World Interaction Networks: Grounding Natural Language to World Updates with Minimal Supervision

Siddharth Karamcheti and **Eugene Charniak** and **Stefanie Tellex**
{siddharth_karamcheti@, eugene_charniak@, stefie10@cs.}brown.edu
Brown University Department of Computer Science
Providence, Rhode Island 02912

Abstract

We examine the problem of interpreting sequences of natural language utterances, for the goal of understanding, reasoning, and answering questions about text. The challenge lies in the dependence across utterances, with the interpretation of one affecting the understanding of the next. Also challenging is that often, the language in question assumes knowledge about, or specifies an interaction with an underlying environment. Existing approaches for these problems are limited in that they either assume a fixed method for communicating with this world, or seek to learn its dynamics implicitly, from incomplete information. In this work, we address these problems by presenting the World Interaction Network (WIN), an approach that explicitly factors in the underlying environment by mapping language utterances to concrete updates on an external world model, via a novel reinforcement learning training procedure. Our approach allows us to leverage powerful computational abilities for reasoning and deduction inherent in the world model, instead of learning these implicitly. Instead, we focus the learning on the easier problem of mapping utterances to atomic updates on the world state. We show that by factoring in the World Model as an explicit component, the WIN framework achieves near state-of-the-art results on three existing human-robot interaction datasets, with a fraction of the supervision utilized by baselines. Additionally, we show that with fewer than 20 fully annotated examples, we can augment the WIN, allowing it to understand and answer questions about short stories of up to 20 sentences, obtaining near-perfect accuracy on a suite of 20 language grounding tasks based on the 1000 example version of the bAbI Question-Answering Tasks.

Introduction

Many tasks across human-robot interaction and natural language processing assume a unifying world model, responsible for maintaining the characteristics of an environment. These models encode important features, ranging from static information about the world's attributes, to dynamic information about the relationships between the entities. In human-robot interaction, for example, a possible world model may include the locations of all the objects a robot may interact with, along with the full set of actions it can take. In question-answering on short stories, a possible world model could include other time-dependent information, like the motivations and goals for each character.

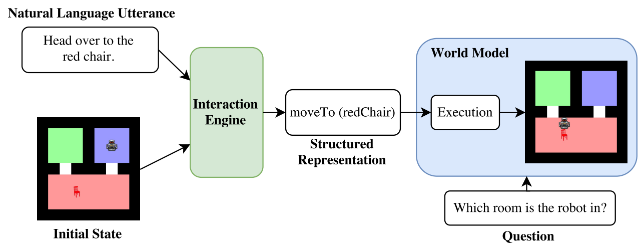

Natural Language Utterance
Head over to the red chair.
Initial State
Interaction Engine
moveTo (redChair)
Structured Representation
World Model
Execution
Which room is the robot in?
Question

Figure 1: World Interaction Network Example. Interaction Engine processes utterances, and outputs a structured representation to be executed (e.g. by a robot) to update world state. We can query the world state to answer questions.

In both of these cases, language acts to update the state of the world. Mapping language to these concrete updates then becomes an important problem, as doing so not only grants the ability to execute these updates (i.e. in a human-robot interaction setting), but also reason about the dynamics of a given world (i.e. in a question-answering setting). Furthermore, if these updates are structured in such a way that they can all be performed on some external world model, like a database, state machine, or the physical world, we can fully interpret how a set of utterances changes an environment. In doing so, arbitrary users can gain transparency into the full dynamics, and utilize complex tools and algorithms to perform structured reasoning, and make logical deductions.

Existing approaches for human-robot interaction and question-answering either try to map utterances to a structured representation, treating the rest of the world as a black-box, or try learn the world dynamics in-model, directly from the data. In the case of the former, most approaches require language that is fully annotated with the ground truth structured representation, which is extremely expensive to collect, and not robust to representational changes. In the case of the latter, most "end-to-end" methods are not easily interpretable, with the current state of the world often existing as some vector representation inside a neural network. Additionally, any structured reasoning needs to be learned, and external tools cannot be effectively leveraged.

In this work, we address these problem by introducing a new framework, the World Interaction Network (WIN), for grounding natural language to structured updates on a preexisting environment. World Interaction Networks are characterized by a World Model, responsible for explicitly tracking the world state, and an Interaction Engine, that maps language to structured world updates. Because all updates happen external to the learned Interaction Engine, we present a novel reinforcement learning algorithm for training WIN models, given minimal feedback from the world. Namely, we show that we can train WIN models solely given the initial state, the set of (possibly numerous) language utterances to ground, and a weak validation constraint to be run on the final world state. We assess the efficacy of our WIN models through application to three recently proposed datasets for human-robot instruction grounding, as well as a hybrid task combining language grounding and text-based question answering. For the latter, we introduce a new dataset based on the well-known bAbI tasks to fully demonstrate the effectiveness of our framework. The bAbI Tasks test a wide variety of skills, from tracking relations between entities, to reasoning and deduction tasks like counting, positional reasoning, and path finding (Weston et al., 2015), making them the perfect testbed for evaluating the WIN.

On tasks in language grounding for human-robot interaction, we show that the WIN can match state-of-the-art neural network methods, given less information. We also show that because the World Model is factored out of the learning process in the WIN, we can shift the reasoning required for question-answering outside of the model, focusing the learning on the task of mapping language to structured world updates. As a result, we show that given fewer than 20 fully-annotated examples (out of the full training set of 1000 examples), our WIN models can obtain higher than 95% accuracy on 18 of the 20 tasks, matching the performance of upper bound models that were given significantly more information, and significantly beating the performance of end-to-end memory-augmented neural network approaches.

Related Work

There is a large body of work that examines different methods for grounding language to structured and unstructured representations. Karamcheti et al. (2017) learn a fully supervised model, the deep recurrent action/goal grounding network (DRAGGN) for mapping human-robot instructions specifying either actions or goal conditions to a factored representation. We utilize the same factored representation in our work, as it is a succinct functional form that grants the ability to handle a wide variety of language in a very efficient manner, unlike other representations like action trajectories or lambda calculus expressions. Other work is concerned with interpreting utterances in weakly supervised settings. Instead of being provided with full annotations, the only supervision signal is based on the final state of the world, after grounding. Artzi and Zettlemoyer (2013) learn a weakly-supervised CCG Parser for navigational instruction following. Williams et al. (2017) also learn a weakly-supervised CCG Parser for instruction grounding, but with a smarter validation function, eliminating the need for explicit planning. More recently, Misra, Langford, and Artzi (2017) use policy gradient methods in a contextual bandit setting with reward shaping to map visual observations and text to actions. Guu et al. (2017) combine REINFORCE (Williams, 1992) with Maximum Marginal Likelihood to map utterances to programs that encapsulate the intent of the language specification. Unfortunately, these newer reinforcement learning methods require significant amounts of data, and tens of thousands of training iterations to converge.

