

# Natural Language Communication with Robots

**Yonatan Bisk**

Information Sciences Institute  
Univ. of Southern California  
ybisk@isi.edu

**Deniz Yuret**

Koç University  
Computer Engineering  
dyuret@ku.edu.tr

**Daniel Marcu**

Information Sciences Institute &  
Department of Computer Science  
Univ. of Southern California  
marcu@isi.edu

## Abstract

We propose a framework for devising empirically testable algorithms for bridging the communication gap between humans and robots. We instantiate our framework in the context of a problem setting in which humans give instructions to robots using unrestricted natural language commands, with instruction sequences being subservient to building complex goal configurations in a blocks world. We show how one can collect meaningful training data and we propose three neural architectures for interpreting contextually grounded natural language commands. The proposed architectures allow us to correctly understand/ground the blocks that the robot should move when instructed by a human who uses unrestricted language. The architectures have more difficulty in correctly understanding/grounding the spatial relations required to place blocks correctly, especially when the blocks are not easily identifiable.

## 1 Motivation

Much of the progress in Natural Language Processing can be attributed to defining problems of broad interest (e.g. parsing and machine translation); collecting or creating publicly available corpora that encode meaningful ⟨input, output⟩ samples (e.g. Penn TreeBank and LDC Parallel Corpora); and devising simple, objective and computable evaluation metrics to automatically assess the performance of algorithms designed to solve the problems of interest, independent of the approach or technology used (e.g. ParseEval and Bleu).

As robots become increasingly ubiquitous, we need to learn to interact with them intelligently, in the same manner we interact with members of our own species. To make rapid progress in this area, we propose to use an intellectual framework that has the same ingredients that have transformed our field: appealing science problem definitions; publicly available datasets; and easily computable, objective evaluation metrics.

In this paper, we study the problem of Human-Robot Natural Language Communication in a setting inspired by a traditional AI problem – blocks world (Winograd, 1972). After reviewing previous work (Section 2), we propose a novel Human-Robot Communication Problem that is testable empirically (Section 3.1) and we describe the publicly available datasets (Section 3.2) and evaluation metric that we devised to support our research (Section 6). We then introduce a set of algorithms for solving our problem and we evaluate their performance both objectively and subjectively (Sections 4–8).

## 2 Previous work

Most research on Human-Robot Interaction (Klingspor et al., 1997; Thompson et al., 1993; Mavridis, 2015) bridges the gap between natural language commands and the physical world via a set of pre-defined templates characterized by a small vocabulary and grammar. Progress on language in this area has largely focused on grounding visual attributes (Kollar et al., 2013; Matuszek et al., 2014) and on learning spatial relations and actions for small vocabularies with hard-coded abstract concepts (Steels and Vogt, 1997; Roy, 2002;

Guadarrama et al., 2013). Language is sometimes grounded into simple actions (MacMahon et al., 2006; Yu and Siskind, 2013) but the data, while multimodal, is relatively formulaic, the vocabularies are small, and the grammar is constrained. Although robots have significantly increased their autonomy and ability to plan, that has not resulted, to date, in more flexible human-robot communication protocols that would enable robots to understand free-form language and/or acquire simple and complex concepts via human-robot interactions.

Recently the connection between less formulaic language and simple actions has been explored successfully in the context of simulated worlds (Branavan et al., 2009; Goldwasser and Roth, 2011; Branavan et al., 2011; Artzi and Zettlemoyer, 2013; Andreas and Klein, 2015) and videos (Malmaud et al., 2015; Venugopalan et al., 2015). However, to our knowledge, there is no body of work that focuses on understanding the relation between natural language and complex actions and goals or on explaining flexibly the actions taken by a robot in natural language utterances. As observed by Klingspor (1997), there is a big gap between the formulaic interactions that are typical of state-of-the-art human-robot communications and human-human interactions, which are more abstract.

### 3 A Framework for Human-Robot Natural Language Communication Research

#### 3.1 Problem Definition

**Problem-Solution Sequences.** In order to build models that understand the ambiguous and complex language used by people when communicating to solve a task, we adopt the Problem-Solution Sequence (PSS) framework proposed by Bisk et al. (2016). Problem-Solution Sequences provide high and low level descriptions of actions in service of a goal; more specifically, they are sequences of images that encode what a robot might see as it goes about accomplishing a goal. In this paper, we work with PSSs specific to a simple world that has blocks placed on a table and a robot that can visually inspect the table and manipulate the blocks on it.

