Robots can ground crowd-proposed symbols by forming theories of group mind

Joey Anetsberger and Josh Bongard

University of Vermont, Burlington, VT 05401
janetsbe@uvm.edu

Abstract

The non-embodied approach to teaching machines language is to train them on large text corpora. However, this approach has yielded limited results. The embodied approach, in contrast, involves teaching machines to ground abstract symbols in their sensory-motor experiences, but how—or whether—humans achieve this remains largely unknown. We posit that one avenue for achieving this is to view language acquisition as a three-way interaction between linguistic, sensorimotor, and social dynamics: when an agent acts in response to a heard word, it is considered to have successfully grounded that symbol if it can predict how observers who understand that word will respond to the action. Here we introduce a methodology for testing this hypothesis: human observers issue arbitrary commands to simulated robots via the web, and provide positive or negative reinforcement in response to the robot's resulting action. Then, the robots are trained to predict crowd response to these action-word pairs. We show that robots do learn to ground at least one of these crowd-issued commands: an association between 'jump', minimization of tactile sensation, and positive crowd response was learned. The automated, open-ended, and crowd-based aspects of this approach suggest it can be scaled up in future to increasingly capable robots and more abstract language[1].

Introduction

Language has been a central concern in Artificial Intelligence research since the field's inception in the 1950s. Like many other aspects of cognition, it has been addressed with non-embodied and embodied approaches. Non-embodied approaches typically train agents on large pre-existing text corpora (Guha and Lenat, 1993; Collobert and Weston, 2008), or on conversations those agents attempt to conduct with humans (Shawar and Atwell, 2003). However, this approach has yielded limited results.

The embodied approach to language acquisition involves helping agents to detect correlations between particular subsets of sensorimotor experiences and categories, to which words can be attached. However, how or whether humans do this, and how best to enable robots to do this, remains an open question.

[1]A video summary of the work described here is available at youtu.be/j3sB85ENgA8 and the source code is available at github.com/Janetsbe/AnetsbergerAlife2016Code.

The Symbol Grounding Problem

The symbol grounding problem is a long-standing open problem in cognition. It concerns how we can assign meaning to parts of language—symbols—without succumbing to an infinite regress. That is, some symbols must at some point be grounded in something, such as categorical or iconic representations, rather than deriving meaning from other symbols. This is a major problem with the cognitivist approach to intelligence (Harnad, 1990).

Evidence is starting to appear in the literature that suggests that, for humans at least, sensorimotor experience is the 'soil' in which language symbols are ultimately grounded. For example, Pulvermüller and Fadiga (2010) describe how spoken language may be closely coupled to neural circuits related to motor functions. Cangelosi and Harnad (2001) have provided theoretical arguments for *why* sensorimotor experience provides such good grounding for language. First, agents perform 'sensorimotor toil': they acquire knowledge of categories (though not symbols to represent them) through the costly effort of learning through action and feedback. They then perform symbolic theft: these categories are given symbolic representations and shared by those who have performed the necessary toil or who have themselves "stolen" from others. However, exactly how this can be instantiated in machines remains an open question.

Sensorimotor Grounding

Steels (2008) claims that a solution to the symbol grounding problem has been found through the creation of robots who can respond appropriately to human-provided commands. However, it is not clear how such approaches can scale up to more complex robots, large numbers of human tutors, and increasingly abstract language. In this paper we introduce a methodolody that may, in future work, help to support all three. The novelty of our approach in its present form is how it allows people to teach robots aspects of human language of the crowd's choosing.

In other work involving non-human languages, Steels has demonstrated a scalable approach to language acquisition among robots (Steels et al., 2002). The robots participate in language games through which they converge on agreement as to the meaning of words generated by the group. Similarly, Schulz et al. (2012) has shown that robots can jointly form words for relative locations and then draw on the combinatorial power of symbols to dictate directions to novel locations by combining these words in novel ways. Whether or how these robot-generated languages could help them understand human languages however has not been addressed. Here, we focus on robots learning human languages.

Crowdsourcing language acquisition.

Despite these initial successes in teaching robots language (or having them teach themselves), there are many open questions that remain. How should robots (or how do humans) ground abstract language in action? Lakoff and Johnson (2008) have argued that embodied metaphors (such as "do not jump to conclusions") hint that we ground even abstract language in sensorimotor experience, but the mechanisms by which this occurs are unclear. How does the acquisition of some symbols facilitate the acquisition of others? Do caregivers spontaneously constrain their utterances to scaffold language learners (Roy et al., 2009)?

