ScienceWorld: Is your Agent Smarter than a 5th Grader?

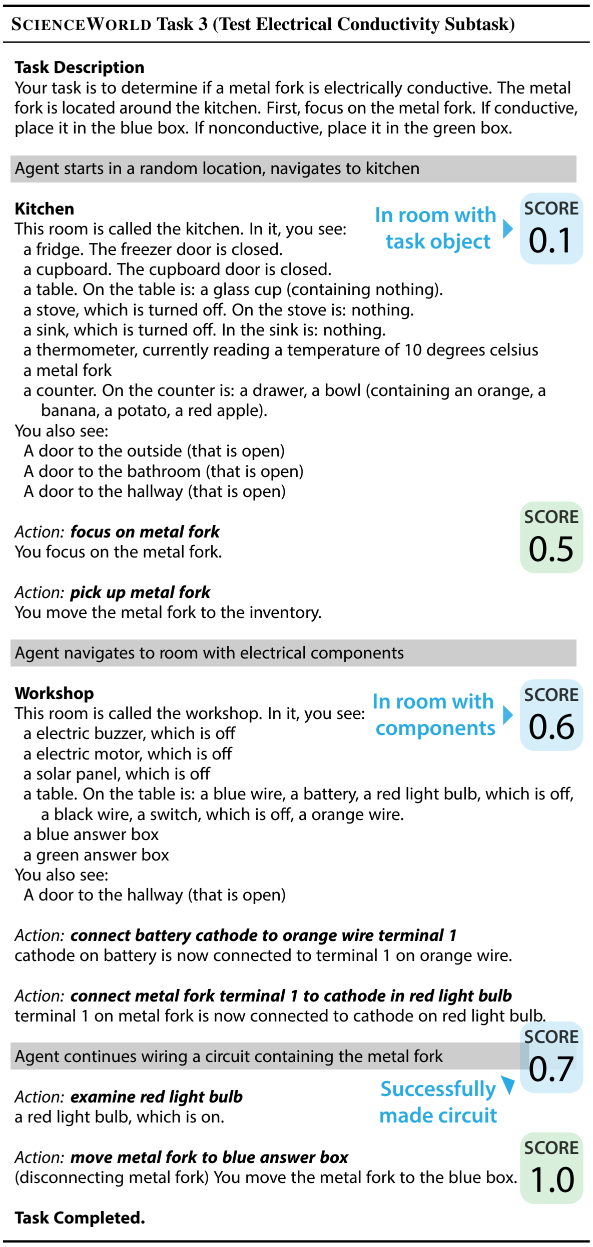
Ruoyao Wang[*,‡], **Peter Jansen**[*,‡], **Marc-Alexandre Côté**[♣], **Prithviraj Ammanabrolu**[◇]
[‡]University of Arizona, Tucson, AZ [♣]Microsoft Research Montréal
[◇]Allen Institute for AI, Seattle, WA
{ruoyaowang,pajansen}@arizona.edu
macote@microsoft.com, raja@allenai.org

Abstract

We present ScienceWorld, a benchmark to test agents' scientific reasoning abilities in a new interactive text environment at the level of a standard elementary school science curriculum. Despite the transformer-based progress seen in question-answering and scientific text processing, we find that current models cannot reason about or explain learned science concepts in novel contexts. For instance, models can easily answer *what* the conductivity of a known material is but struggle when asked *how* they would conduct an experiment in a grounded environment to find the conductivity of an unknown material. This begs the question of whether current models are simply retrieving answers by way of seeing a large number of similar examples or if they have learned to reason about concepts in a reusable manner. We hypothesize that agents need to be grounded in interactive environments to achieve such reasoning capabilities. Our experiments provide empirical evidence supporting this hypothesis—showing that a 1.5 million parameter agent trained interactively for 100k steps outperforms a 11 billion parameter model statically trained for scientific question-answering and reasoning from millions of expert demonstrations.[1][2]