Other work models the world implicitly, via the use of end-to-end neural networks. Most of the following evaluates on the bAbI Tasks (Weston et al., 2015), a set of 20 question-answering tasks that test skills in entity tracking, reasoning, and deduction. End-to-End Memory Networks (Sukhbaatar et al., 2015) are based on the more involved Memory Networks (Weston, Chopra, and Bordes, 2014), and read stories in multiple passes, attending to different sentences with each pass. Henaff et al. (2016) develop the Recurrent Entity Network, a type of Recurrent Neural Network (RNN) consisting of multiple independent modules, each responsible for tracking separate entities of a story, allowing for efficient question answering after only a single pass. Santoro et al. (2017) propose Relation Networks, for explicitly modeling relationships between entities over time. However, such methods fail to be openly interpretable and transparent, because the representation of the world state lives internal to the neural network. Additionally, these models all require a significant number of training examples, and fail to work on smaller datasets (Henaff et al., 2016). Our approach, in contrast, performs all language grounding in the external World Model, in a very transparent, interpretable manner. We also show how we can use our model in low-data situations, with extremely limited supervision.

Problem Setting

We consider the problem of mapping natural language utterances to explicit updates on a world model, such that after processing, we end up in a desired target state. More concretely, given an example $\mathbf{x} = (s_0, \mathbf{u})$, where s_0 denotes the initial state of the world, and $\mathbf{u} = u_1, u_2, \ldots u_n$ are the series of natural language utterances, we seek to find the best set of world updates $\hat{\mathbf{w}} = \hat{w_1} \ldots \hat{w_n}$ that when applied to the initial state s_0 result in goal state $\mathbf{y} = s_T$. Note that in this work, all world updates w_i are completely deterministic, so $s_T = \mathbf{w}(s_0)$. Furthermore, we assume that each utterance u_i uniquely and independently specifies an update w_i. As such, our formal objective is as follows:

$$\hat{w_1} \ldots \hat{w_n} = \operatorname*{arg\,max}_{w_1 \ldots w_n} \left[\prod_{i=1}^{n} \Pr(w_i \mid s_{i-1}, u_i) \right] \qquad (1)$$

We note here that this objective is similar to the way (Guu et al., 2017) formalize their parsing objective. Specifically, we note a similarity between the selection over programs $\mathbf{z}$ based on natural language utterances $\mathbf{u}$ utilized in their semantic parser, and our selection over world updates $\mathbf{w}$.

Approach

In order to decompose the problem of mapping arbitrary natural language utterances to concrete world updates, we define World Interaction Networks, with the following two components: A World Model, used to track the world state,

Algorithm 1 Parallel A2C with Factored Action Space

```
// Parameters: $\theta_f, \theta_a, \theta_v$ (Policies and Value Function)
// Restore $\theta_f, \theta_a$ from supervised policy (if pre-training)
Set episode counter $E \leftarrow 1$
Initialize parallel environments $\mathbf{P}$ with Training Examples
repeat
    Initialize time step $t \leftarrow 1$
    Reset gradients: $d\theta_f \leftarrow 0, d\theta_a \leftarrow 0, d\theta_v \leftarrow 0$
    Create 2D Arrays $\mathbf{s}, \mathbf{u}, \mathbf{v}, \mathbf{r}$ indexed by $t, p \in \mathbf{P}$
    repeat
        Get states, utterances $\mathbf{s_t}, \mathbf{u_t} \leftarrow \mathbf{P}$
        Compute $\pi_\mathbf{f}, \pi_\mathbf{a}, \mathbf{v_t}$ for all $\mathbf{P}$ in single pass
        for $p \in \mathbf{P}$ do
            if $p$ is not terminal then
                Sample $f \sim \pi_\mathbf{f}[p]$, $\mathbf{a} \sim \pi_\mathbf{a}[p]$
                Perform world update $w = (f, \mathbf{a})$
                Receive immediate reward $\mathbf{r_t}[p] \leftarrow r$
            end if
        end for
        $t \leftarrow t + 1$
    until All environments in $P$ are done
    for $p \in \mathbf{P}$ do
        $R \leftarrow 0, \tilde{A} \leftarrow 0$
        for $i \in t-1, \ldots, 1$ do
            // Compute discounted rewards, advantages
            $R \leftarrow \mathbf{r_i}[p] + \gamma R$
            $\tilde{A} \leftarrow \big(\mathbf{r_i}[p] + \gamma \mathbf{v_{i+1}}[p] - \mathbf{v_i}[p]\big) + (\gamma\lambda)\tilde{A}$
            $d\theta_f \leftarrow d\theta_f + \nabla_{\theta_f} \tilde{A} \cdot \log \pi_f(f \mid s_i, u_i; \theta_f)$
            $d\theta_a \leftarrow d\theta_a + \nabla_{\theta_a} \tilde{A} \cdot \log \pi_a(\mathbf{a} \mid s_i, u_i; \theta_a)$
            $d\theta_v \leftarrow d\theta_v + \partial(R - \mathbf{v_i}[\mathbf{p}])^2$
        end for
    end for
    Perform updates of $\theta_f, \theta_a, \theta_v$ using accumulated gradients.
until $E > E_{max}$
```

and an Interaction Engine, responsible for mapping utterances to the corresponding world update.

World Interaction Networks

In this section, we develop the World Interaction Network (WIN) capable of working with externally specified World Models. Because the World Model is external (i.e. it takes the form of some Database, or lives in some code running on a robot), the representation output by the Interaction Engine needs to be structured. Only from this structured representation can the external World Model actually update the world state, to serve as the WIN input at the next time step. Because this hand-off is non-differentiable, and the inputs to the next time step are determined by the external update procedure, it is impossible to train the Interaction Engine in a supervised fashion, with standard back-propagation.