These and other questions can only be addressed by scalable, open-ended and automated infrastructure which enables large numbers of people to teach large numbers of (possibly increasingly abstract) language to increasingly complex robots capable of a broadening set of sensorimotor experiences. Here we introduce such an experimental apparatus that relies on crowdsourcing. Through the web, observers propose arbitrary, natural language commands to robots and provide positive or negative reinforcement for the resulting actions. The robots then learn to predict which actions, under which commands, are likely to generate positive crowd reinforcement. Various hypotheses about language acquisition can then be tested using this apparatus. Here, we first test the hypothesis as to whether robots can indeed ground these crowd-proposed symbols by forming theories of their group mind.

Crowdsourcing has recently been exploited for training robots to interact with humans (Breazeal et al., 2013; Toris et al., 2014), but not for grounding language symbols. We have demonstrated that web participants can collectively design and optimize robots, despite the lack of any explicit reward to the human participants for doing so (Wagy and Bongard, 2014, 2015). In other work we have shown that robots can be trained to form theories of mind about an individual human trainer (Hornby and Bongard, 2012), and even disambiguate between two trainers who disagree about how to reinforce the robot's behavior (Bernatskiy et al., 2014). However, in (Hornby and Bongard, 2012) and Bernatskiy et al. (2014) the robots only learned theories of mind about synthetic trainers: bots who stood in for actual human trainers. Here we demonstrate that, with the right cyberinfrastructure, robots can successfully ground symbols provided by actual human caregivers.

This research was conducted in two phases. First, subjects were recruited to a web interface through which they could issue commands to the robots and reinforce the resulting behaviors. The data generated by this process was then used to train models to predict the crowd's response to a given command and set of actions. See Fig. 1 for a summary of this two-part methodology.

Phase I: Crowd Deployment

In order to allow subjects to see and interact with the robotics simulation, the Twitch.tv[2] video streaming web service was selected. Twitch is a popular internet service for observing others play video games or perform other skilled tasks. Twitch in turn has given rise to "Twitch Plays..." interfaces through which subjects can collectively observe as well as play interactive video games by voting on the next move in the game. It was hoped that the wide appeal and familiarity of this interface would incentivize large-scale participation. In the work described here, we broadcast a live stream from a robotics simulation; subjects could then interact with the robots observed in the stream using live chat (Fig. 2; a video snippet from the deployment can be seen here.).

Phase I Methods

The crowd deployment commenced on October 29th, 2015. The video stream ran continuously and saw use for 22 days. The experiment was terminated then as user traffic had become negligible. During the crowd deployment, subjects were shown a single robotics simulation to collectively interact with through text input. Subjects were allowed to provide candidate commands and reinforcement signals to the simulation (see Table 1 for terminology used throughout this paper). Candidate commands were strings representing votes for what behavior a subject wanted the robot to be evaluated against. These votes were tallied over a three minute interval; the most frequently-issued candidate command at the end of this period became the new, issued command. An issued command was some string dictating how the subjects should reinforce behaviors over the next three minute period. Reinforcement signals were strings indicating whether subjects considered the robot they were viewing to be obeying the issued command ('y') or not ('n').

Robot Simulation. The simulation was developed in the Unity3D[3] engine, with its default graphics renderer, physics engine, and collision solver. It consisted of a scene containing all elements of the simulation: a floor, start loca-