1 Introduction

Question answering (QA) has seen rapid progress recently. Standardized elementary and middle school science exams have served as a challenge task for QA (Clark et al., 2018), as these questions require combining science-domain knowledge with world knowledge in complex reasoning procedures to solve. As large language models have toppled these benchmarks (Clark et al., 2020; Khashabi et al., 2020; Xu et al., 2021a), the focus has shifted away from simply answering questions toward producing human-readable explanations for a model's



ScienceWorld Task 3 (Test Electrical Conductivity Subtask)

Task Description
Your task is to determine if a metal fork is electrically conductive. The metal fork is located around the kitchen. First, focus on the metal fork. If conductive, place it in the blue box. If nonconductive, place it in the green box.

Agent starts in a random location, navigates to kitchen

Kitchen
This room is called the kitchen. In it, you see:
a fridge. The freezer door is closed.
a cupboard. The cupboard door is closed.
a table. On the table is: a glass cup (containing nothing).
a stove, which is turned off. On the stove is: nothing.
a sink, which is turned off. In the sink is: nothing.
a thermometer, currently reading a temperature of 10 degrees celsius
a metal fork
a counter. On the counter is: a drawer, a bowl (containing an orange, a banana, a potato, a red apple).
You also see:
A door to the outside (that is open)
A door to the bathroom (that is open)
A door to the hallway (that is open)

In room with task object
SCORE 0.1

Action: ***focus on metal fork***
You focus on the metal fork.

SCORE 0.5

Action: ***pick up metal fork***
You move the metal fork to the inventory.

Agent navigates to room with electrical components

Workshop
This room is called the workshop. In it, you see:
a electric buzzer, which is off
a electric motor, which is off
a solar panel, which is off
a table. On the table is: a blue wire, a battery, a red light bulb, which is off, a black wire, a switch, which is off, a orange wire.
a blue answer box
a green answer box
You also see:
A door to the hallway (that is open)

In room with components
SCORE 0.6

Action: ***connect battery cathode to orange wire terminal 1***
cathode on battery is now connected to terminal 1 on orange wire.

Action: ***connect metal fork terminal 1 to cathode in red light bulb***
terminal 1 on metal fork is now connected to cathode on red light bulb.

Agent continues wiring a circuit containing the metal fork

SCORE 0.7

Action: ***examine red light bulb***
a red light bulb, which is on.

Successfully made circuit

Action: ***move metal fork to blue answer box***
(disconnecting metal fork) You move the metal fork to the blue box.

SCORE 1.0

Task Completed.

Table 1: Transcript for one of the 30 tasks (testing an object's electrical conductivity, here a *metal fork*) in ScienceWorld. The agent finds the *metal fork*, use it to build an electrical circuit with a *red light bulb*, and observe if the light turns on. Note the transcript has been simplified to fit in available space, and large sections have been omitted (greyed sections).

---

[1] Website: https://sciworld.apps.allenai.org
[2] Code: https://github.com/allenai/ScienceWorld

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 11279–11298
December 7-11, 2022 ©2022 Association for Computational Linguistics

answers (Jansen et al., 2018; Yang et al., 2018; Xie et al., 2020; Valentino et al., 2021; Xu et al., 2021b; Lamm et al., 2021; Aggarwal et al., 2021).

While language models are able to produce compelling answers (Zoph et al., 2022) or explanations (Jansen et al., 2021) to science questions, are they simply retrieving (or shallowly assembling) these answers, or can they understand and use the knowledge they output in a meaningful way? Also, how can we evaluate if a model's explanation is correct?

In this work we explore these two questions by reframing science exam question answering into an interactive task where agents must complete elementary science experiments in a simulated text-based environment called SCIENCEWORLD (see Table 1). Instead of simply answering a question *(e.g., Q: "What will happen to an ice cube when placed on a stove?", A: "it will melt")*, the agent must demonstrate its capacity to combine declarative scientific knowledge with the procedural knowledge required to correctly complete the experiment in the virtual environment. Similarly, the sequence of virtual actions an agent performs can serve as a form of procedural *("how")* explanation to the question, that can be directly evaluated in the virtual environment for correctness (e.g., whether the actions led to the ice cube melting).

