



museums, offices, and other public spaces. More generally, the problem is relevant beyond human-robot interaction to the broader domain of indoor navigation. There are two primary challenges to generating effective, natural language route instructions, which are characteristic of the more general problem of free-form generation.

The first challenge is *content selection*, the problem of deciding what and how much information to convey to the user as part of the directions. In general, the more detailed an instruction is, the less ambiguous it is. However, verbose instructions can be unnatural and hard for followers to remember and, thus, ineffective. Consequently, it is important to balance the value of including particular information as part of a route instruction with the cost that comes with increasing the level of detail. Further, not all information is equally informative. Existing commercial navigational solutions typically rely on a set of hand-crafted rules that consider only street names and metric distances as valid candidates, the latter of which requires that follower’s keep track of their progress. In contrast, studies have shown that people prefer route instructions that reference physical, salient landmarks in the environment [59]. However, no standard exists with regards to what and how these landmarks should be selected, as these depend on the nature of the environment and the demographics of the follower [62, 27].

We propose a method that models this content selection problem as a Markov decision process with a learned policy that decides what and how much to include in a formal language specification of the task (path). We learn this policy via inverse reinforcement learning from demonstrations of route instructions provided by humans. This avoids the need for hand-crafted selection rules, and allows our method to automatically adapt to the preferences and communication style of the target populations and to simultaneously choose to convey information that minimizes the ambiguity of the instruction while avoiding verbosity.

The second challenge is *surface realization*, which is the task of synthesizing a natural language sentence that refers to the selected content. Existing solutions rely on sentence templates, generating sentences by populating manually defined fields (e.g., “turn (direction)”) and then serializing these sentences in a turn-by-turn fashion. As expected, the use of such templates reduces coherence across sentences and limits the ability to adapt to different domains (e.g., from outdoor to indoor navigation). Additionally, while the output is technically correct, the resulting sentences tend to be rigid and unnatural. Studies show that language generated by a robot is most effective when it emulates the communication style that people use [57].

We address the surface realization problem through a neural sequence-to-sequence model that “translates” a formal language specification of the selected command into a natural language sentence. Our model takes the form of an encoder-aligner-decoder architecture that first encodes the formal path specification with a recurrent neural network using long short-term memory (LSTM-RNN) [25]. The model then decodes (translates) the resulting abstraction of the input into a natural language sentence (word sequence), using an alignment mechanism to further refine the selected information and associate output words with the corresponding elements in the input formal specification. The use of LSTMs as the hidden units enables our model to capture the long-term dependencies that exist among the selected

information and among the words in the resulting instruction. We train our surface realization model on instruction corpora, enabling our method to generate free-form directions that emulate the style of human instructions, without templates, specialized features, or linguistic resources.

We evaluate our method on the benchmark SAIL dataset of human-generated route instructions [39]. Instructions generated with our method achieve a sentence-level BLEU score of 72.18%, indicating their similarity with the reference set of human-provided instructions. We perform a series of ablations and visualizations to better understand the contributions of the primary components of our model. We additionally conduct human evaluation experiments that demonstrate that our method generates instructions that people are able to follow as efficiently and accurately as those generated by humans.

The paper presents what is, to the best of our knowledge, the first framework that generates natural language instructions that reflect the preferences and rhetorical style of humans. We achieve this firstly via a decision process that learns to emulate human content selection preferences from demonstrations; and secondly through a neural machine translation model that learns to express this content via natural language. Human experiments reveal that our method produces instructions that are as accurate, effective, and usable as those generated by humans.

## 2. RELATED WORK

Existing research related to the generation of route instructions spans the fields of robotics, natural language processing, cognitive science, and psychology. Early work in this area focuses on understanding the way in which humans generate natural language route instructions [62, 1, 38] and the properties that make “good” instructions easier for people to follow [36, 48, 59]. These studies have shown that people prefer to give directions as a sequence of turn-by-turn instructions and that they favor physical objects and locations as intuitive landmarks.

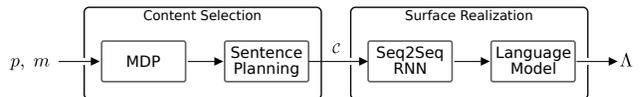
Based on these studies, much of the existing research on generating route instructions involves the use of hand-crafted rules that are designed to emulate the manner in which people compose navigation instructions [51, 13]. Look et al. [36] compose route instructions using a set of templates and application rules engineered based upon a corpus of human-generated route instructions. Look [37] improves upon this work by incorporating human cognitive spatial models to generate high-level route overviews that augment turn-by-turn directions. Similarly, Dale et al. [14] analyze a dataset of route instructions composed by people to derive a set of hand-designed rules that mimic the content and style of human directions. Goedel and Olson [18] describe a particle filter-based method that employs a generative model of direction following to produce templated instructions that maximize the likelihood of reaching the desired destination. Meanwhile, Morales Saiki et al. [44] propose a system that enables robots to provide route instructions using a combination of language and pointing gestures. Unlike our approach, their method is limited to paths with no more than three direction changes and employs four generation templates, limiting the content and structure of the instructions.

The challenge with instruction generation systems that rely upon hand-crafted rules is that it is difficult to design a policy that generalizes to a wide variety of scenarios and fol-

lowers, whose preferences vary depending on such factors as their cultural background [27] and gender [62]. Cuayáhuitl et al. [12] seek to improve upon this using reinforcement learning with hand-crafted reward functions that model the length of the instructions and the likelihood that they will confuse a follower. They then learn a policy that reasons both over the best route and the corresponding navigational instructions. However, this approach still requires that domain experts define the reward functions and specify model parameters. In contrast, Oswald et al. [45] model the problem of deciding what to include in the instruction (i.e., the content selection problem) as a Markov decision process and learn a policy from a human-generated navigation corpus using maximum entropy inverse reinforcement learning. Given the content identified by the policy, their framework does not perform surface realization, and instead generates instructions by matching the selected content with the nearest match in a database of human-generated instructions. Our method also uses inverse reinforcement learning for content selection, but unlike their system, our method also learns to perform surface realization directly from corpora, thus generating newly-composed natural language instructions.