[2] www.twitch.tv

[3] www.unity3d.com

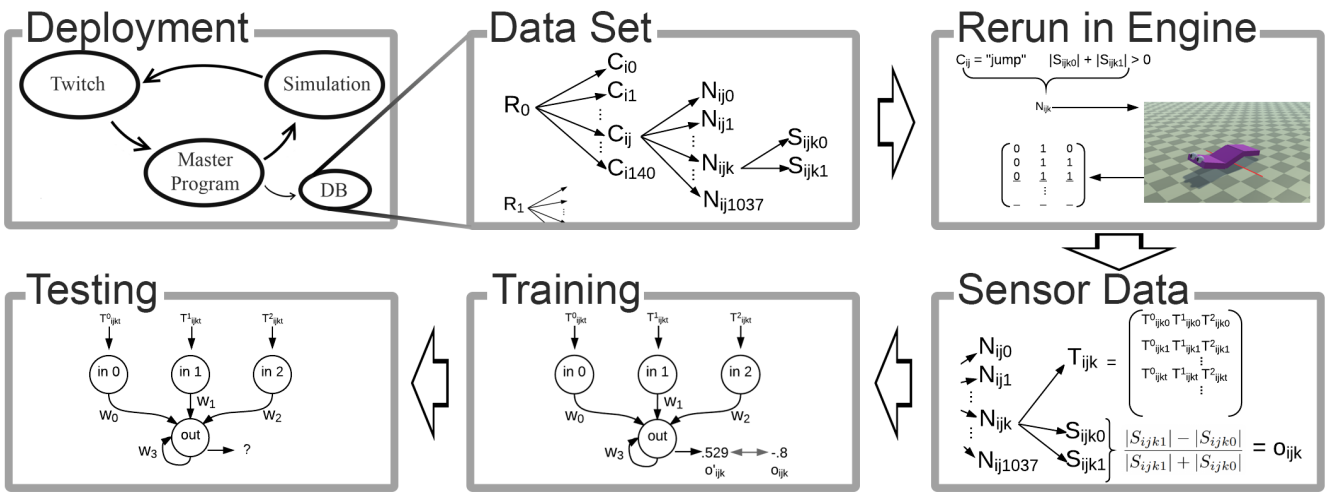

Figure 1: Summary of the methodology. **Deployment**: A master program generates robot controllers, which animate one robot in a physics simulation after another. The video resulting from the simulation is piped in real time to `www.twitch.tv`, where the human subjects issue the robots commands and reinforcement. **Data Set**: The simulations generate a data set comprised of the robots themselves (R), the commands issued to them (C), the controllers run on those robots under those commands (N), and the reinforcement signals provided by the subjects (S). **Rerun in Engine**: All robots which had been issued the command "jump", and received at least one reinforcement signal, were re-simulated. **Sensor Data**: The resulting touch sensor data (T) was recorded and added to the data set, along with the normalized reinforcement signal (o). **Training**: The time series touch sensor data for a randomly-selected half of these controllers were employed to train a recurrent neural network to predict (o'_{ijk}) the crowd's actual response (o_{ijk}) to each controller using CMA-ES. **Testing**: The ability of the trained RNN to ground the symbol "jump" was tested by measuring its ability to predict crowd response to the other half of the controllers.

tion marker, a robot, a camera, and GUI elements providing users with instructions and feedback. Fig. 2 reports a typical scene during deployment. All data pertaining to the robot, its controller, and GUI elements were sent to the simulation by the master program. The simulation ran more or less continously for the 22 days. The simulation was continously recorded and sent as a live video feed to Twitch.tv.

The simulation featured two robot types (Fig. 3). The simple robot was formed of three rigid bodies and two rotational hinge joints. Each body segment was $2.5 \times .6 \times 3.0$ units in size. The simple robot's joints rotated through the sagittal plane, thus restricting movement to forward and backward motion. However, due to asymmetries in collision resolution, the simple robot had the ability to slowly turn, so constraints were imposed in the simulation to frustrate turning. The complex robot consisted of seven segments (three body segments and four legs) and six hinge joints, and was allowed to move about the horizontal plane. Its body segments and leg segments measured $1.2 \times .4 \times 3.0$ and $.6 \times .4 \times 3.0$ units respectively. Both robots were equipped with nonfunctional eyes to provide subjects with a robot-centric frame of reference through which to issue commands (e.g. 'move forward').

During the 22 days of deployment, the two robots alternated every hour: the crowd saw the simple robot for an hour, then the complex robot for an hour, then the simple robot again, and so on. During each hour period, the command issued to the robot changed every three minutes.

During each three minute period, all candidate command votes were counted, recorded, and cleared. The most frequently-input candidate command during this period was issued to the robot for the next three minute period. Within a three minute period, six random controller were evaluated on the robot, each for 30 seconds. We henceforth refer to each of these as a 'robot evaluation'. The total number of positive and negative reinforcement signals issued in response to each robot evaluation was recorded.

A robot's controller was defined as an $N \times 3$ matrix with dimension sizes corresponding to the number of joints and three parameters which dictated a given joint's amplitude (α), frequency (β), and phase offset (γ). Thus, each joint at each time step t was issued a desired angle $\alpha \sin(\beta t + \gamma)$, where $\alpha \in [.6, 1.5]; \beta \in [.01, .09]$; and $\gamma \in [-10\pi, 10\pi]$. These constants were drawn from these ranges randomly using a uniform distribution and populated the matrix for each controller. The ranges were selected to maximize variation in behavior. Thus, the subjects observed random behaviors (rather than evolved or learned ones) in this study.