The contributions of this work are:

1. We construct SCIENCEWORLD, a complex interactive text environment, with simulation engines for thermodynamics, electrical circuits, chemistry reactions, and biological processes.

2. We implement 30 benchmark tasks across 10 topics spanning the elementary science curriculum, including changes of state of matter and the role of pollinators when growing fruit.

3. We evaluate 5 state-of-the-art reinforcement learning and language model agents on this benchmark, empirically showing that they perform poorly on tasks (e.g., melting ice) that 5th grade students can perform with ease.

2 Related Work

Science-domain Inference: Standardized science exams are a challenging task for question answering due to their diverse knowledge and inference requirements (Clark et al., 2013; Jansen et al., 2016; Boratko et al., 2018; Clark et al., 2018). Top performing models can answer more than 90% of multiple-choice questions correctly (Clark et al., 2020), typically through the use of large language models (Khashabi et al., 2020; Zoph et al., 2022).

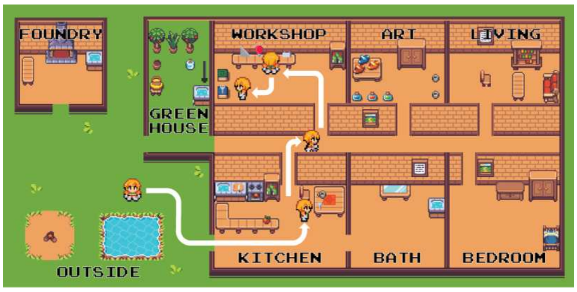


Figure 1: A graphical representation of an agent performing the Electrical Conductivity Subtask in SCIENCEWORLD.

A number of corpora of structured and semi-structured science exam explanations exist for training the explanation-generation task (Jansen et al., 2018; Xie et al., 2020; Khot et al., 2020; Dalvi et al., 2021). Evaluating explanations is challenging, and typically done by comparing generated explanations to a single gold explanation. This has been shown to substantially underestimate explanation generation performance by up to 40% (Jansen et al., 2021). While others have worked to mitigate this by generating multiple alternate explanations for each question, this is expensive, and has only been demonstrated for adding one or two additional explanations per question (Inoue et al., 2020; Jhamtani and Clark, 2020). Here, we propose a partial solution to this evaluation problem by treating the action sequences of agents as structured *manner* explanations for *how* to solve a task. These action sequences can be directly run in the SCIENCEWORLD simulator to automatically determine their correctness (i.e., whether they accomplish the task), independent of any variations in their solution methods. For example, whether an agent melts ice by using a stove, building a campfire, or leaving the ice out on a kitchen counter, the end result is the same and directly measurable through the formal semantics of the simulator.

Environments: Interactive text environments are becoming a vehicle for research in natural language processing (see Jansen (2021) for a review), primarily because of their reduced development costs relative to 3D environments, combined with their ability to easily implement high-level tasks with large action spaces. In spite of these benefits, implementation costs can still be substantial for large complex environments, and text agents are frequently evaluated on a small set of existing interactive fiction games such as Zork (Lebling et al., 1979) using an unified interface like Jericho (Hausknecht et al., 2020). A few purpose-built environments provide simple tasks for studying text agents, typically on procedurally generated pick-and-place or object-combination tasks (e.g., cooking, Yin and May, 2019). Kitchen Cleanup (Murugesan et al., 2020b) and TextWorld Common Sense (Murugesan et al., 2020a) require agents to tidy up one or more rooms in a house environment by putting objects in their typical locations (e.g., a *hat* should be placed on the *hat rack*), testing an agent's declarative knowledge of common object locations with the procedural knowledge required for this pick-and-place task. The closest existing interactive text environment to SCIENCEWORLD is TextLabs (Tamari et al., 2021) which simulates chemistry wet-lab protocols with actions such as *pipetting* and *centrifuging*. Compared to these existing environments, SCIENCEWORLD is generally a larger and more dynamic environment, populated with more complex objects with greater depth of physical simulation. This simulation depth enables more complex tasks associated with elementary science (e.g., thermodynamics, electrical circuits, etc.) to be tested, and a greater variety of solutions.

