreas Stuhlmüller. 2013. Knowledge and implicature: Modeling language understanding as social cognition. *Topics in Cognitive Science* 5(1):173–184.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems* 28:2222–2232.

H. Paul Grice. 1975. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Speech Acts*, Academic Press, New York, volume 3 of *Syntax and Semantics*, pages 41–58.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR* abs/1503.03535.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Janosch Haber, Tim Baumgärtner, Ece Takmaz, Lieke Gelderloos, Elia Bruni, and Raquel Fernández. 2019. The PhotoBook dataset: Building common ground through visually-grounded dialogue. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. 2017. Grounded language learning in a simulated 3D world. *CoRR* abs/1706.06551.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2017. Modeling relationships in referential expressions with compositional modular networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. 2016a. Segmentation from natural language expressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016b. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR* abs/1508.01991.

Gerhard Jäger. 2012. Game theory in semantics and pragmatics. *Semantics: An international handbook of natural language meaning* 3:2487–2516.

Hong Jun Jeon, Smitha Milli, and Anca D Dragan. 2020. Reward-rational (implicit) choice: A unifying formalism for reward learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. ReferItGame: Referring to objects in photographs of natural scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Fereshte Khani, Noah D Goodman, and Percy Liang. 2018. Planning, inference and pragmatics in sequential language games. *Transactions of the Association for Computational Linguistics* 6:543–555.

Hyunwoo Kim, Byeongchang Kim, and Gunhee Kim. 2020. Will I sound like me? Improving persona consistency in dialogues through pragmatic self-consciousness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jin-Hwa Kim, Nikita Kitaev, Xinlei Chen, Marcus Rohrbach, Byoung-Tak Zhang, Yuandong Tian, Dhruv Batra, and Devi Parikh. 2019. CoDraw: Collaborative drawing as a testbed

for grounded goal-driven communication. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations (ICLR)* .

Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. Report on the second NLG challenge on generating instructions in virtual environments (GIVE-2). In *Proceedings of the 6th International Natural Language Generation Conference*.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Emiel Krahmer and Mariët Theune, editors. 2010. *Empirical Methods in Natural Language Generation: Data-oriented Methods and Empirical Evaluation*. Springer-Verlag, Berlin, Heidelberg.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? End-to-end learning for negotiation dialogues. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Chenxi Liu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, and Alan Yuille. 2017. Recurrent multimodal interaction for referring image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Gary Wai Keung Look. 2008. *Cognitively-inspired direction giving*. Ph.D. thesis, Massachusetts Institute of Technology.

Ruotian Luo and Gregory Shakhnarovich. 2017. Comprehension-guided referring expressions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. *Proceedings of the Conference on Artificial Intelligence (AAAI)* .

Junhua Mao, Huang Jonathan, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Bill McDowell and Noah Goodman. 2019. Learning from omission. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Florence, Italy.

Hongyuan Mei, Mohit Bansal, and Matthew Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

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 (EMNLP)*.

Will Monroe, Robert X.D. Hawkins, Noah D. Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics (TACL)* 5:325–338.

Will Monroe and Christopher Potts. 2015. Learning in the Rational Speech Acts model. In *Proceedings of 20th Amsterdam Colloquium*. ILLC, Amsterdam.

Varun K Nagaraja, Vlad I Morariu, and Larry S Davis. 2016. Modeling context between objects for referring expression understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

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 (ACL)*.

Khanh Nguyen and Hal Daumé III. 2019. Help, Anna! Visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Prashant Parikh. 2001. *The use of language*. CSLI Publications, Stanford, CA.

Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. 2018. Zero-shot visual imitation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bryan Plummer, Liwei Wang, Chris Cervantes, Juan Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Christopher Potts. 2011. *Pragmatics*, Oxford University Press.

Christopher Potts. 2012. Goal-driven answers in the Cards dialogue corpus. In Nathan Arnett and Ryan Bennett, editors, *Proceedings of the 30th West Coast Conference on Formal Linguistics*. Cascadilla Press, Somerville, MA.

Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. 2017. Data distillation: Towards omni-supervised learning. *arXiv preprint arXiv:1712.04440* .

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Seymour Rosenberg and Bertram D. Cohen. 1964. Speakers' and listeners' processes in a word-communication task. *Science* 145(3637):1201–1203.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

H. J. Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory* 11(3):363–371.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR* abs/1712.01815.

Nathaniel J. Smith, Noah Goodman, and Michael Frank. 2013. Learning and using language via recursive pragmatic reasoning about other agents. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Kristina Striegnitz, Alexandre Denis, Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Mariët Theune. 2011. Report on the second second challenge on generating instructions in virtual environments GIVE-2.5. In *Proceedings of the 13th European Workshop on Natural Language Generation*.

Jong-Chyi Su, Chenyun Wu, Huaizu Jiang, and Subhransu Maji. 2017. Reasoning about fine-grained attribute phrases using reference games. In *International Conference on Computer Vision*.