The robot's color was changed whenever the controller was changed. The colors cycled through blue, orange, and violet, and then back to blue. Subjects were instructed to indicate the color of the robot that they were reinforcing. For

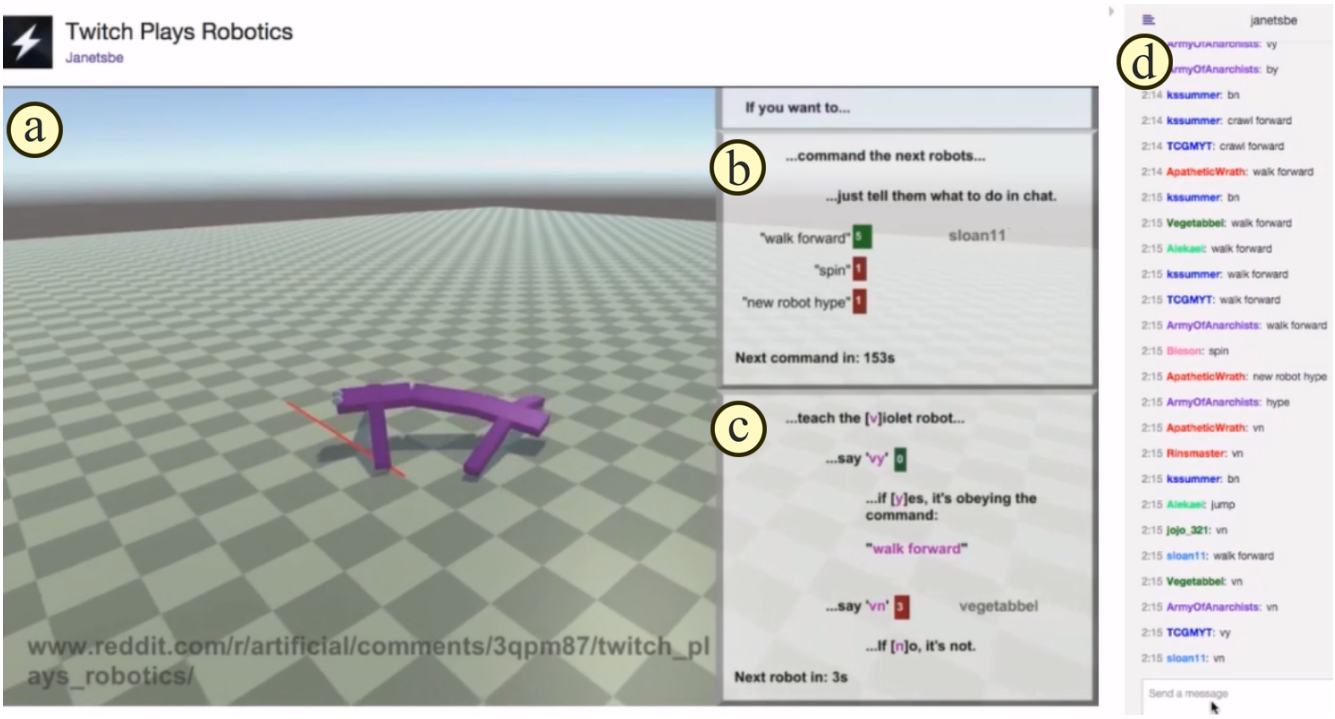

Figure 2: The interface, as seen by participants interacting with the simulation. **a)** The simulation video feed. **b)** A panel prompting participants to command what the robot should do next. At this point, one subject has proposed 'walk forward' as the next command, and two other subjects have voted for this. After a five-minute period, the most popular command takes effect. **c)** A panel prompting users to provide positive ('y') or negative ('n') reinforcement for the current action under the current command, which here is 'walk forward'. Votes for either signal allow subjects to see how others are reinforcing this action. **d)** Twitch's chat interface, through which subjects send commands, reinforcement, or other miscellaneous chatter.

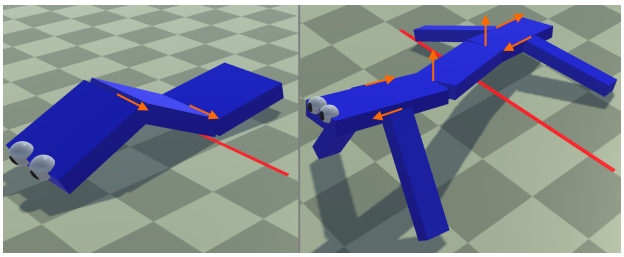

Figure 3: The "simple" (left) and "complex" (right) robot body types. Body segments were connected through hinge joints with normals as indicated by the vectors.

example if the user wished to positively reinforce the blue robot, she would type 'by'. If she wished to negatively reinforce it, she would type 'bn'. Because video streams broadcast through Twitch are delayed for approximately ten to twenty seconds with variation between viewers, this coded input allowed for correct assignment of reinforcement to a controller despite the delay. That is, subjects likely observed a controller (and provided reinforcement to it) after it had already terminated on the host computer.