Simulators: Nearly all text-based world simulations are currently implemented as Z-machine games (Infocom, 1989; Nelson, 2014), frequently through higher-level application-specific languages (such as Inform7, Nelson, 2006) that compile to Z-machine code. TextWorld (Côté et al., 2018) creates environments using linear-logic statements that specify action preconditions and postconditions (Martens, 2015) and generate Inform7 code. Existing tooling is designed for simpler simulations than SCIENCEWORLD, with primarily agent-centered state changes that make modeling autonomous physical processes (e.g., thermodynamics) difficult. As such, in this work we build a novel simulator to model physical processes in text environments.

Agents: A variety of agent models have been proposed for reasoning in interactive text environments. Most approaches frame reasoning as a partially-observable Markov decision process (POMDP), and model inference using reinforcement learning (RL). This includes RL-based models that learn a policies to pick relevant actions from lists of candidate actions (He et al., 2016), or models that mix RL with knowledge graphs (Ammanabrolu and Hausknecht, 2020) or language models (Yao et al., 2020) to aid in next-action selection. Action selection has also been modelled using case-based reasoning (Atzeni et al., 2021), or directly as a sequence-prediction imitation learning task, using language models trained on gold action sequences to predict the next action given the current state (Torabi et al., 2018; Ammanabrolu et al., 2021, 2022). In general, agent performance on solving interactive fictions is still modest, with only easier games close to completion (Jansen, 2021).

In this work, we benchmark state-of-the-art agents on SCIENCEWORLD as well as introduce novel agents. We empirically show that these elementary science tasks are difficult for current agents, and also that smaller and simpler agents can outperform billion-scale parameter language models trained on gold sequences, highlighting the difficulty of this task for transformer-based models.

3 SCIENCEWORLD

SCIENCEWORLD is a simulation of the world abstracted through a complex interactive text environment in English with many objects, actions, and simulation engines. The framework consists of 40k lines of SCALA (speed) with a PYTHON interface.

The SCIENCEWORLD environment contains 10 interconnected locations (see Figure 1), populated with up to 200 types of objects, including devices, instruments, plants/animals, electrical components, substances, containers, and common environment objects such as furniture, books, and paintings. The SCIENCEWORLD action space contains 25 high-level actions, including both science-domain actions (e.g., using `thermometer`) and common actions (e.g., moving, opening containers, picking up items), with approximately 200k possible action-object combinations per step (though only a limited subset of these will be meaningful). See Appendix A for details about SCIENCEWORLD, including the object model, actions, and input parser.

3.1 Simulation Engines

SCIENCEWORLD supports actions commonly found in interactive text environments – for example, objects can be moved or examined, foods can be eaten, and books can be read. In addition, the environment contains a number of elementary science-domain specific processes that either occur automatically (e.g., thermodynamics) or are coupled to actions (e.g., devices, mixing chemicals).

Those simulation engines[3] are:

Thermodynamics: All objects have temperatures and other thermal properties based on their materials. All objects within a container are considered in thermal contact with each other, and transfer heat energy using a simplified conductive heat model. The proportion of heat transferred between objects at each step is mediated by the object's *thermal conduction coefficient*, allowing thermal conductors (like metal pots) and insulators (like ceramics) to be modelled. Every material has phase transition points *(i.e., melting point, boiling point)* and *combustion points* populated based on the best-known or approximate physical values for those materials. Objects that move past these thresholds will change state of matter (e.g., from a solid to a liquid), or begin a combustion process that ultimately ends in the object turning to ash unless its fire is put out. Convective heat transfer is also modelled in the form of heat sources (e.g., `oven`, `stove`) and heat sinks (e.g., `fridge`, `freezer`) that transfer heat energy to or from objects. Rooms also transfer ambient heat energy to/from the objects they contain.