Relatedly, much attention has been paid recently to the “inverse” problem of learning to follow (i.e., execute) natural language route instructions. Statistical methods primarily formulate the problem of converting instructions to actions as either a semantic parsing task [40, 9, 4] or as a symbol grounding problem [31, 54, 34, 26, 10]. Alternatively, Mei et al. [41] translate free-form instructions to action sequences in an end-to-end fashion using an encoder-aligner-decoder.

Meanwhile, selective generation considers the more general problem of converting a rich database to a natural language utterance, with existing methods generally focusing on the individual problems of content selection and surface realization. Barzilay and Lee [7] perform content selection on collections of unannotated documents for the sake of text summarization. Barzilay and Lapata [6] formulate content selection as a collective classification problem, simultaneously optimizing local label assignments and their pairwise relations. Liang et al. [35] consider the related problem of aligning elements of a database to textual description clauses. They propose a generative semi-Markov model that simultaneously segments text into utterances and aligns each utterance with its corresponding entry in the database. Meanwhile, Walker et al. [58] perform surface realization via sentence planners that can be trained to generate sentences for dialogue and context planning. Wong and Mooney [63] effectively invert a semantic parser to generate natural language sentences from formal meaning representations using synchronous context-free grammars. Rather than consider individual sub-problems, recent work focuses on solving selective generation via a single framework [8, 29, 3, 32, 42]. Angeli et al. [3] model content selection and surface realization as local decision problems via log-linear models and employ templates for generation. Mei et al. [42] formulate selective generation as an end-to-end learning problem and propose a recurrent neural network encoder-aligner-decoder model that jointly learns to perform content selection and surface realization from database-text pairs. Unlike our method, they do not reason over the correctness or unambiguity of the summaries, nor do they attempt to model human content selection preferences.



**Figure 2: Our method generates natural language instructions for a given map and path.**

### 3. TASK DEFINITION

We consider the problem of generating natural language instructions that allow humans to navigate environments that are unknown to them a priori. The given map  $m$  takes the form of a hybrid metric-topologic-semantic representation (Fig. 1) that encodes the position of and connectivity between a dense set of locations in the environment (e.g., intersections) and the position and type of objects and environment features (e.g., floor patterns). The path  $p$  is a sequence of poses (i.e., position and orientation) that corresponds to the minimum distance route from a given initial pose to a desired goal pose. We split the path according to changes in direction, representing the path  $p = (p_1, p_2, \dots, p_M)$  as a sequence of intermediate segments  $p_i$ .

Training data comes in the form of tuples  $(m^{(i)}, p^{(i)}, \Lambda^{(i)})$  for  $i = 1, 2, \dots, n$  drawn from human demonstrations, where  $m^{(i)}$  is a map of the environment,  $\Lambda^{(i)}$  is a human-generated natural language route instruction, and  $p^{(i)}$  is the path that a different human took when following the instructions. At test time, we consider only the map and path pair as known and hold out the human-generated instruction for evaluation. The dataset that we use for training, validation, and testing comes from the benchmark SAIL corpus [39].

### 4. MODEL

Given a map and a path, our framework (Fig. 2) performs content selection to decide what information to share with the human follower and subsequently performs surface realization to generate a natural language instruction according to this selected content. Our method learns to perform content selection and surface realization from human demonstrations, so as to produce instructions that are similar to those generated by humans.

Our framework only assumes knowledge of the user’s pose and can be readily implemented on a mobile robot or body-worn device that provides natural language guidance as the user navigates, re-planning and issuing corrections as necessary. By maintaining probabilistic models of information content and ambiguity, our method is amenable to incorporating user feedback (e.g., utterances and gaze) to revise the information shared with the user (e.g., referencing landmarks based upon gaze). By using inverse reinforcement learning, our method can also adapt the content selection policy based on feedback to better learn user preferences.

#### 4.1 Compound Action Specifications

In order to bridge the gap between the low-level nature of the input paths and the natural language output, we encode paths using an intermediate logic-based formal language. Specifically, we use the Compound Action Specification (CAS) representation [39], which provides a formal abstraction of navigation commands for hybrid metric-topologic-semantic maps. The CAS language consists of five *actions* (i.e., Travel, Turn, Face, Verify, and Find), each

of which is associated with a number of *attributes* (e.g., *Travel.distance*, *Turn.direction*) that together define specific commands. We distinguish between CAS *structures*, which are instructions with the attributes left empty (e.g., *Turn(direction=None)*) thereby defining a class of instructions, and CAS *commands*, which correspond to instantiated instructions with the attributes set to particular values (e.g., *Turn(direction=Left)*). For each English instruction  $\Lambda^{(i)}$  in the dataset, we generate the corresponding CAS command  $c^{(i)}$  using the MARCO architecture [39]. For a complete description of the CAS language, see MacMahon et al. [39].

## 4.2 Content Selection

There are many ways in which one can compose a CAS representation of a desired path, both in terms of the type of information that is conveyed (e.g., referencing distances vs. physical landmarks), as well as the specific references to use (e.g., different objects provide candidate landmarks). Humans exhibit common preferences in terms of the type of information that is shared (e.g., favoring visible landmarks over distances) [59], yet the specific nature of this information depends upon the environment and the followers’ demographics [62, 27]. Our goal is to learn these preferences from a dataset of instructions generated by humans.

### 4.2.1 MDP with Inverse Reinforcement Learning

In similar fashion to Oswald et al. [45], we formulate the content selection problem as a Markov decision process (MDP) with the goal of then identifying an information selection policy that maximizes long-term cumulative reward consistent with human preferences (Fig. 2). However, this reward function is unknown a priori and is generally difficult to define. We assume that humans optimize a common reward function when composing instructions and employ inverse reinforcement learning (IRL) to learn a policy that mimics the preferences that humans exhibit based upon a set of human demonstrations.

An MDP is defined by the tuple  $(S, A, R, P, \gamma)$ , where  $S$  is a set of states,  $A$  is a set of actions,  $R(s, a, s') \in \mathbb{R}$  is the reward received when executing action  $a \in A$  in state  $s \in S$  and transitioning to state  $s' \in S$ ,  $P(s'|a, s)$  is the probability of transitioning from state  $s$  to state  $s'$  when executing action  $a$ , and  $\gamma \in (0, 1]$  is the discount factor. The policy  $\pi(a|s)$  corresponds to a distribution over actions given the current state. In the case of the route instruction domain, the state  $s$  defines the user’s pose and path in the context of the map of the environment. We represent the state in terms of 14 *context* features that express characteristics such as changes in orientation and position, the relative location of objects, and nearby environment features (e.g., different floor colors). We encode the state  $s$  as a 14-dimensional binary vector that indicates which context features are active for that state. In this way, the state space  $S$  is spanned by all possible instantiations of context features. Meanwhile, the action space corresponds to the space of different CAS structures (i.e., without instantiated attributes) that can be used to define the path.