The simulation's GUI provided directions and feedback to the subjects. In the command panel, the subjects were told: "If you want to command the next robots, just tell them what to do in chat". In the reinforcement panel, the subjects were told: "If you want to teach the [v]iolet [or [o]range or [b]lue] robot, say 'vy' if yes, it's obeying the command: [command]. Say 'vn' if no, it's not."

Video Streaming and Chat Capture. Video streaming was done on Twitch.tv, an internet video streaming service. The service allows a host user to broadcast a live video stream to an arbitrary number of internet users who may (optionally) talk to each other and the host through an embedded IRC service. Twitch was used due to its established user base, chat functionality, available API, as well as the existence of the previously mentioned "Twitch Plays" phenomenon.

Video streaming was done using `FFSplit`[4] to capture and encode video of the simulation being run on the host machine and stream it to Twitch, where it was viewed by subjects. Subjects saw this simulation and were able to interact with it through Twitch's chat service. Subjects were

[4]http://www.ffsplit.com

Term	Definition
Subject	Any Twitch user who provided either a reinforcement signal or candidate command.
Robot type	$R_i, i \in \{0, 1\}$, The simple (R_0) and complex (R_1) robots.
Candidate command	A string entered by a participant. Considered a vote for a future issued command.
Issued command	C_{ij}. The jth [c]ommand the ith robot was evaluated against.
Controller	N_{ijk}. The kth co[n]troller issued to the ith robot under command j. A matrix of coefficients used to dictate a robot's movements.
Robot evaluation	The 30-second simulation resulting from a controller issued to a robot under a given command. Six such evaluations were performed for each issued command.
Reinforcement signal	S_{ijkl}, for $l \in \{0, 1\}$. The number of negative ($l = 0$) and positive ($l = 1$) reinforcements collected from subjects in response to the kth controller under the jth command.
Normalized reinf. signal	$o_{ijk} \in [-1, 1]$. $o_{ijk} = \frac{\lvert S_{ijk1}\rvert - \lvert S_{ijk0}\rvert}{\lvert S_{ijk1}\rvert + \lvert S_{ijk0}\rvert}$
Touch sensor data	T_{ijk}. Touch sensor data generated by robot i under command j with controller k.
Predictive model	A recurrent neural network (RNN) which predicts o_{ijk} given T_{ijk}. Its output, o'_{ijk}, is an estimation of the crowd's response to robot evaluation ijk.

Table 1: Terminology used throughout this paper.

allowed to provide any combination of characters to the chat. Any string of length greater than two or less than thirty that were not part of Twitch's default filtered word list were considered candidate commands. Subjects could provide commands and reinforcement at any time.

Subjects provided reinforcement though specific coded input. This consisted of strings of length two which matched the pattern of $(b|o|v)(y|n)$ where the first character represented the color of a robot (**b**lue, **o**range, or **v**iolet), and the second either **y**es or **n**o. Further, only reinforcement signals corresponding to the current or immediately previous robot evaluation were counted. For example, if the current evaluation's color was blue and the previous evaluation had been orange, the following strings would have been parsed as reinforcement signals: 'on', 'oy', 'bn', and 'by'.

Recruitment and Incentivization. Subjects were recruited through Reddit, a popular internet message board site consisting of hundreds of sub-sites (called subreddits) which generally limit posts to a specific topic. Posts were issued to relevant subreddits directing subjects towards this main post on the `artificial` subreddit[5]. Here, subjects were directed to an agreement page which sent them to the twitch channel[6].

Subjects were incentivized to interact with the system by adding features to the GUI which were meant to provide the subjects with a sense of involvement with the simulation. This was done by displaying subjects' user names when valid input was given. This showed subjects that their input was being counted and used; their actions had an impact. For the scope and scale of this study, this seemed sufficient.

Phase I Results

The crowd deployment lasted 22 days. During this time, at least 6,388 robot evaluations were seen by hundreds of subjects, who provided hundreds of commands and thousands of reinforcement signals. Table 2 reports the relevant values.

General Figures	**Values**
Subjects	424
Robot evaluations sent	57,108
Robot evaluations observed	$\geq$ 6,388
Subject inputs	16,449
Commands	**Values**
Candidate commands entered	8943
Candidate commands/subject	≈ 21.1
Distinct commands issued	266
Most frequently issued commands	jump (385) walk forward (58) move forward (41) run (26) crawl forward (20).
Reinforcement signals	**Values**
Reinforcement signals entered	7503
Mean reinforcement signals/eval.	≈ 1.18
Mean reinf. signals/subject	≈ 17.7
Proportion of positive reinforcement	$o = 0.28$

Table 2: Crowd deployment participation results.