Electricity: The simulator models simple series electrical circuits, where electrically-powered devices (e.g., `light bulb`, `motor`) can be powered by being connected to electrical energy sources (e.g., `battery`, `solar panel`) through electrical conductors (nominally, `wires`). Polarized and unpolarized components are modelled, with each object having exactly two terminals (anode and cathode for polarized; terminals 1 and 2 for unpolarized). Connection happens through explicit terminal-to-terminal connection actions (e.g., `connect battery anode to blue wire terminal 1`). Every non-electrical object in SCIENCEWORLD has virtual unpolarized terminals, allowing circuits to be build with valid electrical conductors (e.g., using a `metal fork` in place of a `wire`), and for the agent to build circuits that test conductivity by (for example) observing if a light bulb illuminates when a plastic versus metal fork is used in the circuit.

Devices: Many objects are also considered devices, that can be activated or deactivated by the agent (e.g., `stove`, `electrical switch`), or may have environment-specific conditions to being activated (e.g., a `light bulb` will only activate if it is properly electrically connected; a `solar panel` will only produce power if it is outside). Objects can also be used with other objects in specific contexts (e.g., a `thermometer`, to measure an object's temperature; a `shovel`, to dig soil from the ground).

Chemistry: A subset of specific chemical reactions are modelled, where mixing a set of substances together in a container will produce a resultant substance (e.g., `salt` and `water` mix to produce `salt water`). Common chemical reactions taught in elementary science (e.g., water reactions, rust, food reactions, paint mixing) are modelled.

Life Stages: Living things (plants and animals) progress through life stages (e.g., seed, seedling, juvenile plant, adult plant, reproducing plant, dead plant). Progression through life stages happens over time by continuing to meet the needs of that living thing (e.g., `water`, `soil`). If the needs are not met (e.g., a plant is not watered, is removed from soil, or becomes too hot), then it dies.

Reproduction and Genetics: Living things can have genes that express traits (e.g., flower colour, seed shape, leaf size). Genes are inherited from the alleles of both parents, and genotype is determined at the time of reproduction using a Punnett square. Phenotype (expressed, visible traits) are determined based on which genes are dominant versus recessive. Currently, traits are only populated for selected plants to reproduce Mendelian-genetics experiments. Plants reproduce by exchanging pollen (containing their genes) between flowers, typically by a pollinator (such as a `bee`). Pollinated flowers eventually wilt and turn into fruits containing seeds of genetic descendants.

Friction (Inclined Plane): Forces are a significant part of an elementary science curriculum, but difficult to incorporate without 2D or 3D simulation. SCIENCEWORLD models the forces of gravity and friction in the specific context of 1-dimensional inclined plane experiments. Objects placed at the top of an inclined plane will slide down the plane at a speed proportional to the plane's angle, and the friction coefficient of its surface material. The position is described to the agent (e.g., *"an inclined plant with a block 60% of the way down the plane"*), allowing experiments to determine either the relative angle or friction coefficients of different inclined planes based on the speed the object moves down a given plane.

---

[3] For tractability, simulation engines are implemented with fidelity at the level of elementary science. Thermal transfer uses a simplified equation, biological changes happen in stages rather than gradually, only simple series circuits are simulated (no resistance, inductance, or any advanced topics), etc.

Containers: Containers can be always open (e.g., a `metal pot`) or closeable (e.g., a `cupboard`). Objects contained inside containers are not visible until the container is open. Some effects spread beyond a container – for example, a wooden cupboard with a hot object inside may combust, causing other objects in the kitchen to also increase temperature.