Figure 1 shows four intermediate block configurations of a PSS the robot observes as it transforms the initial state block configuration (random) into the fi-

- 1 coca cola , hp , nvidia .
- 2 nvidia , to the right of hp
- 3 place the nvidia block east of the hp block .
- 4 move the nvidia block to the right of the hp block
- 5 place the nvidia block to the east of the hp block .
- 6 move the nvidia block directly to the right of the hp block .
- 7 move the nvidia block just to the right of the hp block in line with the mercedes block .
- 8 put the nvidia block on the right end of the row of blocks that includes the coca cola and hp blocks .
- 9 put the nvidia block on the same row as the coca cola block , in the first open space to the right of the coca cola block .

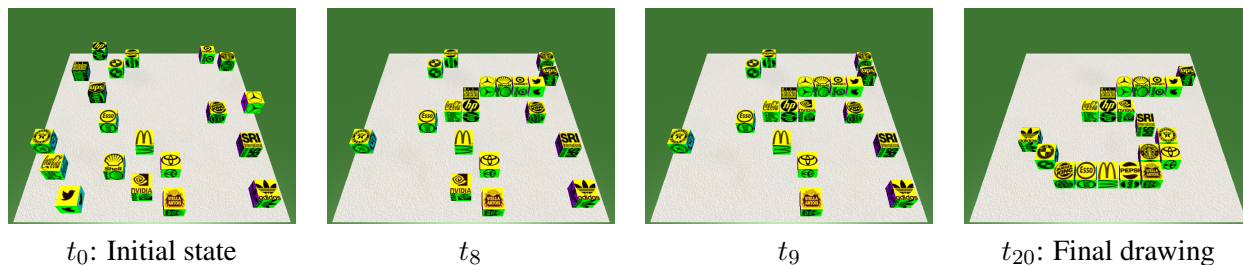
**Table 1:** Examples of the type of natural language instructions seen in our corpus that verbalize the action needed to transition from  $t_8$  to  $t_9$  in Figure 1.

nal one (the number five). The PSS makes explicit the natural language instructions that a human may give to a robot in order to transform the configuration in an  $\text{Image}_i$  into the configuration in an  $\text{Image}_j$  - the two configurations may correspond to one robot action (for adjacent states in the sequence) or to a sequence of robot actions (for non-adjacent states). To account for language variance, each simple action or sequence of actions is associated with a set of alternative natural language instructions. For example, nine descriptions of the same action ( $t_8 \rightarrow t_9$ ) from Figure 1 are shown in Table 1.

We see some structural similarity between the utterances, but they require different amounts of inference to understand, use different (potentially synonymous) language, and choose different blocks as contextual anchors for proper interpretation. Despite this, they each describe the action with equal precision. The natural language instructions encode implicitly partial or full descriptions of the world (“in line with” or “first open space”).

#### Simple Instruction/Command Understanding.

The problem definition we focus on in this paper is that of simple instruction/command understanding: given a state of the world,  $\text{Image}_i$ , and a human-like natural language command,  $C$ , we would like to infer the target world,  $\text{Image}_{i+1}$ , that a robot should construct if it understood  $C$ . If we assume, for simplicity, that only one block can be moved at a time, command understanding has a straightforward se-



**Figure 1:** Above are four states in a 20-action sequence for drawing the digit 5. The world state can be encoded for the learner as IDs and locations or raw images. Annotations are provided between every adjacent state (See Table 1) or between sequences (e.g.  $t_8 \rightarrow t_{20}$ : *Create a four-line diagonal which moves to the southeast. Starting with Heineken ...*) to describe multi-action plans.

mantics: understanding a command amounts to inferring that the block at location  $(x, y, z)_S$  needs to be moved and the location  $(x, y, z)_T$  where the block needs to be moved. The rest of the blocks are not affected by the move.

### 3.2 Data

We follow Bisk et al. (2016)’s methodology and collect PSSs specific to both goal oriented and random actions.<sup>1</sup> Our discussion focuses primarily on the goal oriented data, wherein blocks are used to draw configurations/scenes that look like the digits zero through nine. To create these abstract drawings in a diverse and natural manner, configurations are derived from actual hand-written digits in the MNIST corpus (LeCun et al., 1998). These digits provide an easily recognizable target goal when arranging blocks. To create a sequence of actions that draws out these digits, the MNIST images were sharpened and down-sampled until each had at most 20 active pixels which could be replaced with blocks. The blocks were either decorated with brands (as shown above) or with the numbers one through twenty.

These block configurations were placed into a virtual world and scrambled until the board’s initial state was unrecognizable. This was achieved by randomly relocating adjacent blocks to new locations in the world. Once every block had been placed at a new location, the world appears random (for example, Image<sub>0</sub> in Figure 1), but when these random actions are played in reverse, the sequence of moves recreates the digit in a deliberate and ordered manner. Each of these actions are then shown to Amazon Mechanical Turkers. To ensure that our robot

learns to understand human-like commands, turkers were asked to provide instructions they would give to another person in order to transform a block configuration corresponding to a first image ( $t_i$ ) into a second block configuration corresponding to an image ( $t_{i+1}$ ). Each image pair was presented to three turkers, each of whom had to provide three different instructions for achieving the same goal.