We seek a policy  $\pi(a|s)$  that maximizes expected cumulative reward. However, the reward function that defines the value of particular characteristics of the instruction is unknown and difficult to define. For that reason, we frame the task as an inverse reinforcement learning problem using human-provided route instructions as demonstrations of

the optimal policy. Specifically, we learn a policy using the maximum entropy formulation of IRL [64], which models user actions as a distribution over paths parameterized as a log-linear model  $P(a; \theta) \propto e^{-\theta^\top \xi(a)}$ , where  $\xi(a)$  is a feature vector defined over actions and  $\theta$  is a parameter vector. We consider 9 instruction features (*properties*) that include features expressing the number of landmarks included in the instruction, the frame of reference that is used, and the complexity of the command. The feature vector  $\xi(a)$  then takes the form of a 9-dimensional binary vector. The Supplementary Material presents the full set of context and property features used to parameterize the states and actions. Maximum entropy IRL then solves for the distribution as

$$\begin{aligned} P(a; \theta^*) &= \arg \max_{\theta} P(a; \theta) \log P(a; \theta) \\ \text{s.t. } \xi_g &= \mathbb{E}[\xi(a)], \end{aligned} \quad (1)$$

where  $\xi_g$  denotes the features from the demonstrations and the expectation is taken over the action distribution. For further details regarding maximum entropy IRL, we refer the reader to Ziebart et al. [64].

The policy defines a distribution over CAS structure compositions (i.e., using the *Verify* action vs. the *Turn* action) in terms of their feature encoding. We perform inference over this policy to identify the maximum a posteriori property vector  $\xi(a^*) = \arg \max_{\xi} \pi$ . As there is no way to invert the feature mapping, we then match this vector  $\xi(a^*)$  to a database of CAS structures formed from our training set. Rather than choosing the nearest match, which may result in an inconsistent CAS structure, we retrieve the  $k_c$  nearest neighbors from the database using a weighted distance in terms of mutual information [45] that expresses the importance of different CAS features based upon the context. As several of these may be valid, we employ spectral clustering using the similarity of the CAS strings to identify a set of candidate CAS structures  $\mathcal{C}_s$ .

### 4.2.2 Sentence Planning

Given the set of candidate CAS structures  $\mathcal{C}_s$ , our method next chooses the attribute values such that the final CAS commands are both valid and not ambiguous. We express the likelihood that a command  $c \in \mathcal{C}_s$  is a valid representation of a particular path  $p$  defined on a map  $m$  as

$$P(c|p, m) = \frac{\delta(c|p, m)}{\sum_{j=1}^K \delta(c|\hat{p}_j, m)}. \quad (2)$$

The index  $j$  iterates over all possible paths that have the same starting pose as path  $p$ . We define  $\delta(c|p, m)$  as

$$\delta(c|p, m) = \begin{cases} 1 & \text{if } \eta(c) = \phi(c, p, m) \\ 0 & \text{otherwise} \end{cases}$$

where  $\eta(c)$  is the number of attributes defined in the command  $c$ , and  $\phi(c, p, m)$  is the number of attributes in the command  $c$  that are valid relative to the path  $p$  and map  $m$ .

For each candidate CAS structure  $c \in \mathcal{C}_s$ , we generate multiple CAS commands by iterating over the possible attributes values. We evaluate the correctness and ambiguity of each configuration according to Equation 2. A command is deemed valid if its likelihood is greater than a threshold  $P_t$ . Since the number of possible configurations for a structure increases exponentially with respect to the number of attributes, we assign attributes using greedy search. The iteration is constrained to use only objects and properties of

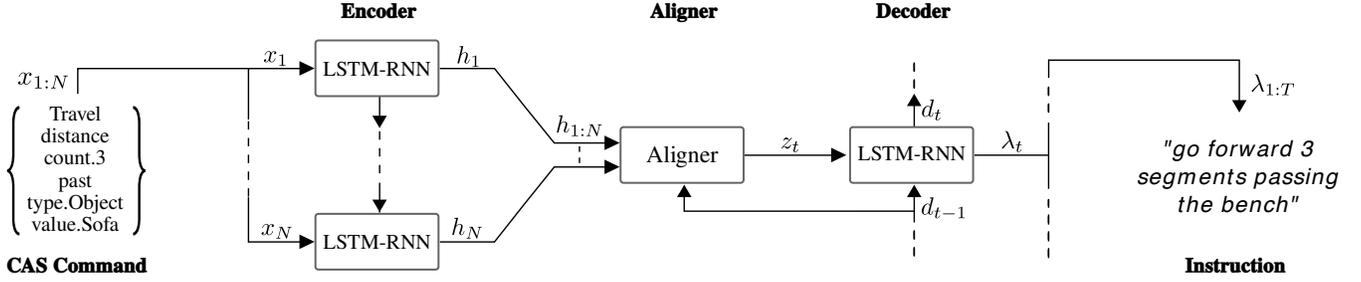


Figure 3: Our encoder-aligner-decoder model for surface realization.

the environment visible to the follower. The result is a set  $\mathcal{C}$  of valid CAS commands.

### 4.3 Surface Realization

Having identified a set of CAS commands suitable to the given path, our method then proceeds to generate the corresponding natural language route instruction. We formulate this problem as one of “translating” the instruction specification in the formal CAS language into its natural language equivalent.<sup>1</sup> We perform this translation using an encoder-aligner-decoder model (Fig. 3) that enables our framework to generate natural language instructions by learning from examples of human-generated instructions, without the need for specialized features, resources, or templates.