4 Experiments

To understand how contemporary approaches to text agents perform at SCIENCEWORLD tasks, we benchmark a selection of recent architectures.

Tasks. To support our goal of generating a diverse set of tasks, we identified a candidate set of 10 broad science exam topics from the list of 400 fine-grained science curriculum topics of Xu et al. (2020). Topics were chosen that would be amenable to text-based simulation, and that did not have critical fine-grained spatial reasoning requirements, and include: changes of state, temperature measurement, electrical circuits, friction, object classification, chemical mixtures, plants and pollinators, life spans, life stages, and Mendelian genetics. Each topic was further divided into between 2 and 4 specific tasks for agents to perform, producing a total of 30 science-domain tasks. These topics and tasks are described in Appendix B.2.

To prevent overfitting and encourage generalization, each subtask contains between 10 and 1400 parametric variations (with 7200 total variations across all 30 subtasks). Variations change critical task objects (e.g., the specific substance to be melted), the agent's starting location in the environment, as well as randomly vary the contents of the environment itself (e.g., whether the living room contains a bookshelf, or a painting, or both).

Train, Development, Test sets: For a given subtask, variations are split into 50% training, 25% development, and 25% test sets. Variations are sorted such that critical unseen variations (e.g., substances, animals, or plants unseen during training) are found in development and test sets.

Goals and Rewards. To reduce reward sparsity, each task includes between 2 and 15 optional subgoals (such as *turning on the stove*, or *the substance increasing in temperature by 10C*) that help nudge agents in the direction of canonical solutions, if desired. Meeting required and optional subgoals increases the agent's score on a given subtask. Scores for all tasks are normalized to between 0 and 1.

Oracle Agents To support imitation learning, we provide gold trajectories from 30 hand-coded oracles on all subtasks and variations. For tractability these solutions represent canonical solution methods (e.g., using a stove to boil water), rather than all possible solution methods that lead to the goal state (e.g., building a campfire to boil water).

Learning Agents. An interactive text environment can be cast as a partially observable Markov decision process (POMDP) defined by $\langle S, T, A, R, O, \Omega, \gamma \rangle$ representing the set of possible states (S), conditional transition probabilities between states (T), available text commands (A), reward function ($R : S \times A \rightarrow \mathbb{R}$), set of possible text observations (O), observation conditional probabilities ($\Omega : S \rightarrow O$), and discount factor ($\gamma \in [0, 1]$). The goal for a learning agent is then to learn a policy $\pi_\theta(o) \rightarrow a$ that chooses or generates a text command $a \in A$ given the text observation $o \in \Omega(s)$ of state $s \in S$ that maximizes the expected discounted sum of rewards $\mathbb{E}[\sum_t \gamma^t R(s_t, a_t)]$. In SCIENCEWORLD, the agent is also provided with a task description d.

To provide a fair comparison, all models were run using identical experiment configurations when possible. Reinforcement learning models were run with identical training regiments (8 environment threads at 100k steps per thread). Training episodes reset after meeting an end state (success or failure), or after reaching a maximum number of steps (we used 100 in all experiments). Additional model details are provided in Appendix C.

Random Baseline: At each time step t, this baseline randomly chooses an action a from the set of valid actions A_t obtained from the simulator.

DRRN (He et al., 2016): The Deep Reinforcement Relevance Network (DRRN) learns separate representations of the observation space and action space of an environment, then trains a policy that selects from A_t the action that is the most relevant given the current text observation o_t (which also includes the description of the current room o_t^{look} and the current agent inventory o_t^{inv}) and that would lead to an increased reward. The DRRN is a strong baseline with near state-of-the-art performance on many medium-to-hard interactive text environments (Hausknecht et al., 2020).