A total of 100 digits sequences were annotated (10 drawings for every digits). The sequences were split so that half were decorated with numbers and half with logos. The sides of every block have a different color and a logo or number is overlaid on every side. Eight sequences for every digit are included in Training, one for development and two for testing. Overall, the training data has 11,871 commands, while the development and test corpora each have 1,719 and 3,177 commands, respectively.

For each learning example, we thus have access to an input that consists of an Image <sub>$i$</sub>  (what the robot sees), the  $(x, y, z)$  coordinates of each block in Image <sub>$i$</sub>  (a discrete representation of the world corresponding to the image), and a natural language command that a robot needs to understand in order to operate on the world in Image <sub>$i$</sub> . The output consists of an Image <sub>$i+1$</sub>  (what the robot should build) and the  $(x, y, z)$  coordinates of each block in Image <sub>$i+1$</sub> .

The training/development/test sections of the data contain  $\sim 177\text{K}/31\text{K}/48\text{K}$  tokens for the decorated blocks. The overall lexical type and token counts for our data are presented in Table 2. To compute statistics all text was lower-cased and tokenized using Stanford’s CoreNLP (Manning et al., 2014). For the MNIST configurations, digits 0-9 are present in the test data as drawn with both logos and numbered blocks. In contrast, only half the digits appear with

<sup>1</sup>All data (both single actions and sequences) and links to code are available at <http://nlg.isi.edu/language-grounding/>

|       | MNIST        |        | Blank       |        |
|-------|--------------|--------|-------------|--------|
|       | Types        | Tokens | Types       | Tokens |
| Train | 1,506        | 177K   | 961         | 58K    |
| Dev   | 583          | 31K    | 444         | 8K     |
| Test  | 645          | 48K    | 575         | 17K    |
| Total | 1,359 / 257K |        | 1,172 / 84K |        |

**Table 2:** Type and token counts for the Logo and Number decorated block data sets (left) and the Blank blocks (right).

a given decoration in the development data.

Perhaps because annotators were not constrained or told they were giving instructions to a robot, the breadth of constructions and variance in command length is substantial. For example, the length of the commands varies wildly. Table 3 shows the number of training commands in a given length range. Some commands span multiple sentences. The average command length is 15 tokens with a standard deviation of 8.

The free form nature of the language and task allows for both utilitarian descriptions as well as more flowery instructions which utilize the full three-dimensional world:

*take a plunger . plunge the top of the adidas block . lift it into the air , and place it directly to the left of the burgerking block , making sure the right edge of the adidas block and the left edge of the burgerking block are touching . remove the plunger .*

**Random Blank Blocks.** In both Table 2 and 3 we also present statistics on a second much more challenging dataset of blank blocks used to build random configurations. For this data, blocks were randomly placed alongside each other, on top of one another, or randomly scattered in the space. Additionally, these blocks have no identifying labels so they are more difficult to describe, making the grounding problem difficult. This leads to more interesting language with more spatial cues and counting (e.g. *third block from the top...*). This manifests as much longer descriptions averaging 23.5 words with a standard deviation of 9 words. We see this length bias in Table 3. Finally, this data also presents the challenge of stack creation in the third-dimension. As capturing the language and phenomena of this data is largely out of the scope of our current work, we present baseline results to demonstrate its dif-

| Command Length ( $l$ ) | # of Commands |        |
|------------------------|---------------|--------|
|                        | MNIST         | Random |
| $1 \leq l \leq 5$      | 81            | 0      |
| $5 < l \leq 10$        | 3,817         | 61     |
| $10 < l \leq 20$       | 5,752         | 995    |
| $20 < l \leq 40$       | 2,028         | 1,329  |
| $40 < l \leq 80$       | 192           | 107    |

**Table 3:** A breakdown of the number of commands in the training data by the number of tokens per sentence.

iculty in the hope of motivating future research. The complete blank blocks corpus consists of 2,493 training commands, 360 for development and 720 for testing or a total of  $\sim 58\text{K}/8\text{K}/17\text{K}$  tokens.

## 4 Learning Problems of Interest

Implicitly encoded in our data are three tasks with varying amounts of abstraction and context-specific language: Entity grounding, Spatial Relation grounding, and Planning.