Participation produced a data set in which 6,388 controllers received at least one reinforcement signal. From this data set, no significant differences were found between how the crowd interacted with the simple and the complex robots: both had similar distributions in their spread and frequency of commands, and both received similar consistency of reinforcement.

Phase II: Offline Learning

In the second phase of this experiment, models were trained to ground symbols by learning relationships between com-

[5] www.reddit.com/r/artificial
[6] www.twitch.tv/janetsbe

mands, sensor data, and reinforcement.

The most commonly issued command for both robots was "jump": 1072 simple and 698 complex robot controllers were evaluated under this command. Data belonging to robots given this command were used to test whether the crowd was able to provide trainable input. Due to the online nature of the live deployment, some robot evaluations were prematurely terminated and did not remain visible to the crowd for their full thirty seconds. These were discarded, resulting in 1037 simple and 675 complex robot evaluations used during this phase. Since each controller directly determined the robot's behavior for the duration of its run, any controller could be re-evaluated to obtain any additional data not recorded during the live deployment.

Phase II Methods

During this phase, the goal was to determine if there exists features in the "jump" data set which could be used to train a model that predicts the crowd's response. A recurrent neural network (RNN) was employed as such a model (Fig. 1). It was trained such that, when supplied with sensor data generated by a controller, it outputs a successful prediction of the normalized reinforcement signal o_{ijk} (Table 1).

Since the chosen command was "jump", it seemed likely that this command should have some relationship to whether, when and how the robot contacts the floor. For this reason touch sensor data was recorded for these controllers.

All of the controllers evaluated under the command "jump" and which received at least one reinforcement signal in return were then re-evaluated with touch sensors added to the robots. If a body segment touched the ground, its touch sensor recorded a value of 1; otherwise a 0 was recorded. This was stored in matrix T_{ijk} such that element $t_{ijkmn} \in 0, 1$ indicated whether body part n contacted the ground at time step m for controller N_{ijk}. Each of these controllers was re-evaluated for the same duration as its original run during deployment. Since the master program gathering data from the Twitch channel was also controlling the simulation's timing, there was some slight variation in the duration of each robot evaluation, which resulted in slight variations in the time step length. Thus, m ranged between 3350 and 3761 and 3133 to 3719 for the simple and complex robots respectively. During model training and testing, sensor data was thus truncated to $m = 3350$ and $m = 3133$ for the simple and complex robots respectively to make data consistent in shape for each robot type.

Model training. Offline learning was conducted separately for the simple and complex robot types. For each of the two robots, half of their controllers were assigned randomly to the training set. This was repeated 100 times, leading to 100 models trained on 100 partially-overlapping training sets. The RNN associated with the simple robot had three input nodes (corresponding to the number of touch sensors and body segments) and one output node, with one synapse from each input to the output, and a recurrent synapse on the output node. The RNN trained and tested on the complex robot had seven input nodes and was otherwise identical. A graphical representation of this RNN for the simple robot's data can be seen in Fig. 1.

During training, each controller in the training set for R_i had its sensor data run through the neural network such that the network was updated m times. Then, the output node's value was read out as o'_i, the predicted normalized reinforcement signal for the ith controller based on its sensor data. This was done for each controller in the training set. Using this, the error for the model was calculated using the following objective function:

$$e = \left(\sum_{i=1}^{N_y} \frac{|o_i - o'_i|}{2N_y} + \sum_{j=1}^{N_n} \frac{|o_j - o'_j|}{2N_n} \right) /2 \qquad (1)$$

Since the normalized reinforcement signals mostly consisted of unanimous negative reinforcement ($o_i = -1$), accounting for 77.7% and 77.8% of robot evaluations for the simple and complex robots respectively, this error function weighed two subsets of the training set equally in calculating error to avoid over-fitting the model to output -1 for any set of sensor values. One subset consists of all robot evaluations with $o_i = -1$ (of which there were N_n) and the other for all others ($o_i > -1$; N_y), which must by definition contain at least one positive reinforcement signal. In doing so, the error calculation weighs both groupings equally and an constant output of $o_i = -1$ for every controller would result in, at best, $e = 0.5$. Error is, then, the sum of differences in each group, averaged, where o is the actual normalized reinforcement signal for each robot evaluation and o' is the model's prediction.

The popular, continuous-value optimization method CMA-ES (Hansen et al., 2003) was used to train RNNs against each of the 100 training sets, for both robots, resulting in 200 runs of CMA-ES. For each run, a random initial solution array was used. Synaptic weights were bounded on $[-1, 1]$, with an initial step size of 0.1. At the termination of each run, the RNN with lowest error was extracted, leading to 100 RNNs for each robot.