KG-A2C (Ammanabrolu and Hausknecht, 2020): This model represents the state space with a knowledge graph built dynamically from the text observations o_t using OpenIE triples (Angeli et al., 2015) such as *(`glass bottle`, `contains`, `water`)*, while the action space is represented using action templates with placeholders (e.g., `open` `OBJ`) obtained from SCIENCEWORLD. The model learns a policy that selects relevant action templates then populates them with objects from the knowledge graph.

CALM (GPT2) (Yao et al., 2020): We collect transcripts of expert demonstrations for the train variations of the tasks using the oracle agents, then use them to fine-tune a pre-trained language model (GPT-2, Radford et al. (2019)). At runtime, the language model is provided with the current observation o_t, last observation o_{t-1}, and last action a_{t-1}, then generates a shortlist of 30 possible actions to take. This shortlist serves as input to an RL model similar to the DRRN, which re-ranks the actions and chooses the next action to perform.

Behavior Cloning (Torabi et al., 2018): We follow the methodology of Ammanabrolu et al. (2021) in adapting the popular imitation learning method of behavior cloning from observations to text agents. We used the same transcripts of demonstrations as the CALM (GPT2) agent to extract 211,092 training examples with $(d, o_{t-1}, a_{t-1}, o_t)$ as inputs and a_t as targets. We fine-tune a transformer-based text-to-text model with a T5 architecture (Raffel et al., 2020) initialized with the weights of a Macaw (Tafjord and Clark, 2021) model designed to answer science questions.

At test time, the agent performs zero-shot inference online in the simulator by generating a fixed number of actions with beam search on the unseen test variations for each task. Despite training on a large number of demonstrations, the generated actions are often invalid or not useful—resulting in zero scores. Thus, we treat the beam search's output as a ranked-list and run the highest-ranked action appearing in A_t, similar to the language-model-to-valid-action aligner of Huang et al. (2022).

Text Decision Transformer: Inspired by the Decision Transformer (Chen et al., 2021), we create a novel text game agent that models the entire POMDP trajectory as a sequence and has the ability to potentially predict actions that maximize future long term expected reward. We again used the same transcripts of demonstrations as the two previous agents to extract 224,902 training examples with $(d, o_{t-1}, \hat{R}_{t-1}, a_{t-1}, o_t, \hat{R}_t)$ as inputs and a_t as targets. Here $\hat{R}$ is the returns-to-go (i.e., sum of future rewards) $\hat{R} = \sum_{t'=t}^{T} r_{t'}$ where $r_{t'}$ is the future reward obtained by the expert at step t'—enabling models to predict actions that maximize future expected rewards. The architecture, pre-training, parameter sizes, and test inference are otherwise similar to the behavior cloning agent.

Both the Behavior Cloning and Text Decision Transformer agents learn to perform SCIENCEWORLD tasks offline from demonstrations once pre-trained for scientific QA. They use the prevailing paradigm for achieving state-of-the-art on many language benchmarks (e.g., QA (Khashabi et al., 2020; Tafjord and Clark, 2021), language understanding (Raffel et al., 2020; Brown et al., 2020)).

Results. Performance for all agents across each SCIENCEWORLD task is shown in Table 2. Overall, these tasks are challenging for current models, with the best model (DRRN) achieving an average score of 0.17 across all 30 subtasks. Models that rely on the valid action detection aid generally perform better than those that learn to generate *valid* actions in addition to learning what actions to pick to increase task performance. All models relying on large language model for action selection (CALM, BC-T5, TDT-T5) generally achieve low performance as they tend to generate few valid actions in their candidate action lists.