**Entity Grounding.** As one would expect based on Gricean maxims, there are many ways an object might be referred to in everyday speech. These are context specific and depend on the perceived ambiguity of a scene. For example, the decoration of blocks with logos allows for easy indexing (“nvidia block”) which uniquely identifies the referent. If a human feels the brand is not sufficiently recognizable they may choose to describe *Texaco* as “the star block”, or *Mercedes* as “three lines in a circle”. In these cases, the speaker is appealing to more basic geometric knowledge in lieu of brand recognition.

The introduction of numbered blocks complicates the grounding as many actions also contain measures of how far to move a block:

*put block 10 four spaces below block 9 .*

In this case, the user has decided to denote the block IDs with the numerals 0-9, but distances by spelling out the number. As is to be expected, most strategies do not hold across users, and an individual user may be very inconsistent:

*move block seven two spaces to the right of block 6*

This inconsistency, while difficult for a learner, introduces no actual ambiguity for a fluent speaker.

Finally, blank block descriptions use lots of spacial references the involve often complicated recursive structure:

*Move the block that is to the [left of [ the block closest to [ the right side of the table ]]] so that it is on top of the block that is at the [ top of the [ group of blocks [ closest to the [ left side of the table ]]]].*

It follows that an important subtask for our models/algorithms is to correctly ground the entities referenced in naturally occurring commands. In general this may require a thorough understanding of syntax and scoping.

**Spatial Relation Grounding.** Another subtask is that of understanding spacial relations. Again, we are presented with lots of linguistic ambiguity which can be resolved by the shared context and references of the speaker and listener. For example, the first command in Table 1 is simply a list of three brands:

*coca cola, hp, nvidia.*

This statement on its own is meaningless, but in the context of an image that contains the first two brands in sequence from left to right, the list implies that the final logo should be appended to the existing line.

Naturally occurring commands also assume basic knowledge of physics. The final description in Table 1, asks that we place a block in the “first open space to the right ...”. Implicit in this statement is a shared understanding that two blocks cannot occupy the same space. This implies that a human or robot knows to search for the open space, which may be arbitrarily far away. Knowledge of physics or basic geometric shapes appears to be common in the instructions and injecting this knowledge into our models may be helpful.

*use block 2 as a bridge to complete the diagonal line formed by blocks 17, 8, 6, 4 and 1.*

Finally, when analyzing the data, we found that command givers would often create an ad hoc grid to assist in specifying where a block should be placed. This was particularly common when placing the first block ( $t_0 \rightarrow t_1$ ) to start the drawing. This initial block may need to be placed in a position that is not near any existing blocks. Common solutions include specifying midpoints between blocks to center a conversation:

*place the adidas block in the column between the columns that contain the mercedes and esso blocks, but two block spaces below either.*

Often the referenced blocks may be very far apart but appear opposite one another and equidistant from a useful reference point in space.

**Plan Recognition.** The third problem of interest is plan recognition. The annotation of sequences of actions shows that natural commands are also used in a manner that assumes the ability to plan and execute individual and complex actions:

*slide the adidas block 2 blocks straight up. then slide it 6 block spaces to the left.*

The models we introduce in this paper have difficulty dealing with these kind of commands.

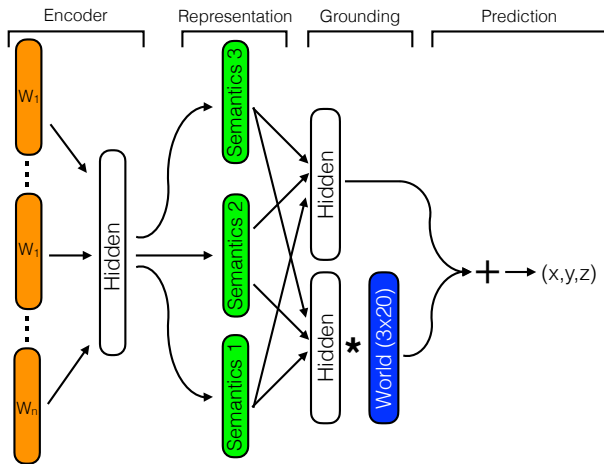
## 5 Models

In order to correctly interpret commands in context, we need models that ground entities and understand spatial relations, shapes, and the compositionality of language. This is a large and fertile space for exploration. In contrast with previous work which attempts to produce deep semantic interpretations of commands (Kim and Mooney, 2012; Dukes, 2014), in this paper we explore the degree to which we can solve our communication problem using semantic free models. We are quick to note though that our framework can also assess the performance of semantic-heavy approaches.

We outline here three basic neural models that provide a set of reasonable baselines for other researchers interested in solving this problem. Each approach assumes less knowledge injection than the previous. As discussed in Section 3.1, in all three models, the eventual output is a tuple specifying where to find the block to move and where to move it:  $(x, y, z)_S$  and  $(x, y, z)_T$ . For each model presented below we will try both simple feed-forward and recurrent neural network architectures for encoding the input utterance.