Phase II Results

The ability of these 200 models to generalize their predictions to unseen controllers was measured as follows. The mean error of each model when exposed to its testing set was first computed using Eqn. 1. Then these errors for the 100 models for each robot were in turn averaged. The resulting mean errors are reported as 'experiment' in Fig. 4.

In order to determine the accuracy of these predictions, the models were also exposed to the same testing set, but the normalized reinforcement signals in the set were randomly permuted ('permuted control' in Fig. 4). A second

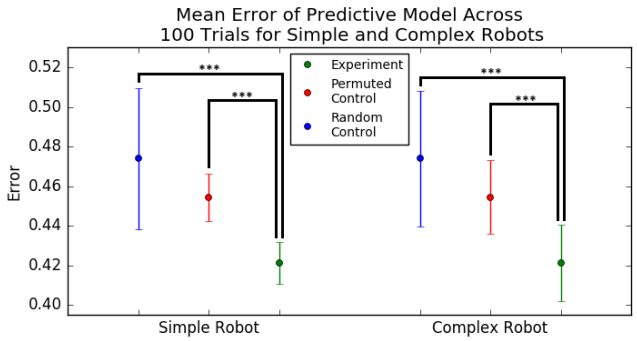

Figure 4: Results from the predictive model on unseen test data, the permuted test data set, and random RNNs on test data. *** denotes $p < .001$ as reported by a Student's T-test.

control experiment was also conducted in which RNNs with randomly-assigned synapses also made predictions on the same, unpermuted test sets ('random control' in Fig. 4).

Discussion

Fig. 4 shows that both robots employed here were able to successfully ground at least one symbol ('jump') in their own sensorimotor experiences. Further, since this symbol and its meaning were provided solely by human subjects, this suggests that the crowd reached some implicit, mutually agreed upon recognition of this category of behavior in the robots, and that they were able to pass this category on to non-humanoid robots through a simple interface in a short time period, with no explicit reward from the investigators.

Precisely how the robots may have grounded "jump" remains unknown. It may be the case that robots who tended to spend less time touching the ground were more likely to be positively reinforced by the crowd. This hypothesis is supported by the finding that there is a negative correlation between the proportion of time a robot spent with at least one body segment touching the ground and the crowd's normalized reinforcement signal, for both robots (Fig. 5). Despite this, it is unclear whether the models learned this relationship, or instead discovered some more subtle function of the touch sensor data that better predicts the crowd's response. In a similar manner, it is not clear if each robot grounded this symbol in the same way.

Conclusions and Future Work

Here we have demonstrated a unique methodology for enabling robots to gradually acquire language: crowd-generated language symbols are grounded by detecting correlations between sensorimotor experience and crowd-generated reinforcement.

Given the unconstrained nature of the interface, the subjects could have tried to teach the robots unlearnable commands such as 'xyzf' or 'prove Fermat's Last Theorem'.

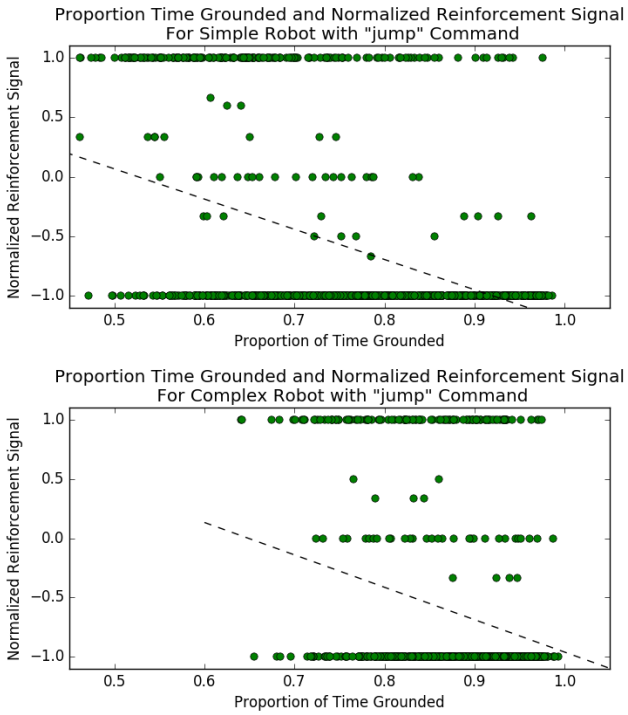

Figure 5: Comparison of normalized reward signal and proportion of time grounded for the 1037 simple and 675 complex robots with the "jump" command. Proportion of time grounded is the number of time steps where at least one sensor value is 1 divided by the total number of time steps. These values are negatively correlated with $p < .001$.