Instead, we reformulate the objective in Eq. 1 as a reinforcement learning problem, where at every time step, the Interaction Engine receives a new observation, or utterance u_i, and picks an action w_i to apply to the World Model to maximize the total expected reward (of ending up in the desired goal state). This allows us to use policy gradient methods (Williams, 1992), providing a gradient that can be used to train the Interaction Engine. Specifically, we provide an algorithm based on on Advantage Actor-Critic (A2C) (Mnih et al., 2016) for training our Interaction Engine.

World Model We assume a World Model $\mathcal{W}$, consisting of a set of entities and objects. Because it is externally specified, we also assume access to the set of structured representations $\mathcal{R}$ used to represent all possible world updates.

While there are many valid structured representations we could use, including lambda calculus, a structured query language like SQL, or even a lightweight functional programming language (Guu et al., 2017), we opt to use the representation utilized by Karamcheti et al. (2017) in their system for language grounding, where all world updates can be parameterized by a single function that takes a variable number of arguments. Not only is this a lightweight representation, but because of it's factored nature (arguments can be shared across different function calls), we can perform very efficient learning. Concretely, we assume access to the set of functions $\mathcal{F}$ and arguments $\mathcal{A}$ such that $\mathcal{R} = \mathcal{F} \times \mathcal{A}$. Arguments include the primitive objects and entities of the underlying world, as well as simple predicates or attributes of the same. Note that in this work, we assume that every world update w consists of a single function call $f \in \mathcal{F}$, called with (possibly several) arguments $\mathbf{a} = a_1 \ldots a_n \in \mathcal{A}$. We leave the handling of nested function calls to future work. Given such a parameterization, we can consider the World Model $\mathcal{W}$ as a state machine consisting of a state s, where every function call f with arguments $\mathbf{a}$ transforms the state in a pure, deterministic manner ($s_{t+1} = f(\mathbf{a}, s_t)$).

Interaction Engine Training Objective We now develop the Interaction Engine objective and training algorithm given this setup of the problem. If we return to the overall objective outlined in Eq. 1, we see that:

$$\prod_{i=1}^{n} \Pr(w_i \mid s_{i-1}, u_i) = \prod_{i=1}^{n} \Pr(f_i, \mathbf{a_i} \mid s_{i-1}, u_i) \quad (2)$$

If, at training time, we were given fully annotated data of the form $(u_i, f_i, \mathbf{a_i})$, this would be a straightforward objective, learnable via traditional supervised learning techniques. However, we don't have any knowledge about the mapping between utterances and function-argument tuples. Instead, we are only given examples of the form $(s_0, u_1 \ldots u_n, s_T)$, which just tells us the desired goal state of the world.

We can reformulate Eq. 2 to work in this setting by treating this as a model-free reinforcement learning problem. Concretely, at each time step t, with the current world state s_t, we receive a new utterance u_t, that informs our selection over actions, or world state updates $w_t = (f_t, \mathbf{a_t})$. To be explicit, an action in our setup consists of the function f to be called, and the arguments $\mathbf{a}$ it is to be called with. Every time we perform an update, we receive a reward r_t that is dependent on our current state, and the chosen update. In our setup, because we do not have a measure of the quality of performing an update w_t at any intermediate step s_t, the majority of our rewards r_t are 0. Our only non-zero reward comes at the last time step $T-1$, where $r_{T-1} = +1$ if $s_T = w_{T-1}(s_{T-1})$, and is 0 otherwise. To smooth our rewards over all time steps, we decay our reward over the entire time horizon by a discount factor γ, that assigns more reward to recent update actions taken. We refer to the discounted reward received at time step t as R_t.

As such, instead of maximizing the probability of selecting the correct update given each natural language utterance, as in Eq. 1, the problem now becomes one of maximizing the total expected reward, across all natural language utterances:

$$\max J(\theta) = \mathbb{E}_{w \sim \pi(w|s,u;\theta)} \big[\sum_t R_t \big] \tag{3}$$

where the expectation is over $\pi(w \mid s, u) = \pi(f, \mathbf{a} \mid s, u)$, the probability of generating the update represented by function-argument $(f, \mathbf{a})$ given the utterance u, and the current state of the world s. We represent this policy π with a deep neural network parameterized by weights θ.

One way to optimize this is to use REINFORCE (Williams, 1992):

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{\pi(w|s,u;\theta)} \big[\sum_t R_t \big] \\ &= \mathbb{E}_\pi \big[\nabla_\theta \log \pi(w \mid s, u; \theta) R_t \big] \end{aligned}$$

Here, gradient updates are given by the gradient of the logarithm of our policy, scaled by the discounted reward at time t. Unfortunately, a known problem with REINFORCE is high variance, resulting in several training episodes for the policy network to converge. To remedy this, we instead use Advantage Actor-Critic (A2C), where we reformulate the above objective by substituting the discounted reward R_t with an estimate of the advantage A_t (Mnih et al., 2016; Sutton et al., 1999):

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_\pi \big[\nabla_\theta \log \pi(w \mid s, u; \theta) A_t \big] \\ \text{where:} \quad A_t &= Q_\pi(s, u, w) - V_\pi(s, u) \\ Q_\pi(s, u, w) &= \mathbb{E}_\pi[R_t \mid s, u, w] \\ V_\pi(s, u) &= \mathbb{E}_\pi[R_t \mid s, u] \end{aligned}$$

Instead of maximizing expected reward, we maximize advantage, the difference of our state-action value function $Q_\pi(s, u, w)$, and a zero-expectation baseline, our state value function $V_\pi(s, u)$. Intuitively, this learns to prioritize update actions that result in better than average reward, and deprioritize actions resulting in worse than average reward.

Because the actual state-action value function Q_π is unknown, we can estimate it by using our discounted rewards R_t, giving us an estimate of advantage $\tilde{A}_t = R_t - V_\pi(s, u)$, where V_π is learned along with the policy π. However, Schulman et al. (2015) introduce a better estimation for Advantage called Generalized Advantage Estimation (GAE), which we utilize in this work. GAE allows us to control the weight of the state-value function estimate and the weight of recent actions with hyper-parameters, giving us more control over the learning process.

The final piece is the decomposition of our policy $\pi(w \mid s, u; \theta)$ into two independent policies over the functions f and arguments $\mathbf{a}$ (recall that $w = (f, \mathbf{a})$). Sharma et al. (2017) show that any compositional policy $\pi(f, \mathbf{a} \mid s, u)$ in an actor-critic model (like A2C/A3C) can be factored as the product of separate policies $\pi(f \mid s, u)$ and $\pi(\mathbf{a} \mid s, u)$.