### 5.1 Model Architecture

The goal of our models is to convert an utterance and world state into a location prediction in the world. We tackle this problem by breaking the problem into four steps: Encoding, Representation, Grounding, Prediction. Components of this pipeline



**Figure 2:** Our models all follow the above architecture. 1-Hot word vectors (orange) are fed as input to a Feed-Forward or Recurrent Neural Network for encoding. A semantic representation is extracted (green), which in conjunction with knowledge of the world (blue) is grounded to predict an action.

can be trained independently (Sections 5.2 and 5.3) or jointly as a single End-to-End model (Section 5.4). This division of labor also allows for differing amounts of human intervention both during training and in the interpretation of actions and bears some resemblance to (Andreas et al., 2016). Specifically, we will first present results where the model predicts a fixed semantic interpretation of actions which are easily human interpretable (Encoder + Representation). In this setting, the experimenter/human then must convert the semantics to actions in the world. Second, we remove the human interpreter and train a model for Grounding and Predicting from our semantic representation. Finally, we maintain our architecture but remove the human entirely, forcing the model to both converge to and interpret its own internal semantic representation.

The model architecture, regardless of how it is trained, at least implicitly, encodes our beliefs about the best way to solve the learning problem: performing single actions requires identifying anchors in the world that can be used as spacial referents from which a target location can be offset.

## 5.2 Discrete Predictions of a Fixed Semantics

Our first model assumes a setup with very simple semantics. Despite all blocks existing in a real-valued world, we will assume that a final location is parameterized by knowledge of a reference block and the

direction from the reference to the target position.

*Move the Adidas block to the right of the BMW.*

For example, in the simple command above, we can distill three pieces of relevant information:

Source: Adidas  
Reference: BMW  
Direction: right (*east*)

By assuming a grid world, BMW can be converted to its location in the world  $(x, y, z)_{BMW}$ , which we shift *east* by changing the  $y$  component to yield:  $(x, y + \delta, z)$ . In practice we define a set of nine relative positions:

|                       |                      |                       |
|-----------------------|----------------------|-----------------------|
| NW<br>$(x-1, y-1, z)$ | N<br>$(x, y-1, z)$   | NE<br>$(x+1, y-1, z)$ |
| NW<br>$(x-1, y, z)$   | TOP<br>$(x, y, z+1)$ | E<br>$(x+1, y, z)$    |
| SW<br>$(x-1, y+1, z)$ | S<br>$(x, y+1, z)$   | SE<br>$(x+1, y+1, z)$ |

First our model produces an encoding of the sentence. We present two approaches:

**Feed-Forward Neural Network (FFN):** This model produces a sentence encoding by concatenating one-hot word vectors as input to a hidden layer. We pad sentences so all inputs are the same length.

**Recurrent Neural Network (RNN):** In contrast, the RNN encoder consumes the full sentence, each word passing through a hidden layer one at a time, before returning a final representation.

Additionally, in both encoding approaches, words which only occur a single time during training are replaced with an UNK token.

We use a single hidden layer architecture with a softmax for prediction and train with cross entropy loss. We train a separate model for each prediction (The Encoder and Representation stages of Figure 2). Once three versions of the model have been used to predict the Source, Reference and Direction, this triple is used to compute both the source  $(x, y, z)_S$  and target  $(x, y, z)_T$  locations. The former is computed via a simple look-up table, while the latter amends the reference look-up with the appropriate offset from the aforementioned grid.

When the model predicts a reference block which is not on the board (not all configurations use all 20 blocks) we set the reference location to the center of the board and then apply the relative position transformation to this hallucinated block location.

### 5.3 Continuous Valued Predictions From a Semantic Triple

At this point, we have constructed a model for predicting a specific semantic triple, but rely on the human to convert its output to physical locations given the current state of the world. To address this, we train a simple architecture which is shown the world and automatically learn direction offsets (the Grounding and Prediction stages of Figure 2).

The model takes knowledge of the Source, Reference and Direction and passes them to two hidden layers. One is multiplied by the world (a  $3 \times 20$  matrix of coordinates) and then both are summed to produce a final  $(x, y, z)$  prediction. The world matrix columns are the locations of each block, with a fixed ordering. If any block is missing from the configuration the matrix is padded with  $[-1, -1, -1]$ . This component of the network is then trained with a mean-square error regression loss.

Running this model on the predicted representation of Section 5.2 creates a simple pipeline from Sentence and World representations to location predictions, with no human intervention. We are particularly interested in this model’s performance because its intermediary representation is forced to conform to the simple, interpretable semantics we chose for this domain and task.