Despite the fact that there was a long tail of infrequently-proposed commands such as these, the most frequently-issued commands (Table 2) were motoric as well as appropriate for the robot's morphology (as opposed to 'clap your hands'). This suggests that the observed behavioral limitations of the robots may have steered the crowd toward at least one command that, with sufficient reinforcement, was learnable ('jump'). This observation accords with Roy et al. (2009), who showed that three human caregivers constrained their utterances given the current language abilities of a human child.

Because of the recent success of deep learning approaches, much work in AI has become focused on recognition rather than understanding. Furthermore, recognition is much easier to measure than understanding: The ability of an algorithm to recognize human faces in an image is much easier to quantify than its understanding of humans. Our approach is predicated on the speculation that understanding—even the understanding of abstract concepts—is ultimately grounded in sensorimotor experience. We provide a method for quantifying this in the domain of language: the robots tested here understand the word 'jump' in the sense that they have learned an association between that word, a set of ac-

tions generated in response to that word, and the crowd's responses to those actions.

For this paper, a small subset of the overall data set acquired during deployment was analyzed. However, many other potentially groundable symbols were provided by the crowd. Future work will involve instrumenting the robots with more sensors to attempt the grounding of more of these symbols, and expanding the models such that they can potentially ground more than one symbol, or detect semantic similarities between grounded symbols. Further, since we have data corresponding to different morphologies, we may be able to discover if morphology impacts the way a robot grounds symbols. In subsequent deployments of the system we also plan to enable robots to ground symbols in real time rather than retroactively. Also, we wish to exploit the fact that the models form a theory of group mind about the crowd to minimize user fatigue: one robot should be able to predict how the crowd will react to another robot, even before the latter robot is shown to them. In this way, not every robot would need to be reinforced by the crowd.

Finally, we wish to investigate how increasingly complex robot morphologies, task environments, and behaviors influence the crowd's behavior such that they incrementally train the robots to understand increasingly abstract language.

Acknowledgments. This work was supported by the National Science Foundation under projects PECASE-0953837 and INSPIRE-1344227.

References

Bernatskiy, A., Hornby, G. S., and Bongard, J. C. (2014). Improving robot behavior optimization by combining user preferences. In *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, volume 14, pages 726–733.

Breazeal, C., DePalma, N., Orkin, J., Chernova, S., and Jung, M. (2013). Crowdsourcing human-robot interaction: New methods and system evaluation in a public environment. *Journal of Human-Robot Interaction*, 2(1):82–111.

Cangelosi, A. and Harnad, S. (2001). The adaptive advantage of symbolic theft over sensorimotor toil: Grounding language in perceptual categories. *Evolution of Communication*, 4(1):117–142.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM.

Guha, R. V. and Lenat, D. B. (1993). Cyc: A midterm report. In *Readings in Knowledge Acquisition and Learning*, pages 839–866. Morgan Kaufmann Publishers Inc.

Hansen, N., Müller, S. D., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346.

Hornby, G. S. and Bongard, J. (2012). Learning comparative user models for accelerating human-computer collaborative search. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, pages 117–128. Springer.

Lakoff, G. and Johnson, M. (2008). *Metaphors We Live By*. University of Chicago press.

Pulvermüller, F. and Fadiga, L. (2010). Active perception: sensorimotor circuits as a cortical basis for language. *Nature Reviews Neuroscience*, 11(5):351–360.

Roy, B. C., Frank, M. C., and Roy, D. (2009). Exploring word learning in a high-density longitudinal corpus. In *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*.

Schulz, R., Wyeth, G., and Wiles, J. (2012). Beyond here-and-now: extending shared physical experiences to shared conceptual experiences. *Adaptive Behavior*, 20(5):360–387.

Shawar, B. A. and Atwell, E. (2003). Using dialogue corpora to train a chatbot. In *Proceedings of the Corpus Linguistics 2003 Conference*, pages 681–690.

Steels, L. (2008). The symbol grounding problem has been solved. So what's next? In *Symbols and Embodiment: Debates on Meaning and Cognition*, pages 223–244. Academic Press, New Haven.

Steels, L., Kaplan, F., McIntyre, A., and Van Looveren, J. (2002). Crucial factors in the origins of word-meaning. *The Transition to Language*, 12:252–271.

Toris, R., Kent, D., and Chernova, S. (2014). The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. *Journal of Human-Robot Interaction*, 3(2):25–49.

Wagy, M. and Bongard, J. (2014). Collective design of robot locomotion. In *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, volume 14, pages 138–145.

Wagy, M. D. and Bongard, J. C. (2015). Combining computational and social effort for collaborative problem solving. *PloS ONE*, 10(11):e0142524.