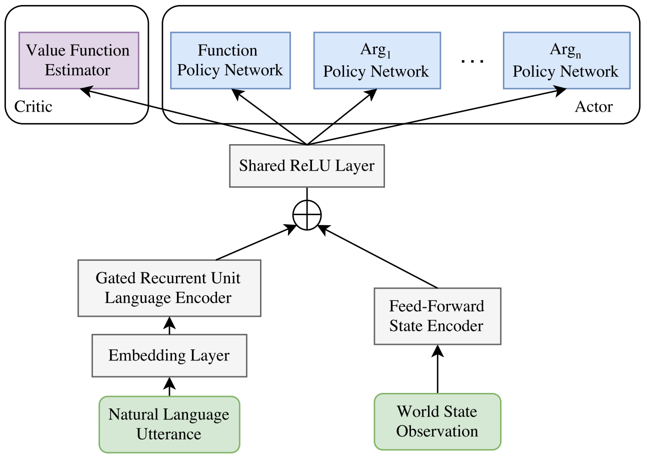


Figure 2: World Interaction Network Architecture.

More importantly, they demonstrate during train time, actions can be chosen by sampling independently from the separate sub-policies, often leading to faster training, and more generalizable behavior . With this in mind, our final Interaction Engine training objective is as follows:

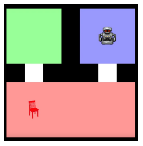
$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_\pi \big[\nabla_\theta \log \pi(f, \mathbf{a} \mid s, u; \theta) \big] \cdot \tilde{A}_t \\ &= \mathbb{E}_\pi \big[\nabla_\theta \log \big(\pi_f(f \mid s, u; \theta) \pi_a(\mathbf{a} \mid s, u; \theta) \big) \big] \cdot \tilde{A}_t \\ &= \mathbb{E}_\pi \big[(\nabla_{\theta_f} \log \pi_f) + (\nabla_{\theta_a} \log \pi_a) \big] \cdot \tilde{A}_t \end{aligned}$$

We present a variant of the A2C Algorithm for training this objective with the factored action space in Algorithm 1.

Interaction Engine Neural Architecture We implement our Interaction Engine with the neural network architecture shown in Figure 2. The Interaction Engine takes as input at every time step t the natural language utterance u_t, as well as a representation of the current world state s_t. The natural language utterance u_t is first embedded via an embedding layer, where the embeddings start off as random. The variable length utterance embeddings are then mapped into a fixed size vector with a Gated Recurrent Unit (Cho et al., 2014), a type of Recurrent Neural Network cell popular in many natural language processing tasks, like machine translation, question answering, as well as prior work in human-robot interaction (Cho et al., 2014; Karamcheti et al., 2017).

The state representation s_t can be represented arbitrarily, and encoded with a corresponding architecture. For example, if the state were represented as an image, a Convolutional Neural Network (CNN) architecture would be appropriate. However, here we assume that the state s_t is represented as a vector. We encode it via a two-layer feed-forward network, with the ReLU activation function. The encoded state and utterance are then concatenated, and fed to a shared hidden layer, with another ReLU. The resulting vector is then fed to multiple sub-networks, for generating the state-value function estimate V_t, as well as the function policy $\pi_f(f \mid s, u)$, and the argument policy $\pi_a(\mathbf{a} \mid s, u)$.

In the case of functions with multiple arguments, we allow for multiple argument sub-networks. Each sub-network


(a) Cleanup World

	Examples	Funcs	Args	Tasks
WeakSup	500	1	5	5/5
Goals	779	3	5	6/15
Actions	2955	4	7	15/28

(b) Cleanup World Dataset Statistics.

	Random	WIN	DRAGGN (UB)
WeakSup	20%	93.0%	93.0%
Goals	16.67%	84.8%	86.0%
Actions	6.67%	*71.9%*	95.9%

(c) Cleanup World Results. Results use model with highest validation accuracy across three random seeds.

Figure 3: Cleanup World Language Grounding Dataset Statistics and Results.

consists of a single hidden layer with a ReLU activation, followed by the output layer. In the case of the state-value estimate, this final layer has a linear activation, while each of the policy sub-networks have a final softmax activation, to generate probability distributions over functions/arguments. To regularize the network, we use Dropout (Srivastava et al., 2014). We train our network using the Adam Optimizer (Kingma and Ba, 2014), with a learning rate of .00001. All models are implemented in Tensorflow (Abadi et al., 2015).

Policy Training with Limited Annotated Examples In many scenarios, we may have access to a small number of fully annotated examples, where each utterance is annotated with the exact function/argument structured update tuple. Especially in scenarios where there are a large number of possible function/argument tuples, or where there are several utterances in a row without a reward signal, including this small amount of information can significantly learning.

To formalize this, we assume access to a small number of annotated examples (u, s, w), where $w = (f, \mathbf{a})$. With this, we now have a well-defined supervised learning objective – namely, maximize the probability of generating $f, \mathbf{a}$ from the given state s and utterance u. Equivalently, we minimize the sum of the cross-entropy losses between the predicted distributions $\pi(f \mid s, u)$, $\pi(\mathbf{a} \mid s, u)$ and the true labels $f, \mathbf{a}$.

At train time, we perform this training steps by initially pre-training the policy networks $\pi(f), \pi(\mathbf{a})$ to convergence on the given set of annotated examples. We then train the rest of the WIN via Reinforcement Learning, using the procedure outlined in Algorithm 1.

Experiments

We evaluate the WIN on two applications: understanding natural language instructions in a human-robot interaction setting, and question-answering on short stories. The first application shows that with even in cases of a single instruction, the WIN enables training with less supervision than prior approaches. The second application shows that the WIN is able to leverage the World Model effectively, synthesizing information from multiple sentences.

Cleanup World Language Grounding

The first set of experiments are conducted on the Cleanup World Mobile Manipulator Domain (MacGlashan et al., 2015), pictured in Figure 3a. The Cleanup World domain consists of a robot agent, one or more objects, and three rooms of different colors. Possible robot actions involve fine-grained actions, like moving some steps in a certain direction , to more general actions like having the agent move an object between rooms. We utilize datasets put forth by recent language grounding approaches. Statistics describing the datasets can be found in Figure 3b.