### 5.4 End-to-End Model

Finally, we present a single model which takes as input the sentence and world as before and predicts either the location of the block to move or its final location. This corresponds to training the neural architecture (Figure 2). We train the model twice, once to predict the source location  $(x, y, z)_S$  and a second time to predict the target location  $(x, y, z)_T$ . While the model architecture implicitly assumes the presence of an internal semantics we do not train it directly, but rather rely on the model to discover one based solely on its prediction error in the world. This approach is likely to allow for easier future extensions that encode finer-grained direction information and scaling (see analysis in Section 8).

## 6 Evaluation Metric

In our formulation, understanding a command  $C$  in the context of a world configuration  $\text{Image}_i$  amounts

to inferring the block  $(x, y, z)_S$  that needs to be moved and the target location  $(x, y, z)_T$  where the block needs to be moved to. Given that we control the manner in which the data is generated, we always have access to the gold interpretation of command  $C$ . Therefore, it is trivial to measure the performance of various command understanding algorithms by tracking two metrics: (i) we measure our ability to identify the block to be moved (and reference/direction information when available) using standard accuracy figures; (ii) and we measure our ability to select and place the block to be moved by measuring the distance between our predicted locations and the gold locations.

The first evaluation is presented in Table 4 under the S, R, and D columns. The distance errors are computed in terms of block lengths and we present the mean and median errors both for the source block’s initial and final location.

## 7 Baselines and Human Performance

Since the gold annotations make explicit only the block to move and its target location, the Fixed Semantics models do not have gold training data for predicting the reference block used for anchoring or the direction to offset. To remedy this, we use a simple string matching heuristic that chooses a reference block during training and that computes a “gold” direction from its location. The reference is chosen as the closest block mentioned in the sentence, other than the source.

**Oracle.** To evaluate the strength of this heuristic, we perform an oracle evaluation (Table 4): we assume perfect knowledge of the source block that is moved; we apply our string matching heuristic to choose a reference block; and then assume perfect knowledge of the quadrant in which we place the block that is moved. For the blank blocks, our string matching heuristic fails, so we simply use the closest block to the target location to the reference location. This, unsurprisingly, leads to higher error.

**Human Performance.** We randomly sampled 50 utterances from each dataset to evaluate human performance. The participants in our experiment were not affiliated with the project and were not provided any guidance about the task; for example, they were

|     |                        | MNIST Patterns with labeled blocks |             |             |             |     |     | Random Patterns with blank blocks |             |             |             |             |     |     |     |
|-----|------------------------|------------------------------------|-------------|-------------|-------------|-----|-----|-----------------------------------|-------------|-------------|-------------|-------------|-----|-----|-----|
|     |                        | Source                             |             | Target      |             | S   | R   | D                                 | Source      |             | Target      |             | S   | R   | D   |
|     |                        | Med                                | Mean        | Med         | Mean        |     |     |                                   | Med         | Mean        | Med         | Mean        |     |     |     |
|     | Human Performance      | 0.00                               | 0.00        | 0.21        | 0.53        | 100 |     |                                   | 0.00        | 0.30        | 0.37        | 1.39        | 93  |     |     |
|     | Oracle                 | –                                  | –           | 0.00        | 0.45        | 100 | 100 | 100                               | –           | –           | 1.00        | 1.09        | 100 | 100 | 100 |
| FFN | Discrete Predictions   | <b>0.00</b>                        | 0.49        | <b>1.09</b> | 2.17        | 93  | 69  | 63                                | 5.28        | 5.09        | 5.51        | 5.46        | 9   | 15  | 32  |
|     | Continuous Predictions | 0.49                               | 1.00        | 1.59        | 2.42        |     |     |                                   | 4.25        | 4.04        | 3.86        | <b>3.93</b> |     |     |     |
|     | End-to-End             | 0.02                               | <b>0.38</b> | 1.14        | <b>1.81</b> |     |     |                                   | <b>3.45</b> | <b>3.52</b> | <b>3.60</b> | 3.94        |     |     |     |
| RNN | Discrete Predictions   | <b>0.00</b>                        | <b>0.14</b> | <b>0.00</b> | <b>0.98</b> | 98  | 92  | 78                                | 5.29        | 5.00        | 5.51        | 5.57        | 10  | 7   | 46  |
|     | Continuous Predictions | 0.47                               | 0.64        | 1.23        | 1.60        |     |     |                                   | 4.16        | 4.05        | 3.71        | 3.87        |     |     |     |
|     | End-to-End             | 0.03                               | 0.19        | 0.53        | 1.05        |     |     |                                   | <b>3.29</b> | <b>3.47</b> | <b>3.60</b> | <b>3.70</b> |     |     |     |
|     | Center Baseline        | –                                  | –           | 3.46        | 3.43        | 100 |     |                                   | –           | –           | 4.09        | 4.06        | 100 |     |     |
|     | Random Baseline        | 6.37                               | 6.49        | 6.12        | 6.21        | 5   | 5   | 11                                | 4.90        | 4.97        | 5.51        | 5.44        | 10  | 11  | 12  |