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 (EMNLP)*.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. 2017. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407* .

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Richard S. Sutton, David A McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence*.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. 2019. Vision-and-dialog navigation. In *Conference on Robot Learning (CoRL)*.

David R Traum. 1994. A computational theory of grounding in natural language conversation. Technical report, Rochester University Department of Computer Science.

Takuma Udagawa and Akiko Aizawa. 2019. A natural language corpus of common grounding under continuous and partially-observable context. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Takuma Udagawa and Akiko Aizawa. 2020. An annotated corpus of reference resolution for interpreting common grounding. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Takuma Udagawa, Takato Yamazaki, and Akiko Aizawa. 2020. A linguistic analysis of visually grounded dialogues based on spatial expressions. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Arun Balajee Vasudevan, Dengxin Dai, and Luc Van Gool. 2018. Object referring in visual scene with spoken language. In *Winter Conference on Applications of Computer Vision (WACV)*.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Adam Vogel, Max Bodoia, Christopher Potts, and Daniel Jurafsky. 2013. Emergence of Gricean maxims from multi-agent decision theory. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Mingzhe Wang, Mahmoud Azab, Noriyuki Kojima, Rada Mihalcea, and Jia Deng. 2016a. Structured matching for phrase localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Sida I. Wang, Samuel Ginn, Percy Liang, and Christopher D. Manning. 2017. Naturalizing a programming language via interactive learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016b. Learning language games through interaction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. 2018. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. 2017. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Julia White, Jesse Mu, and Noah D. Goodman. 2020. Learning to refer informatively by amortizing pragmatic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci)*.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tom Williams, Gordon Briggs, Bradley Oosterveld, and Matthias Scheutz. 2015. Going beyond literal command-based instructions: Extending robotic natural language interaction capabilities. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

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.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2017a. The neural noisy channel. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. 2018. MAttNet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. 2017b. A joint speaker-listener-reinforcer model for referring expressions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Appendix A

Submission to Vision and Language Navigation Challenge

We participated in the Vision and Language Navigation Challenge[1], an online challenge for the vision-and-language navigation task on the R2R dataset. We submitted the predictions from our full method to the evaluation server, using single models for the speaker and listener, without any additional ensembling. At the time of writing, our method (under the team name "Speaker-Follower") remains the top-performing method on the challenge leader-board with a success rate of 53.49% (the same success rate as in the Table 3.2 test split).

When generating the predictions for the challenge submission, we modified the method for generating final routes to comply with the challenge guidelines. In Table 3.1, Table 3.2 and Table 3.3, the performance of our full model with pragmatic inference is evaluated using a single top-ranking candidate route, where the candidate routes are generated with state-factored search in Section 3.2. Hence, our reported navigation error, success rate and oracle success rate are all computed by choosing a single candidate route per instruction. However, the challenge guidelines require that the submitted trajectories must be generated from *a single independent evaluation run*, where the agent must move sequentially and all its movements during inference must be recorded. So just returning the route selected by pragmatic inference, or by search, would violate the contest guidelines, as this route may not contain all world states explored during search. On the other hand, the agent is allowed to backtrack to a previous location on its path, as long as the agent does not teleport and all its movements are recorded in the final trajectory (which we confirmed with the challenge organizer).

To comply with the guidelines and maintain physical plausibility, we log all states visited by the search algorithm in the order they are traversed. The agent expands each route one step forward at a time, and then switches to expand the next route. When switching from one route to another route, the agent first backtracks to the closest common ancestor

---

[1]https://evalai.cloudcv.org/web/challenges/challenge-page/97/overview

state of the two routes in the search (which could be the starting location). From there it goes to the frontier of the other route and takes an action there to expand it. Once the set of candidate routes has been completed, they are ranked according to Equation 3.1 for pragmatic inference, selecting a route that ends at a target location. Finally, the agent moves from its last visited location in the search to this target location. It then takes the stop action to end the episode. As a result of this, all the agent's movements, including route expansion and route switching, are recorded in its final trajectory.

By design, the sequential inference procedure above yields *exactly the same success rate* as the pragmatic inference procedure, since it returns routes with the same end states. Unsurprisingly, the oracle success rate increases substantially (from 63.9% to 96.0%), since the resulting final trajectory records the visited locations from all routes and the oracle success rate is determined by distance between the ground-truth target location and the closest visited location. We also note that the agent's final trajectory is very long (1257.38m per instruction on average) since it needs to walk a substantial distance to sequentially expand all candidate routes and to switch between them.

In addition, we also evaluated the predictions from our model without pragmatic inference on the challenge server. Without pragmatic inference, our model achieves 35.08% success rate, 44.45% oracle success rate and 6.62m navigation error on the challenge test set, with an average trajectory length of 14.82m. This is close to the performance of our model under the same setting on val unseen split (Row 6 in Table 3.1).