#### 4.3.1 Sequence-to-Sequence Model

We formulate the problem of generating natural language route instructions as inference over a probabilistic model  $P(\lambda_{1:T}|x_{1:N})$ , where  $\lambda_{1:T} = (\lambda_1, \lambda_2, \dots, \lambda_T)$  is the sequence of words in the instruction and  $x_{1:N} = (x_1, x_2, \dots, x_N)$  is the sequence of tokens in the CAS command. The CAS sequence includes a token for each action (e.g., *Turn*, *Travel*) and a set of tokens with the form *attribute.value* for each attribute-value pair; for example, *Turn(direction=Right)* is represented by the sequence (*Turn*, *direction.Right*). Generating an instruction sequence then corresponds to inference over this model.

$$\lambda_{1:T}^* = \arg \max_{\lambda_{1:T}} P(\lambda_{1:T}|x_{1:N}) \quad (3a)$$

$$= \arg \max_{\lambda_{1:T}} \prod_{t=1}^T P(\lambda_t | \lambda_{1:t-1}, x_{1:N}) \quad (3b)$$

We formulate this task as a sequence-to-sequence learning problem, whereby we use a recurrent neural network (RNN) to first encode the input CAS command

$$h_j = f(x_j, h_{j-1}) \quad (4a)$$

$$z_t = b(h_1, h_2, \dots, h_N), \quad (4b)$$

where  $h_j$  is the encoder hidden state for CAS token  $j$ , and  $f$  and  $b$  are nonlinear functions that we define later. An aligner computes the context vector  $z_t$  that encodes the language instruction at time  $t \in \{1, \dots, T\}$ . An RNN decodes the context vector  $z_t$  to arrive at the desired likelihood (Eqn. 3)

$$P(\lambda_t | \lambda_{1:t-1}, x_{1:N}) = g(d_{t-1}, z_t), \quad (5)$$

<sup>1</sup>Related work [40, 4, 41] similarly models the inverse task of language understanding as a machine translation problem.

where  $d_{t-1}$  is the decoder hidden state at time  $t-1$ , and  $g$  is a nonlinear function.

**Encoder** Our encoder (Fig. 3) takes as input the sequence of tokens in the CAS command  $x_{1:N}$ . We transform each token  $x_i$  into a  $k_e$ -dimensional binary vector using a word embedding representation [43]. We feed this sequence into an RNN encoder that employs LSTMs as the recurrent unit due to their ability to learn long-term dependencies among the instruction sequences without being prone to vanishing or exploding gradients. The LSTM-RNN encoder summarizes the relationship between elements of the CAS command and yields a sequence of hidden states  $h_{1:N} = (h_1, h_2, \dots, h_N)$ , where  $h_j$  encodes CAS words up to and including  $x_j$ . In practice, we reverse the input sequence before feeding it into the neural encoder, which has been demonstrated to improve performance for other neural translation tasks [53].

Our encoder is similar to that of Graves et al. [19]

$$\begin{pmatrix} i_j^e \\ f_j^e \\ o_j^e \\ g_j^e \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T^e \begin{pmatrix} x_j \\ h_{j-1} \end{pmatrix} \quad (6a)$$

$$c_j^e = f_j^e \odot c_{j-1}^e + i_j^e \odot g_j^e \quad (6b)$$

$$h_j = o_j^e \odot \tanh(c_j^e) \quad (6c)$$

where  $T^e$  is an affine transformation,  $\sigma$  is the logistic sigmoid that restricts its input to  $[0, 1]$ ,  $i_j^e$ ,  $f_j^e$ , and  $o_j^e$  are the input, output, and forget gates of the LSTM, respectively, and  $c_j^e$  is the memory cell activation vector (the Supplementary Material provides a visualization of an LSTM unit). The memory cell  $c_j^e$  summarizes the LSTM’s previous memory  $c_{j-1}^e$  and the current input, which are modulated by the forget and input gates, respectively.

**Aligner** Having encoded the input CAS command into a sequence of hidden annotations  $h_{1:N}$ , the decoder then seeks to generate a natural language instruction as a sequence of words. We employ an alignment mechanism [5] (Fig. 3) that permits our model to match and focus on particular elements of the CAS sequence that are salient to the current word in the output instruction. We compute the context vector as

$$z_t = \sum_j \alpha_{tj} h_j. \quad (7)$$

The weight  $\alpha_{tj}$  associated with the  $j$ -th hidden state is

$$\alpha_{tj} = \exp(\beta_{tj}) / \sum_k \exp(\beta_{tk}), \quad (8)$$

where the alignment term  $\beta_{tj} = f(d_{t-1}, h_j)$  expresses the degree to which the CAS element at position  $j$  and those around it match the output at time  $t$ . The term  $d_{t-1}$  represents the decoder hidden state at the previous time step. The alignment is modeled as a one-layer neural perceptron

$$\beta_{tj} = v^\top \tanh(Wd_{t-1} + Vh_j), \quad (9)$$

where  $v$ ,  $W$ , and  $V$  are learned parameters.

**Decoder** Our model employs an LSTM decoder (Fig. 3) that takes as input the context vector  $z_t$  and the decoder hidden state at the previous time step  $d_{t-1}$ , and outputs the conditional probability distribution  $P_{\lambda,t} = P(\lambda_t | \lambda_{1:t-1}, x_{1:N})$  over the next token as a deep output layer

$$\begin{pmatrix} i_t^d \\ f_t^d \\ o_t^d \\ g_t^d \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T^d \begin{pmatrix} d_{t-1} \\ z_t \end{pmatrix} \quad (10a)$$

$$c_t^d = f_t^d \odot c_{t-1}^d + i_t^d \odot g_t^d \quad (10b)$$

$$d_t = o_t^d \odot \tanh(c_t^d) \quad (10c)$$

$$q_t = L_0(L_d d_t + L_z z_t) \quad (10d)$$

$$P_{\lambda,t} = \text{softmax}(q_t) \quad (10e)$$

where  $L_0$ ,  $L_d$ , and  $L_z$  are parameters to be learned.

**Training** We train our encoder-aligner-decoder model so as to predict the natural language instruction  $\lambda_{1:T}^*$  for a given input sequence  $x_{1:N}$  using a training set of human-generated reference instructions. We use the negative log-likelihood of the reference instructions at each time step  $t$  as our loss function.

**Inference** Given a CAS command represented as a sequence of tokens  $x_{1:N}$ , we generate a route instruction as the sequence of maximum a posteriori words  $\lambda_{1:T}^*$  under our learned model (Eqn. 3). We use beam search to perform approximate inference, but have empirically found greedy search to often perform better.<sup>2</sup> For that reason, we generate candidates using both greedy and beam search.