**Table 4:** Model error when trained on only the subset of the data with decorated blocks or blank blocks. Where appropriate S, R, and D are the model’s predictive accuracy at identifying the Source, Reference and Direction. All models are evaluated on the Median and Mean prediction error the source block and its final target location. Distances are presented in block-lengths.

not told about the high-level goal of drawing a number. Despite this, Table 4 shows human performance is very similar to Oracle performance. Although humans did not place blocks in line “perfectly”, they were comparable to or outperformed the oracle.

**Baselines.** Finally, Table 4 also shows the results obtained by two baseline models. One (Center) has perfect knowledge of the source block to move, but always places it in the center of the table. The second baseline (Random), chooses random values for the source, reference, and direction. As expected, the performance of these baselines is abysmal.

## 8 Results

The results in Table 4 show that there is a massive difference in performance between block configurations that use blocks marked with identifiers (logos and digits) and those without. When the blocks are marked with clearly identifiable logos, all models outperform our baselines by a wide margin. However, when blocks are blank the situation is flipped.

The results in Table 4 also highlight a noticeable gap in performance between the simplest Discrete model and the two location predicting models. The comparable performance of the Continuous and End-to-End models on labeled blocks seems to imply that the End-to-End model is capturing/discovering similar anchoring information without being explicitly told to do so. On the blank block

data, the End-to-End model performs best by learning its own more appropriate representation.

**Parameters.** Where appropriate, we used 256 unit hidden-layers, 0.5 dropout, and the Adam optimizer with a learning rate of 0.001. With the exception of the FFN Discrete Predictions model, SGD parameter grid-search did not yield an improvement.

### 8.1 Subjective Error Analysis

In Table 5, we collected 50 of our models worst errors on the decorated blocks data and categorized them into five classes of error. Eliminating most of these errors require more knowledge or a richer representation than currently afforded by our simple semantic triples. This is often due to the use of multiple reference blocks, but grammatical ambiguity and a knowledge of some basic geometric primitives also account for many of the mistakes.

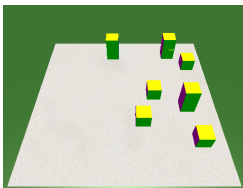
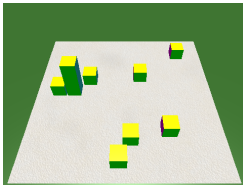
### 8.2 Future Work on Blank Blocks

One of the most jarring results we present is the the clear performance gap between easily grounded blocks (MNIST data) and the Blank blocks (Random) which require a much richer understanding of the world. We do not believe this is due to additional complexity in the types of relations present in the data, but rather the difficulty in grounding the references. When analyzing the data we see that much of the data still follow a very simple (Source, Ref-



| Error Type              | Count | Example   |
|-------------------------|-------|---|
| Multi-Relation Actions  | 20    | Place block 20 parallel with the 8 block and slightly to the right of the 6 block. Place block 15 on the same vertical column as blocks 16 and 17, and two rows above blocks 11 and 3.  |
| Geometric Understanding | 10    | Continue the diagonal row of 20, 19 and 15 downward with 13. Put block 12 in the column between the columns with blocks 4 and 13, and on the same row as the lowest block on the board. |
| Grammatical Ambiguity   | 10    | 19 moved from behind the 8 to under the 18th block. Burger King tile should be directly above the Coca Cola tile. Move Coca Cola.   |
| Grounding Names         | 5     | Put the block that looks like a taurus symbol just above the bird.  |
| Understanding Distance  | 5     | move the texaco block 5 block lengths above the BMW block   |

**Table 5:** We performed a subjective error analysis of the results of our Fixed Semantics model using the RNN encoder. Example sentences and the frequency of each type of error are reported above from the worst 50 errors on the development data.

| Scene   | Utterance   |
|---|---|
|    | Move the block that is currently located closest to the top left corner to the bottom left of the table, slightly higher than the block in the bottom right corner. |
| <b>Error:</b>   | 7.29 Block lengths  |
|  | Move the block closest to the top left corner so it is above half a block length to the right of the blocks near the lower left corner of the table.                |
| <b>Error:</b>   | 0.94 Block lengths  |

**Table 6:** Above are two commands and the worlds they apply to. Below we see the prediction error of our best model.

erence, Direction) paradigm, but automatically extracting that semantics is now more difficult and the purview for future work with scene understanding.