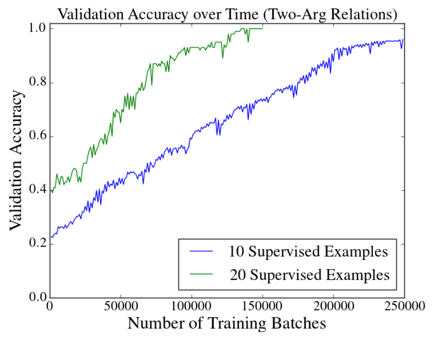
The first Cleanup World Experiment (*WeakSup*) utilizes the dataset introduced by Williams et al. (2017) in their work on weakly-supervised CCG Semantic Parsing. The second and third Cleanup World experiments (*Goals, Actions*) utilize the datasets presented by Karamcheti et al. (2017), for grounding action-oriented and goal-oriented language. These datasets consist of single natural language utterances annotated with the states of the world before and after grounding. Even though we only ground a single utterance before validation, this is still a critical task, as the instruction language is rather ambiguous and complex. A model that works in such a setting would benefit the human-robot interaction community, as this weakly supervised data is easier to collect, while also being transferable to other representations. To validate the structured representation output by our WIN, we use the procedure developed by Williams et al. (2017) that returns $+1$ reward if the given structured representation results in the desired final state, and 0 otherwise. Full details about how we utilized these datasets in our experiments can be found in the supplemental material.

To fairly evaluate the WIN, we limit ourselves to baselines that utilize the same function/argument representation (as the complexity of the structured representation significantly affects grounding performance). As such, in addition to WIN results, we present the DRAGGN results on each of the datasets, as an upper bound (UB). The DRAGGN model (Karamcheti et al., 2017) gets strictly more data than the WIN as it is trained in a fully supervised setting. In this way, it provides a meaningful upper bound, as it is given the ground truth groundings that the WIN is trying to learn.

Results: Table 3c has the WIN and DRAGGN upper bound results for the three different Cleanup World experiments. On the Weakly Supervised dataset (*WeakSup*), the WIN is able to perfectly match the upper bound performance, obtaining a near perfect 93.0% grounding accuracy, given the small number of examples. We note that this is especially interesting, as the validation function the WIN is provided with contains strictly less information than the fully annotated utterances the DRAGGN model is provided with. On the Goal-Oriented language corpus (*Goals*), the WIN is able to achieve close to the DRAGGN performance, obtaining a final grounding accuracy of 84.8% (compared to

	LSTM (LB)	MemN2N (LB)	WIN + 10	WIN + 20	DRAGGN (UB)	MemNN (UB)
1 Supporting Fact	50.0	100.0	**100.0**	100.0	100.0	100.0
2 Supporting Facts	20.0	91.7	**96.9**	99.8	99.8	100.0
3 Supporting Facts	20.0	59.7	**96.7**	96.7	96.7	100.0
2 Arg Relations	61.0	97.2	*95.8*	**100.0**	100.0	100.0
3 Arg Relations	70.0	86.9	*53.7*	*85.7*	*85.8*	98.0
Yes/No Questions	48.0	92.4	**98.3**	100.0	100.0	100.0
Counting	49.0	82.7	83.1	**95.6**	99.8	85.0
Lists/Sets	45.0	90.0	*57.1*	**96.5**	100.0	100.0
Simple Negation	64.0	86.8	92.1	**100.0**	100.0	100.0
Indefinite Knowledge	44.0	84.9	90.7	**95.1**	99.2	98.0
Basic Coreference	62.0	99.1	**99.5**	100.0	100.0	100.0
Conjunction	74.0	99.8	**100.0**	100.0	100.0	100.0
Compound Coreference	94.0	99.6	*93.0*	*95.8*	100.0	100.0
Time Reasoning	27.0	98.3	*92.3*	**100.0**	100.0	99.0
Basic Deduction	21.0	100.0	**100.0**	100.0	100.0	100.0
Basic Induction	23.0	98.4	**100.0**	100.0	100.0	100.0
Positional Reasoning	51.0	49.0	51.9	69.0	90.1	65.0
Size Reasoning	52.0	88.9	**96.6**	99.6	99.6	95.0
Path Finding	8.0	17.2	79.7	**100.0**	100.0	36.0
Agent Motivation	91.0	100.0	**100.0**	100.0	100.0	100.0
Solved Tasks ($> 95\%$)	0	8	11	**18**	18	17

(a) Hybrid bAbI Results. WIN Models are trained via RL, with 10 (1%), and 20 (2%) of the stories fully annotated. *Italics* denote cases where WIN results are worse than MemN2N, while **Bold** denotes best WIN model that solves the task ($> 95\%$ accuracy).



(b) Learning curves plotting accuracy on validation set vs. number of training iterations for the Two-Arg Relations Hybrid bAbI Task.

Figure 4: Hybrid bAbI Tasks Results.

the maximum possible 86.0%). This dataset contains a task output space slightly larger than the previous Weakly Supervised dataset, and contains significantly more complex language. Again, we note the WIN's ability to match DRAGGN performance given significantly less information. However, on the Action-Oriented corpus (*Actions*), we see that the WIN has performance that fails to match DRAGGN, by a large margin (71.9% accuracy vs. 95.9% accuracy). Upon thorough error analysis, we found this was due to mode collapse, as the original dataset is very imbalanced (of the 2955 examples, 2264 all have the same argument label). We note this as an interesting failure mode, and hope to perform more experiments in future work.

Hybrid bAbI Grounding and Question-Answering

The second set of experiments are conducted on a new dataset, based on the 20 well-known synthetic bAbI Question-Answering Tasks (Weston et al., 2015). In the original bAbI dataset, each task consists of 1000 story, question, and answer tuples, where stories consist of several sentences (for our work, all stories are truncated to be no longer than 20 sentences). Note that we utilize the 1000 (1k) version of the dataset, rather than the 10,000 example (10k) version utilized by most recent work (Henaff et al., 2016; Santoro et al., 2017). We do this to show that WIN models can be effective given small amounts of data, unlike the cited approaches. The tasks test varying skills, from tracking relations between multiple entities, to the ability to deduce or induce missing information, to the ability to reason about size, position, or geographical location. Because of this wide variety of well-defined tasks, the bAbI dataset makes a good testbed for the evaluation of language grounding models.