### 4.3.2 Language Model

The inference procedure results in multiple candidate instructions for a given segment, and additional candidates may exist when there are multiple CAS specifications. We rank these candidate instructions using a language model (LM) trained on large amounts of English data. We formulate this LM as an LSTM-RNN [52] that assigns a perplexity score to each of the corresponding instructions.

Given the CAS specifications for a segmented path  $p = (p_1, p_2, \dots, p_M)$ , we generate the final instruction  $\Lambda$  by sequencing the  $M$  sentences  $\{\Lambda_1^*, \dots, \Lambda_M^*\}$  (i.e., one for each path segment)

$$\Lambda_i^* = \arg \min_{\Lambda_{ij}} L(\Lambda_{ij}), \quad (11)$$

where  $\Lambda_{ij}$  is the  $j$ -th candidate for the  $i$ -th segment and  $L(\Lambda_{ij})$  is the perplexity score assigned by the language model to the sentence  $\Lambda_{ij}$ .

<sup>2</sup>This phenomenon has been observed by others [3, 42], and we attribute it to training the model in a greedy fashion.

## 5. EXPERIMENTAL SETUP

### 5.1 Dataset

We train and evaluate our system using the publicly available SAIL route instruction dataset collected by MacMahon et al. [39]. We use the original data without correcting typos or wrong instructions (e.g., confusing “left” and “right”). The dataset consists of 3213 demonstrations arranged in 706 paragraphs produced by 6 instructors for 126 different paths throughout 3 virtual environments, where each demonstration provides a map-path-command tuple  $(m^{(i)}, p^{(i)}, \Lambda^{(i)})$ . We partition the dataset into separate training (70%), validation (10%), and test (20%) sets. We utilize command-instruction pairs  $(c^{(i)}, \Lambda^{(i)})$  from the training, validation and test sets for training, hyper-parameter tuning, and testing of our encoder-aligner-decoder model, respectively. We use path-command pairs  $(p^{(i)}, c^{(i)})$  from the training set for IRL and pairs from the validation set to tune the hyper-parameters of the content selection model. Finally, we use path-instruction pairs  $(p^{(i)}, \Lambda^{(i)})$  from the test set for evaluations with human participants.

#### 5.1.1 Data Augmentation

The SAIL dataset is significantly smaller than those typically used to train neural sequence-to-sequence models. In order to overcome this scarcity, we augment the original dataset using a set of rules. We use the augmented dataset to train the neural encoder-aligner-decoder model and the original dataset to train the content selection model. The Supplementary Material provides further details.

### 5.2 Implementation Details

We implement and test the proposed model using the following values for the system parameters:  $k_c = 100$ ,  $P_t = 0.99$ ,  $k_e = 128$ , and  $L_t = 95.0$ . The encoder-aligner-decoder consists of 2 layers for the encoder and decoder with 128 LSTM units per layer. The language model similarly includes a 2-layer recurrent neural network with 128 LSTM units per layer. The size of the CAS and natural (English) language vocabularies is 88 and 435, respectively, based upon the SAIL dataset. All parameters are chosen based on the performance on the validation set. We train our model using Adam [30] for optimization. At test time, we perform approximate inference using a beam width of two. Our method requires an average of 33s (16s without beam search) to generate instructions for a path of 9 movements on a laptop with an Intel Core i7 CPU (2.0 GHz) and 8 GB of RAM.

### 5.3 Automatic Evaluation

To the best of our knowledge, we are the first to use the SAIL dataset for the purposes of generating route instructions. Consequently, we evaluate our method by comparing our generated instructions with a reference set of human-generated commands from the SAIL dataset using the BLEU score (a 4-gram matching-based precision) [46]. For each command-instruction pair  $(c^{(i)}, \Lambda^{(i)})$  in the validation set, we first feed the command  $c^{(i)}$  into our model to obtain the generated instruction  $\Lambda^*$ , and secondly use  $\Lambda^{(i)}$  and  $\Lambda^*$  as the reference and hypothesis, respectively, to compute the BLEU score. We consider the average BLEU score at the individual sentence (macro-average precision) and the full-corpus (micro-average precision) levels.

## 5.4 Human Evaluation

The use of BLEU score indicates the similarity between instructions generated via our method and those produced by humans, but it does not provide a complete measure of the quality of the instructions (e.g., instructions that are correct but different in prose will receive a low BLEU score). In an effort to further evaluate the accuracy and usability of our method, we conducted a set of human evaluation experiments in which we asked 42 novice participants on Amazon Mechanical Turk (21 females and 21 males, ages 18–64, all native English speakers) to follow natural language route instructions chosen randomly from two equal-sized sets of instructions generated by our method and by humans for 50 distinct paths of various lengths. The paths and corresponding human-generated instructions were randomly sampled from the SAIL test set. Given a route instruction, human participants were asked to navigate to the best of their ability using their keyboard within a first-person, three-dimensional virtual world representative of the three environments from the SAIL corpus. The Supplementary Material provides an example of the participants’ field of view while following route instructions. After attempting to follow each instruction, each participant was given a survey comprised of eight questions, three requesting demographic information and five requesting feedback on their experience and the quality of the instructions that they followed. We collected data for a total of 441 experiments (227 using human annotated instructions and 214 using machine-generated instructions). The system randomly assigned the experiments to discourage the participants from learning the environments or becoming familiar with the style of a particular instructor. No participants experienced the same scenario with both human annotated and machine-generated instructions. The Supplementary Material provides further details regarding the experimental procedure.

## 6. RESULTS

We evaluate the performance of our architecture by scoring the generated instructions using the 4-gram BLEU score commonly used as an automatic evaluation mechanism for machine translation. Comparing to the human-generated instructions, our method achieves sentence- and corpus-level BLEU scores of 74.67% and 60.10%, respectively, on the validation set. On the test set, the method achieves sentence- and corpus-level BLEU scores of 72.18% and 45.39%, respectively. Figure 1 shows an example of a route instruction generated by our system for a given map and path.

### 6.1 Aligner Ablation

Our model employs an aligner in order to learn to focus on particular CAS tokens that are salient to words in the output instruction. We evaluate the contribution of the aligner by implementing and training an alternative model in which the last encoder hidden state is fed to the decoder.