Finally, we note that our method in this work is designed mostly to optimize success rate and navigation error, and we leave potential improvement to reduce trajectory length via inference (such as ordering the routes by location to reduce switching overhead) and modeling (such as building speaker models that can rank partial incomplete routes) to future work.

Appendix B

Dialogue Examples

We show one successful and one failed dialogue from our human evaluations (Section 4.5) for each system (Figure B.1).

(a) An unsuccessful dialogue between a human and the system of Udagawa and Aizawa (2020).
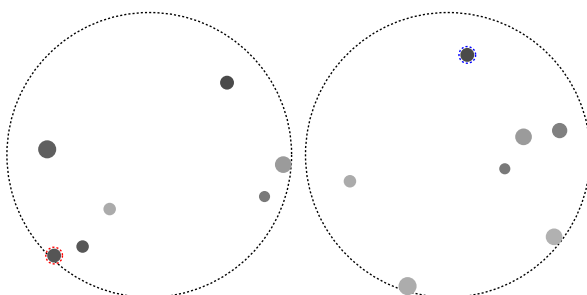
Human's view   U&A20's view

U&A'20: i have a large light gray dot with a small black dot above it and to the left
Human: I don't see that. I have a dark gray dot on the line between 7 and 8 o'clock.
U&A'20: yes
Human: Let's pick that!
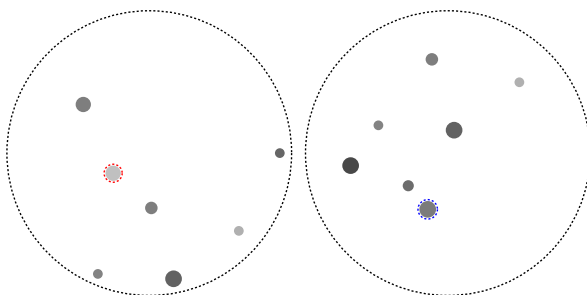U&A'20: ok
Human: SELECT red
U&A'20: SELECT blue

(b) A successful dialogue between a human and the system of Udagawa and Aizawa (2020).
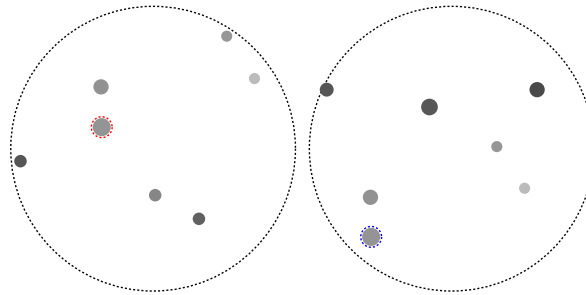
Human's view Full+Prag's view

U&A'20: i have a large black dot with a smaller light grey dot to the left of it
Human: I don't see that. I have a medium dark dor to the left of all the others.
U&A'20: i have a large black dot with a smaller one that is a little lighter and to the left of it .
Human: I have a large light gray with a medium light gray above it.
U&A'20: yes , let 's click the larger one
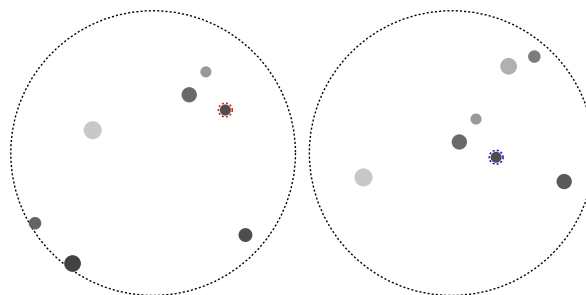Human: SELECT red
U&A'20: SELECT blue

(c) An unsuccessful dialogue between a human and our FULL+PRAG system.

Human's view Full+Prag's view

FULL+PRAG: large black dot with a small light grey dot above it
Human: Also have a large gray dot with 3 smaller dots around it
FULL+PRAG: i do n't have that
Human: Medium sized and light gray dot towards the middle
FULL+PRAG: yes , let 's pick the larger one
Human: SELECT red
FULL+PRAG: SELECT blue

(d) A successful dialogue between a human and our FULL+PRAG system.

Human's view Full+Prag's view

FULL+PRAG: i have a group of 3 dots , a tiny light grey and a large dark grey
Human: yes me too
FULL+PRAG: let us select the small black
Human: okay
FULL+PRAG: pick the small black dot
Human: okay i will
FULL+PRAG: ok
Human: SELECT red
FULL+PRAG: SELECT blue

Figure B.1: Example dialogues collected during our human evaluation (Section 4.5). We show one unsuccessful (left) and one successful (right) example for each system. The top row is our reimplementation of Udagawa and Aizawa (2020) and the bottom row is FULL+PRAG, our full system with pragmatic inference.