The full procedure used to convert the original bAbI dataset into the Hybrid dataset can be found in the supplemental material. We find that we can represent a given task's world updates with functions from up to 4 classes, and up to three arguments from up to 14 classes. Stories are limited to be 20 sentences long, which means that in many cases, the WIN has to choose world updates from the set of possible functions and arguments over a horizon of 10-20 separate examples without a reward signal. From this, it is impractical to assume that a random policy will learn to properly ground such stories without a significant amount of train time. However, we find that if we start with a small number of annotated examples, we can significantly accelerate learning. As such, we test two versions of the WIN, one where it is provided with 10 fully annotated stories (only 1% of the total dataset), and another where it is provided with 20 stories (only 2%). The full Hybrid bAbI QA/Language Grounding Dataset annotated with the full set of function/argument tuples can be found at the following URL: `https://sites.google.com/site/winsupplemental/`.

We implement the World Model as a state machine (implemented in code) that consumes world updates and updates its internal fields in a pure and consistent manner. Some example updates include agentToRoom(entity, location) that moves a specified person to the given location, link(entity, object), for picking up an object, and isA(entity, property) for assigning characteristics to an entity. Question-Answering is done on the final state, after all updates have been processed. Any reasoning that is necessary for answering the question is also handled by the World Model. For example, if we have a question like "How many objects is Mary carrying," we have a function (in code) that explicitly counts the objects that have been linked to Mary by the set of world updates, rather than having to track the quantity implicitly, as is done in existing methods. As mentioned previously, this ability to use external tools to perform reasoning is a strong benefit of our approach.

To put the WIN results in context, we present a series of lower and upper bounds. Again, this is because any strict evaluation is based on the complexity of the underlying update representation. We present two lower bounds (LB), each of which receive strictly less information than the WIN. These neural models only receive story, question, answer tuples, and are trained end-to-end. The first lower bound we present is a simple LSTM approach, while the second lower bound model is an End-to-End Memory Network (MemN2N) (Sukhbaatar et al., 2015) that explicitly reads the story via multiple passes, then finally generates the answer. The two upper bounds receive strictly more information than the WIN. The first upper bound, the DRAGGN, receives fully annotated stories. We note that this is the most meaningful model for interpreting the WIN results, as it directly bounds the results in the same manner as in the prior experiments. The second upper bound is the Memory Network with Supporting Fact Supervision (MemNN) (Weston, Chopra, and Bordes, 2014). The Memory Network gets the original sentences of the story, along with the set of sentences necessary to answering a question. Because the subset of sentences necessary to answer are significantly smaller than the total number of sentences in the story, the MemNN model has the best performance on the original dataset at the cost of needing the most annotated data.

Results: Table 4a has the results for both versions of the WIN, as well as all lower and upper bounds. Notably, we show that providing just 10 annotated examples is enough to gain greater than 95% accuracy on 11 of the 20 tasks. If we increase the number to 20 annotated stories, we solve 18 of the hybrid tasks. In other words, solely by providing the World Model, and a very small number of annotated examples (2% of the total training set), we can obtain near-perfect performance. Figure 4b shows learning curves on the "2 Arg Relations" Task, one of the only tasks requiring more than 20,000 training steps. Note that with more annotated examples, the WIN learns faster and more efficiently.

Discussion

Our evaluation shows that not only is the WIN able to successfully operate in the context of human-robot interaction, as per our Cleanup World experiments, but also in the context of story understanding, solving 18 out of the 20 Hybrid bAbI question tasks.

On Cleanup World, we find that most of the time, the WIN is able to match (or almost match) the upper bound DRAGGN state-of-the-art results. This seems to strongly indicate the effectiveness of the WIN, as it takes minor performance hits, with the major benefit of requiring less data to train. Not only is it significantly easier to collect data of the form required for the WIN (namely utterance, initial state, and final state), but it allows for the use of many different types of structured representations with the same dataset. In the case of the fully supervised DRAGGN models, the language utterances all must be annotated with the required function/argument pairs, which is not only extremely expensive, but not robust to representational changes.

While the results of the Cleanup World experiments are strong, the crucial benefits of the WIN are found in the Hybrid bAbI results. Most notable are the cases where the WIN is able to significantly outperform the End-to-End Memory Network lower bound, for example, on the tasks of "Counting", "Indefinite Knowledge", and "Path Finding". The key similarity between each of these three tasks is the need for relatively complex reasoning and deduction to come up with the answer given the story. For example, in the "Path Finding" task, stories consist of descriptions of the world (i.e. "The kitchen is north of the hallway, the garden is east of the hallway"), and questions asking to find paths between certain locations (i.e. "How do I get from the kitchen to the garden?"). This requires complex reasoning to go from the story information describing the world, to the required path (i.e. "south, east"). In the case of the End-to-End Memory Network, because this reasoning is happening internal to the neural network model, it is hard to generate the correct answer, as the network is not structured to perform the explicit logical steps. However, in the case of the WIN, because the World Model is specified externally (namely, in code), any set of tools (in our case, a simple search algorithm) can be leveraged to find the given path from the story information.

Also important are the cases where the WIN fails to outperform the End-to-End Memory Network lower bound, namely, on the "3 Arg Relations" and "Compound Coreference" Tasks. We note that the key similarity between these tasks is that they all require world updates that are represented with a single function, and exactly three arguments (the maximum number of arguments we needed across all tasks). For the "3 Arg Relations" task, this amounts to a total number of world updates equal to $4 \cdot 5 \cdot 14 \cdot 14 = 3,920$ possible options. While the WIN is clearly able to learn how to map utterances to the correct world state, it is not able to do so efficiently, as in some sense, it is trying to solve a harder problem than is necessary (both these tasks have a large amount of unnecessary information in the stories). Because the End-to-End Memory Network is given the question a priori, it can effectively search over the entire story and attend to only the important parts, while the WIN has to ground every utterance with high accuracy, which in these cases prove to be a harder task.

In other words, because the WIN factors out the World Model, we can not only effectively ground language for human-robot interaction, but also answer questions for text-based story understanding. The external model allows us to not only fully interpret and understand the grounding procedure, but also lets us use a wide variety of specialized tools and algorithms for performing reasoning and deduction, something that would be impossible with an end-to-end model. Best of all, we can obtain strong results with the WIN, either matching or exceeding the existing state-of-the-art, with significantly less data and supervision.