	Full Model	No Aligner
sentence-level BLEU	<b>74.67</b>	74.40
corpus-level BLEU	<b>60.10</b>	57.40

Table 1: Aligner ablation results.

Table 1 compares the performance of the two models on

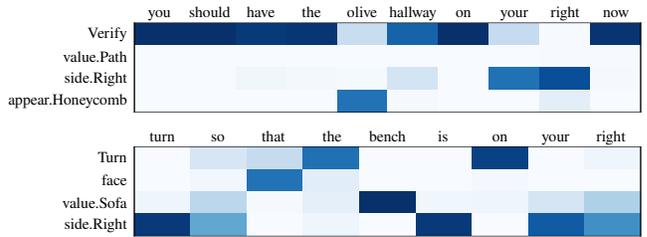


Figure 4: Alignment visualization for two pairs of CAS and natural language instructions.

the original validation set. The inclusion of an aligner results in a slight increase in the BLEU score of the generated instructions relative to the human-provided references, and is also useful as a means of visualizing the inner workings of our model (as discussed later). Additionally, we empirically find that the aligner improves our model’s ability to learn the association between CAS elements and words in the output, thereby yielding better instructions.

### 6.2 Language Model Ablation

Our method employs a language model to rank instructions generated for the different candidate CAS commands with different beam width settings. In practice, the language model, trained on large amounts of English data, helps to remove grammatically incorrect sentences produced by the sequence-to-sequence model, which is only trained on the smaller pairwise dataset. Table 2 presents two instruction candidates generated by our encoder-aligner-decoder model for two CAS commands. Our language model successfully assigns high perplexity to the incorrect instructions, with the chosen instruction being grammatically correct.

LM-score	Candidate
105.00	“so so a straight chair to your left”
27.65	“turn so that the chair is on your left side”
101.00	“keep going till the blue flor id on your left”
11.00	“move until you see blue floor to your right”

Table 2: Language model ablation outputs.

### 6.3 Aligner Visualization

Figure 4 presents heat maps that visualize the alignment between a CAS command that serves as input for surface realization (rows) and the generated instruction (columns) for two different scenarios drawn from the SAIL validation set. The visualizations demonstrate that our method learns to align elements of the formal CAS command with their corresponding words in the generated instruction. For example, the network learns the association between the honeycomb textured floor and its color (top); that “bench” refers to sofa objects (bottom); and that the phrase “you should have” indicates a verification action (top).

### 6.4 Human Evaluation

We evaluate the accuracy with which human participants followed the natural language instructions in terms of the Manhattan distance  $d$  between the desired destination (i.e., the last pose of the target path) and the participant’s location when s/he finished the scenario. Figure 5 compares the

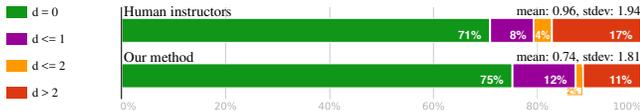


Figure 5: Participants’ distances from the goal.

accuracy of the participants’ paths when following human-generated instructions with those corresponding to instructions that our method produced. We report the fraction of times that participants finished within different distances from the goal.<sup>3</sup> The results demonstrate that participants reached the desired position 4% more often when following instructions generated using our method compared to the human instruction baseline. When they didn’t reach the destination, participants reached a location within one vertex away 8% more often given our instructions. Meanwhile our method yields a failure rate ( $d > 2$ ) that is 6% lower, though the difference is not statistically significant, as confirmed by a p-value of 0.131 relative to a standard significance level of 0.05. Of scenarios in which participants reached the destination, the total time required to interpret and follow our method’s instructions is 9.52s less than those generated by humans.

Figure 6 presents the participants’ responses to the survey questions that query their experience following the instructions. By using IRL to learn a content selection policy for constructing CAS structures, our method generates instructions that convey enough information to follow the command and were rated as providing too little information 15% less frequently than the human-generated baseline (Fig. 6(a)). Meanwhile, participants felt that our instructions were easier to follow (Fig. 6(b)) than the human-generated baselines (72% vs. 52% rated as “easy” or “very easy” for our method vs. the baseline). Participants were more confident in their ability to follow our method’s instructions (Fig. 6(c)) and felt that they had to backtrack less often (Fig. 6(d)). Meanwhile, both types of instructions were confused equally often as being machine-generated (Fig. 6(e)), however participants were less sure of who generated our instructions.

Figure 7 compares the paths that participants took when following our instructions with those that they took given the reference human-generated directions for one scenario. In particular, five participants failed to reach the destination when provided with the human-generated instruction. Two of the participants went directly to location 1, two participants navigated to location 2, and one participant went to location 2 before backtracking and taking a right to location 1. We attribute the failures to the ambiguity in the human-generated instruction that references “fish walled areas,” which could correspond to most of the hallways in this portion of the map (as denoted by the pink-colored lines). On the other hand, each of the five participants followed the intended path (shown in green) and reached the goal when following the instruction generated using our method.

## 7. CONCLUSION

We presented a model for natural language generation in the context of providing indoor route instructions that exploits a structured approach to produce unambiguous, easy to follow and grammatically correct human-like route in-

<sup>3</sup>We note that the  $d = 0$  accuracy for the human-generated instructions is consistent with that reported elsewhere [9].

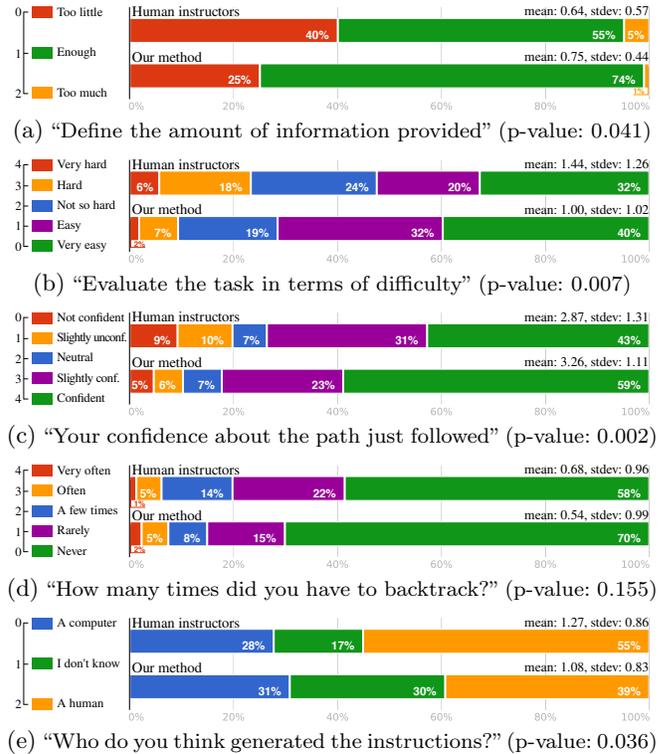


Figure 6: Participants’ survey response statistics.