To remove the possibility that this performance difference is due to sparsity, we down-sampled the training data from the decorated blocks to match that of the blank ones. We found the development errors grew (Average 0.27 and 1.35 on source and target, respectively) but were still substantially lower than those observed with blank block data.

Because extracting the semantics for training is so difficult, a particularly nice result is that while the End-to-End model was slightly weaker than the others on the MNIST based data, it actually performs best in this domain, where we cannot provide an ex-

plicit training signal for the representation.

The nature of the language in the blank blocks differs quite dramatically due to this grounding difficulty. Table 6 shows the two sentences we perform best (and worst) on in the development data and that make use of a reference and direction.

## 9 Conclusion

We showed how human-robot communication can be attacked within an empirical framework that supports alternative models to be evaluated and compared using objective metrics. We introduced a set of simple algorithms for human-robot, in-context command/instruction understanding that should serve as strong baselines for future research in this field. The datasets present unique and important challenges for NLU, in which the interpretation of the language has varying amounts of dependence on the world in which it is uttered. The datasets we created in support of this work are made publicly available and should support the development of increasingly sophisticated models and algorithms for solving the problem defined in this paper, as well as additional problems that concern human-robot communication.

## Acknowledgments

This work was supported by Contract W911NF-15-1-0543 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO).

## References

- Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1165–1174, Lisbon, Portugal, September. Association for Computational Linguistics.
- Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*.
- Yoav Artzi and Luke S Zettlemoyer. 2013. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics*, pages 49–62.
- Yonatan Bisk, Daniel Marcu, and William Wong. 2016. Towards a dataset for human computer communication via grounded language acquisition. In *Proceedings of the AAAI'16 Workshop on Symbiotic Cognitive Systems*.
- SRK Branavan, Harr Chen, Luke S Zettlemoyer, and Regina Barzilay. 2009. Reinforcement Learning for Mapping Instructions to Actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore, August.
- S R K Branavan, David Silver, and Regina Barzilay. 2011. Learning to Win by Reading Manuals in a Monte-Carlo Framework. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 268–277, Portland, Oregon, USA, June.
- Kais Dukes. 2014. Semeval-2014 task 6: Supervised semantic parsing of robotic spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 45–53, Dublin, Ireland, August.
- Dan Goldwasser and Dan Roth. 2011. Learning From Natural Instructions. In *Twenty-Second International Joint Conferences on Artificial Intelligence*.
- Sergio Guadarrama, Lorenzo Riano, Dave Golland, Daniel Göhring, Jia Yangqing, Dan Klein, Pieter Abbeel, and Trevor Darrell. 2013. Grounding Spatial Relations for Human-Robot Interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1640–1647.
- Joohyun Kim and Raymond Mooney. 2012. Unsupervised pcf induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 433–444, Jeju Island, Korea, July.
- Volker Klingspor, John Demiris, and Michael Kaiser. 1997. Human-Robot-Communication and Machine Learning. *Applied Artificial Intelligence Journal*, 11:719–746.
- Thomas Kollar, Jayant Krishnamurthy, and Grant Strimel. 2013. Toward Interactive Grounded Language Acquisition. In *Robotics: Science and Systems*.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November.
- Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2, AAAI'06*, pages 1475–1482. AAAI Press.
- Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nicholas Johnston, Andrew Rabinovich, and Kevin Murphy. 2015. Whats cookin? interpreting cooking videos using text, speech and vision. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 143–152, Denver, Colorado, May–June.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.
- Cynthia Matuszek, Liefeng Bo, Luke S Zettlemoyer, and Dieter Fox. 2014. Learning from Unscripted Deictic Gesture and Language for Human-Robot Interactions. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- Nikolaos Mavridis. 2015. A review of verbal and non-verbal human–robot interactive communication. *Robotics and Autonomous Systems*, 63:22–35, January.
- Deb K Roy. 2002. Learning visually grounded words and syntax for a scene description task. *Computer speech & language*, 16(3-4):353–385, July.
- Luc Steels and Paul Vogt. 1997. Grounding Adaptive Language Games in Robotic Agents. *Proceedings of the Fourth European Conference on Artificial Life*.
- H. Thompson, A. Anderson, E. Bard, G. Doherty-Sneddon, A. Newlands, and C. Sotillo. 1993. The HCRC map task corpus: natural dialogue for speech recognition. In *HLT '93: Proc. of the workshop on Human Language Technology*, pages 25–30. ACL.
- Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate

- Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1494–1504, Denver, Colorado, May–June.
- Terry Winograd. 1972. *Understanding Natural Language*. Academic Press.
- Haonan Yu and Jeffrey Mark Siskind. 2013. Grounded language learning from video described with sentences. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 53–63, Sofia, Bulgaria, August.