While the external World Model can clearly lead to significant benefits, it is limiting because it cannot be directly learned. Not only does the World Model need to be expertly designed, but the process of adding additional literals or actions may require non-trivial tweaks. As such, scaling to new domains would necessitate an updated World Model, capable of handling new update actions. However, we argue that the interpretability, transparency, and ability to perform structured reasoning provided by the World Models exceed the cost, especially in the use-cases we have outlined above. We hope to address some new applications of the WIN, as well as some of the aforementioned problems in future work.

Conclusion and Future Work

We present the World Interaction Network (WIN), a framework for mapping language to structured world updates. We develop the WIN using a deep neural network to map language utterances and state observations into a representation based on functions and arguments. We also present a novel reinforcement learning algorithm based on Advantage Actor-Critic (A2C) for training the WIN, where the only inputs are the initial state of the world, the set of utterances to ground, and a final validation constraint to run on the final state. Finally, our results show that we can use the WIN to great effect in applications to tasks in human-robot interaction, as well as text-based question answering. On the latter, the WIN grants full transparency into the grounding process, allowing for the use of external tools to perform structured reasoning, solving 18 of the 20 Hybrid bAbI Tasks.

While the WIN is a step in the right direction, there remain open challenges. One challenge would be to learn to compose primitive functions to represent nested behaviors, without the need for additional hand-engineered forms. These "macro-functions" would allow for the generalization to more tasks, with less information. Another possible direction would be to explore the capabilities of the external world model. Because you can use arbitrary tools in tandem with the world model, it would be interesting to see the possibilities of the WIN framework in combination with an external database, to do more general question answering.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I. J.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Józefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D. G.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P. A.; Vanhoucke, V.; Vasudevan, V.; Viégas, F. B.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR* abs/1603.04467.

Artzi, Y., and Zettlemoyer, L. 2013. Weakly supervized learning of semantic parsers for mapping instructions to actions. In *Annual Meeting of the Association for Computational Linguistics*.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Guu, K.; Pasupat, P.; Liu, E. Z.; and Liang, P. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *CoRR* abs/1704.07926.

Henaff, M.; Weston, J.; Szlam, A.; Bordes, A.; and LeCun, Y. 2016. Tracking the world state with recurrent entity networks. *CoRR* abs/1612.03969.

Karamcheti, S.; Williams, E. C.; Arumugam, D.; Rhee, M.; Gopalan, N.; Wong, L. L. S.; and Tellex, S. 2017. A tale of two draggns: A hybrid approach for interpreting action-oriented and goal-oriented instructions. *CoRR* abs/1707.08668.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

MacGlashan, J.; Babeş-Vroman, M.; DesJardins, M.; Littman, M. L.; Muresan, S.; Squire, S.; Tellex, S.; Arumugam, D.; and Yang, L. 2015. Grounding English commands to reward functions. In *Robotics: Science and Systems*.

Misra, D.; Langford, J.; and Artzi, Y. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *ICML*.

Santoro, A.; Raposo, D.; Barrett, D. G. T.; Malinowski, M.; Pascanu, R.; Battaglia, P. W.; and Lillicrap, T. P. 2017. A simple neural network module for relational reasoning. *CoRR* abs/1706.01427.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *CoRR* abs/1506.02438.

Sharma, S.; Suresh, A.; Ramesh, R.; and Ravindran, B. 2017. Learning to factor policies and action-value functions: Factored action space representations for deep reinforcement learning. *CoRR* abs/1705.07269.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Sukhbaatar, S.; Szlam, A.; Weston, J.; and Fergus, R. 2015. End-to-end memory networks. In *NIPS*.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

Weston, J.; Bordes, A.; Chopra, S.; and Mikolov, T. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR* abs/1502.05698.

Weston, J.; Chopra, S.; and Bordes, A. 2014. Memory networks. *CoRR* abs/1410.3916.

Williams, E. C.; Rhee, M.; Gopalan, N.; and Tellex, S. 2017. Learning to parse natural language to grounded reward functions with weak supervision. In *AAAI Fall Symposium on Natural Communication for Human-Robot Collaboration*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.

Supplemental Material

Cleanup World Language Grounding

The first Cleanup World Experiment (*WeakSup*) utilizes the dataset introduced by Williams et al. (2017) in their work on weakly-supervised CCG Semantic Parsing. The dataset presented is small, containing only 500 examples, spanning 5 different tasks. The tasks all involve the robot agent moving to a specific location, either a room or an object. Each example consists of a single utterance, a set of different pre-condition states of the world prior to grounding, as well as a set of post-condition states that are all satisfiable given the correct grounding and respective pre-condition state. Williams et al. (2017) additionally define a validation function that outputs 1 or 0 based on if the produced CCG parse satisfies all post-condition states given each pre-condition state of the example in question.

To apply the WIN to this dataset, we simplify the lambda calculus parse language utilized in the original work into a language of functions and arguments, identical to the representation used by Karamcheti et al. (2017). This structured representation consists of a single function, and five different arguments. We additionally modify the presented validation function to take in the modified functional language. As such, the WIN setup involves feeding in the utterance and initial states, generating the function/argument tuple, and then calculating the reward (via the validation function), based on if the respective update satisfies each post-condition state of the example.

The second and third Cleanup World experiments (*Goals, Actions*) utilize the datasets presented by Karamcheti et al. (2017), in their work on grounding action-oriented and goal-oriented language. The authors present two datasets, one consisting of goal-oriented language, and the other consisting of action-oriented language. Because Karamcheti et al. (2017) work in a fully supervised context, each example consists of the utterance, the initial state of the world, and the correct function/argument tuple to generate. To turn this into a weakly supervised problem, such that we can illustrate the effectiveness of the WIN, we create a validation function similar to that of Williams et al. (2017), by executing each function/argument tuple to get the final post-condition state of the world. We then discard the ground truth function/arguments that are supposed to be generated, and train our WIN solely using the raw utterances, states, and validation function.

Training Details: For all Cleanup World experiments, we train the WIN only via Reinforcement Learning, for 1600 iterations of Algorithm 1. The initial states of the Cleanup World domain are represented as hot-encoded vectors that store the location of the robot agent. We use an embedding size of 30, an RNN encoder size of 128, processing 8 separate utterances per forward pass. Results are obtained by applying the model with the highest validation set accuracy, over three random seeds.