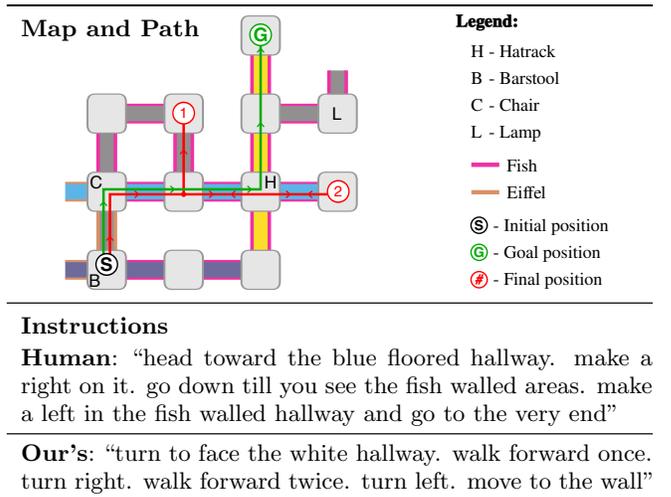


Figure 7: Paths followed by five participants given human-generated (red) and our (green) directions.

structions. Currently, our model generates natural language route instructions for the shortest path to the goal. However, there are situations in which a longer path may afford instructions that are more straightforward [48] or that increase the likelihood of reaching the destination [22]. Another interesting direction for future work would be to integrate a model of instruction followers [41] with our architecture in an effort to learn to generate instructions that are easier to follow. Such an approach would permit training the model in a reinforcement learning setting, directly optimizing over task performance.

## References

- [1] G. L. Allen. From knowledge to words to wayfinding: Issues in the production and comprehension of route directions. In *Proc. Int'l Conf. on Spatial Information Theory (COSIT)*, pages 363–372, 1997.
- [2] S. Andrist, E. Spannan, and B. Mutlu. Rhetorical robots: Making robots more effective speakers using linguistic cues of expertise. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, pages 341–348, March 2013.
- [3] G. Angeli, P. Liang, and D. Klein. A simple domain-independent probabilistic approach to generation. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pages 502–512, 2010.
- [4] Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Trans. Assoc. for Computational Linguistics*, 1:49–62, 2013.
- [5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.
- [6] R. Barzilay and M. Lapata. Collective content selection for concept-to-text generation. In *Proc. Human Language Technology Conf. and the Conf. on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 331–338, 2005.
- [7] R. Barzilay and L. Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. *arXiv preprint cs/0405039*, 2004.
- [8] D. L. Chen and R. J. Mooney. Learning to sportscast: a test of grounded language acquisition. In *Proc. Int'l Conf. on Machine Learning (ICML)*, pages 128–135, 2008.
- [9] D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI)*, pages 859–865, August 2011.
- [10] I. Chung, O. Propp, M. R. Walter, and T. M. Howard. On the performance of hierarchical distributed correspondence graphs for efficient symbol grounding of robot instructions. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)*, October 2015.
- [11] A. Correa, M. R. Walter, L. Fletcher, J. Glass, S. Teller, and R. Davis. Multimodal interaction with an autonomous forklift. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, pages 243–250, March 2010.
- [12] H. Cuayáhuitl, N. Dethlefs, L. Frommberger, K.-F. Richter, and J. Bateman. Generating adaptive route instructions using hierarchical reinforcement learning. In *Proc. Int'l Conf. on Spatial Cognition*, pages 319–334, 2010.
- [13] A. C. Curry, D. Gkatzia, and V. Rieser. Generating and evaluating landmark-based navigation instructions in virtual environments. In *Proc. Europ. Workshop on Natural Language Generation (ENLG)*, pages 90–94, September 2015.
- [14] R. Dale, S. Geldof, and J.-P. Prost. Using natural language generation in automatic route. *J. Research and Practice in Information Technology*, 37(1):89, 2005.
- [15] R. Deits, S. Tellex, P. Thaker, D. Simeonov, T. Kollar, and N. Roy. Clarifying commands with information-theoretic human-robot dialog. *J. of Human-Robot Interaction*, 2(2): 58–79, 2013.
- [16] M. T. Dzindolet, S. A. Peterson, R. A. Pomranky, L. G. Pierce, and H. P. Beck. The role of trust in automation reliance. *Int'l J. of Human-Computer Studies*, 58(6):697–718, June 2003.
- [17] T. Fong, C. Thorpe, and C. Baur. Collaboration, dialogue, and human-robot interaction. In *Int'l Symp. of Robotics Research (ISRR)*, November 2001.
- [18] R. Goedel and E. Olson. DART: A particle-based method for generating easy-to-follow directions. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 1213–1219, 2012.
- [19] A. Graves, M. Abdel-rahman, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proc. IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013.
- [20] V. Groom and C. Nass. Can robots be teammates?: Benchmarks in human-robot teams. *Interaction Studies*, 8(3):483–500, 2007.
- [21] B. J. Grosz. Collaborative systems. *AI Magazine*, 17(2): 67–85, 1996.
- [22] S. Haque, L. Kulik, and A. Klippel. Algorithms for reliable navigation and wayfinding. In *Proc. Int'l Conf. on Spatial Cognition (ICSC)*, pages 308–326, 2006.
- [23] K. Hayashi, D. Sakamoto, T. Kanda, M. Shiomi, S. Koizumi, H. Ishiguro, T. Ogasawara, and N. Hagita. Humanoid robots as a passive-social medium: A field experiment at a train station. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, pages 137–144, March 2007.
- [24] S. Hemachandra and M. Walter. Information-theoretic dialog to improve spatial-semantic representations. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)*, October 2015.
- [25] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [26] T. Howard, S. Tellex, and N. Roy. A natural language planner interface for mobile manipulators. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2014.
- [27] A. M. Hund, M. Schmettow, and M. L. Noordzij. The impact of culture and recipient perspective on direction giving in the service of wayfinding. *J. of Environmental Psychology*, 32(4): 327–336, 2012.
- [28] T. Kanda, T. Hirano, D. Eaton, and H. Ishiguro. Interactive robots as social partners and peer tutors for children: A field trial. *J. Human-Computer Interaction*, 19(1):61–84, June 2004.
- [29] J. Kim and R. J. Mooney. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proc. Int'l Conf. on Computational Linguistics*, pages 543–551, 2010.
- [30] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [31] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, pages 259–266, 2010.
- [32] I. Konstas and M. Lapata. Unsupervised concept-to-text generation with hypergraphs. In *Proc. Conf. of the North American Chapter of the Assoc. for Computational Linguistics (NAACL)*, pages 752–761, 2012.
- [33] L. Kunze, K. Kumar, and N. Hawes. Indirect object search based on qualitative spatial relations. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2014.