Hybrid bAbI Grounding and Question-Answering

To convert the original bAbI Question-Answering dataset into the Hybrid bAbI Question-Answering/Language Grounding dataset, we first define a set of functions and arguments that encapsulate the full set of world updates that are represented for each of the 20 tasks. We find that on average, each task's set of world updates can be represented with around a function from up to 4 classes, and up to three arguments from up to 14 classes. We then modify the original code used to generate the tasks (Weston et al., 2015), to annotate each sentence of the original bAbI QA tasks (version 1.2) with the corresponding world state update (function/argument tuple). Note that the stories in our dataset are exactly the same as those in the original (we are not generating new stories). To generate the final validation constraint, we turn each of the question/answer pairs from the original dataset into a boolean function, that returns 1 reward if True given the final state of the world, and 0 reward otherwise. More precisely, if we are given a question like "Where is Mary?" with the answer "kitchen", our validation function returns 1 reward if and only if in the final world state, Mary ends up in the kitchen. As mentioned in the paper, the full Hybrid bAbI QA/Language Grounding Dataset annotated with the full set of function/argument tuples can be found at the following URL: `https://sites.google.com/site/winsupplemental/`.

We implement the World Model as a state machine (implemented in code) that consumes a given world update and updates its internal fields in a pure and consistent manner. For example, given an utterance like "Mary went to the kitchen" which maps to the function/arguments agentToRoom(Mary, kitchen), the state machine merely updates its internal dictionary storing locations to have the key "Mary" point to the location "kitchen." This is a simple procedure that is not only efficient at modeling the full state of the world, but is effective to query at validation time. Any reasoning that needs to happen for validation is also executed in code, which is a crucial benefit of our approach. For example, if we have a question like "How many objects is Mary carrying," we have a function (in code) that explicitly counts the objects that have been linked to Mary by the given set of world updates, rather than having to track the quantity implicitly, as is done in existing end-to-end methods.

We specifically use such reasoning to great effect in the following reasoning-focused tasks: "Counting", "Lists/Sets", "Indefinite Knowledge", "Basic Deduction", "Basic Induction", and "Path Finding." In the "Path Finding" task specifically, in which we report significantly better results than the lower bound model, we found this reasoning to be extremely important. More precisely, we used a simple dynamic programming algorithm to take stories (of the form "The garden is east of the hallway," "The kitchen is north of the hallway") and questions (of the form "How do I get from the kitchen to the garden") and compute the actual path necessary for the answer (namely, "south, east"). Having this algorithm responsible for the structured reasoning allowed for perfect performance on this task, whereas all end-to-end methods failed.

Tables 1 and 2 (next page) give exhaustive dataset statistics, and provide the full list of all world update functions utilized for the Hybrid bAbI Tasks.

Training Details: For all bAbI experiments, we train the WIN via the hybrid supervised learning/reinforcement learning procedure outlined in the Approach section, for up to 250,000 iterations of Algorithm 1, with early stopping after 20,000 iterations if the accuracy on the validation set exceeds 95% (we find only a small number of tasks need more than 20,000 iterations). We perform an initial pre-training of 120 supervised iterations on the limited number of annotated examples. Initial states are represented by hot encoding any information relevant to the task (i.e. entity locations in the entity relation tasks, presence of objects in reasoning tasks) in a vector. To further improve the stability of the training procedure, every several thousand iterations of the reinforcement learning procedure, we run a very small number of supervised training steps on the annotated examples. We find empirically that this helps prevent catastrophic forgetting. To this end, we perform 20 supervised iterations on the limited number of annotated examples every 2,000 reinforcement learning iterations. We use an embedding size of 30, with an RNN size of 128, processing 8 separate utterances per forward pass. Results are obtained by applying the model with the highest validation set accuracy over three random seeds.

	Max Story Length	Functions	Argument 1	Argument 2	Argument 3
1 Supporting Fact	10	1	5	7	-
2 Supporting Facts	20	3	5	10	-
3 Supporting Facts	20	3	5	10	-
2 Arg Relations	2	4	7	7	-
3 Arg Relations	20	4	5	14	14
Yes/No Questions	20	3	5	10	-
Counting	20	4	5	14	-
Lists/Sets	20	3	5	10	-
Simple Negation	10	1	5	7	-
Indefinite Knowledge	10	2	5	7	7
Basic Coreference	10	1	7	7	-
Conjunction	10	1	7	5	5
Compound Coreference	10	2	7	6	5
Time Reasoning	14	4	5	7	-
Basic Deduction	8	2	9	5	-
Basic Induction	9	2	6	9	-
Positional Reasoning	2	4	7	7	-
Size Reasoning	15	2	7	7	-
Path Finding	5	4	7	7	-
Agent Motivation	12	3	5	12	-

Table 1: Dataset Statistics for the Hybrid bAbI Tasks Dataset

Function Signature	Description
agentToRoom(entity, location)	Move an entity to the given location.
link(entity, object)	Link an object to an entity (for picking up objects).
unlink(entity, object)	Unlink an object and entity (for dropping objects).
isNorthOf(room1, room2)	Specify that room1 is north of room2 (positional reasoning).
isEastOf(room1, room2)	Specify that room1 is east of room2 (positional reasoning).
isWestOf(room1, room2)	Specify that room1 is west of room2 (positional reasoning).
isSouthOf(room1, room2)	Specify that room1 is south of room2 (positional reasoning).
transfer(person1, person2, object)	Transfer object from person1 to person2.
addMotive(person, motive)	Add motivation (hungry, thirsty, etc.) to given person.
fitsInside(object1, object2)	Specify that object1 fits inside object2 (size reasoning).
isBiggerThan(object1, object2)	Specify that object1 is bigger than object2 (size reasoning).
isA(entity, characteristic)	Assign given characteristic to entity.
isColor(entity, color)	Assign given color to entity.
isAfraidOf(entity1, entity2)	Specify that entity1 fears entity2.
agentToRoomYesterday(entity, location)	Specify that entity went to location yesterday.
agentToRoomMorning(entity, location)	Specify that entity went to location this morning.
agentToRoomAfternoon(entity, location)	Specify that entity went to location this afternoon.
agentToRoomEvening(entity, location)	Specify that entity went to location this evening.
agentsToRoom(entity1, entity2, location)	Move entity1 and entity2 to location.
agentsMaybeIn(entity1, entity2, location)	Specify that entities might be in location.

Table 2: Exhaustive Set of Functions defined for Hybrid bAbI Tasks. Corresponding updates implemented in World Model.