- [34] C. Landsiedel, R. D. Nijs, K. Käihnlitz, D. Wollherr, and M. Buss. Route description interpretation on automatically labeled robot maps. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2013.
- [35] P. Liang, M. I. Jordan, and D. Klein. Learning semantic correspondences with less supervision. In *Proc. Assoc. for Computational Linguistics (ACL)*, pages 91–99, 2009.
- [36] G. Look, B. Kottahachchi, R. Laddaga, and H. Shrobe. A location representation for generating descriptive walking directions. In *Proc. Int'l Conf. on Intelligent User Interfaces (IUI)*, pages 122–129, 2005.
- [37] G. W. K. Look. *Cognitively-inspired direction giving*. PhD thesis, Massachusetts Institute of Technology, 2008.
- [38] K. L. Lovelace, M. Hegarty, and D. R. Montello. Elements of good route directions in familiar and unfamiliar environments. In *Int'l Conf. on Spatial Information Theory*, pages 65–82, 1999.
- [39] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI)*, 2006.
- [40] C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, pages 251–258, March 2010.
- [41] H. Mei, M. Bansal, and M. R. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI)*, 2016.
- [42] H. Mei, M. Bansal, and M. R. Walter. What to talk about and how? Selective generation using LSTMs with coarse-to-fine alignment. In *Proc. Conf. of the North American Chapter of the Assoc. for Computational Linguistics (NAACL)*, 2016.
- [43] T. Mikolov and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 2013.
- [44] L. Y. Morales Saiki, S. Satake, T. Kanda, and N. Hagita. Modeling environments from a route perspective. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, pages 441–448, 2011.
- [45] S. Oswald, H. Kretschmar, W. Burgard, and C. Stachniss. Learning to give route directions from human demonstrations. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 3303–3308, 2014.
- [46] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proc. Assoc. for Computational Linguistics (ACL)*, pages 311–318, 2002.
- [47] V. Raman, C. Lignos, C. Finucane, K. C. T. Lee, M. Marcus, and H. Kress-Gazit. Sorry Dave, I'm afraid I can't do that: Explaining unachievable robot tasks using natural language. In *Proc. Robotics: Science and Systems (RSS)*, June 2013.
- [48] K.-F. Richter and M. Duckham. Simplest instructions: Finding easy-to-describe routes for navigation. In *Proc. Int'l Conf. on Geographic Information Science (GIScience)*, pages 274–289, 2008.
- [49] A. Sauppé and B. Mutlu. Effective task training strategies for human and robot instructors. *Autonomous Robots*, 39(3):313–329, October 2015.
- [50] A. St. Clair and M. Matarić. How robot verbal feedback can improve team performance in human-robot task collaborations. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, pages 213–220, March 2015.
- [51] K. Striegnitz, A. Denis, A. Gargett, K. Garoufi, A. Koller, and M. Theune. Report on the second challenge on generating instructions in virtual environments (GIVE-2.5). In *Proc. Europ. Workshop on Natural Language Generation (ENGL)*, pages 270–279, September 2011.
- [52] M. Sundermeyer, R. Schlüter, and H. Ney. LSTM neural networks for language modeling. In *Proc. of Interspeech*, pages 194–197, 2012.
- [53] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.
- [54] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI)*, pages 1507–1514, August 2011.
- [55] S. Tellex, P. Thaker, R. Deits, D. Simeonov, T. Kollar, and N. Roy. Toward information theoretic human-robot dialog. In *Proc. Robotics: Science and Systems (RSS)*, July 2012.
- [56] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy. Asking for help using inverse semantics. In *Proc. Robotics: Science and Systems (RSS)*, July 2014.
- [57] C. Torrey, S. R. Fussell, and S. Kiesler. How a robot should give advice. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, pages 275–282, March 2013.
- [58] M. A. Walker, O. Rambow, and M. Rogati. SPoT: A trainable sentence planner. In *Proc. Conf. of the North American Chapter of the Assoc. for Computational Linguistics (NAACL)*, June 2001.
- [59] D. Waller and Y. Lippa. Landmarks as beacons and associative cues: their role in route learning. *Memory & Cognition*, 35(5):910–924, 2007.
- [60] M. L. Walters, K. Dautenhahn, S. N. Woods, and K. L. Koay. Robotic etiquette: Results from user studies involving a fetch and carry task. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, pages 317–324, March 2007.
- [61] N. Wang, D. V. Pynadath, and S. G. Hill. Trust calibration within a human-robot team: Comparing automatically generated explanations. In *Proc. ACM/IEEE Int'l. Conf. on Human-Robot Interaction (HRI)*, March 2016.
- [62] S. L. Ward, N. Newcombe, and W. F. Overton. Turn left at the church, or three miles north: A study of direction giving and sex differences. *Environment Behavior*, 18(2):192–213, 1986.
- [63] Y. W. Wong and R. J. Mooney. Generation by inverting a semantic parser that uses statistical machine translation. In *Proc. Conf. of the North American Chapter of the Assoc. for Computational Linguistics – Human Language Technologies (NAACL HLT)*, pages 172–179, 2007.
- [64] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI)*, pages 1433–1438, July 2008.