Figure 2 shows episode reward curves for four selected tasks (with reward curves for all tasks included in Appendix C). These four tasks include the current best-performing task *(Task 4-2: Find a non-living thing)*, which requires an agent to focus on any non-living thing in the environment, pick it up, and place it in a specific container (typically in a different location than the agent's starting location). Most RL models quickly solve the majority of this task, but struggle with picking up and moving the object to the final container. In contrast, other more open-ended tasks (such as *Task 1-4*, which requires agents to perform any state-of-matter change to a specific substance) are performed poorly by all models. Finally, SCIENCEWORLD includes pairs of identical tasks where one can be solved by retrieving some critical component of the answer, while the other requires conducting the experimental procedure successfully. For example, in *Task 3-3*, an agent could look up that a `metal fork` is an electrical conductor and solve the task with comparatively fewer steps then in its paired *Task 3-4*, where the substance name is randomly generated (e.g., `unknown substance B`) and the experiment must be completed to get the answer. We do not yet

#	Topic	Task	Random-Valid	DRRN	KG-A2C	CALM	BC	TDT
Rely on Test Time Valid Action Detection Aid			✓	✓			✓	✓
1-1	Matter	Changes of State (Boiling)	0.00	0.03	0.00	0.00	0.00	0.00
1-2	Matter	Changes of State (Melting)	0.00	0.04	0.00	0.00	0.00	0.01
1-3	Matter	Changes of State (Freezing)	0.00	0.01	0.04	0.00	0.01	0.00
1-4	Matter	Changes of State (Any)	0.00	0.03	0.00	0.00	0.00	0.00
2-1	Measurement	Use Thermometer	0.00	0.10	0.06	0.01	0.04	0.04
2-2	Measurement	Measuring Boiling Point (known)	0.00	0.08	0.11	0.01	0.01	0.02
2-3	Measurement	Measuring Boiling Point (unknown)	0.00	0.06	0.04	0.01	0.01	0.02
3-1	Electricity	Create a circuit	0.01	0.13	0.07	0.05	0.03	0.07
3-2	Electricity	Renewable vs Non-renewable Energy	0.01	0.10	0.04	0.07	0.02	0.05
3-3	Electricity	Test Conductivity (known)	0.01	0.07	0.04	0.02	0.05	0.05
3-4	Electricity	Test Conductivity (unknown)	0.00	0.20	0.04	0.02	0.04	0.05
4-1	Classification	Find a living thing	0.03	0.26	0.18	0.10	0.29	0.16
4-2	Classification	Find a non-living thing	0.63	0.56	0.44	0.54	0.19	0.17
4-3	Classification	Find a plant	0.01	0.19	0.16	0.10	0.17	0.19
4-4	Classification	Find an animal	0.01	0.19	0.15	0.08	0.21	0.19
5-1	Biology	Grow a plant	0.07	0.09	0.06	0.02	0.08	0.03
5-2	Biology	Grow a fruit	0.02	0.16	0.11	0.04	0.03	0.05
6-1	Chemistry	Mixing (generic)	0.01	0.20	0.17	0.03	0.06	0.10
6-2	Chemistry	Mixing paints (secondary colours)	0.01	0.29	0.19	0.06	0.16	0.20
6-3	Chemistry	Mixing paints (tertiary colours)	0.00	0.11	0.04	0.03	0.05	0.07
7-1	Biology	Identify longest-lived animal	0.02	0.48	0.43	0.06	0.26	0.20
7-2	Biology	Identify shortest-lived animal	0.03	0.47	0.32	0.10	0.14	0.16
7-3	Biology	Identify longest-then-shortest-lived animal	0.01	0.31	0.23	0.04	0.02	0.20
8-1	Biology	Identify life stages (plant)	0.00	0.09	0.05	0.04	0.04	0.02
8-2	Biology	Identify life stages (animal)	0.00	0.10	0.10	0.00	0.02	0.07
9-1	Forces	Inclined Planes (determine angle)	0.01	0.13	0.04	0.00	0.05	0.04
9-2	Forces	Friction (known surfaces)	0.00	0.13	0.04	0.03	0.05	0.04
9-3	Forces	Friction (unknown surfaces)	0.01	0.13	0.04	0.02	0.04	0.04
10-1	Biology	Mendelian Genetics (known plants)	0.01	0.19	0.11	0.02	0.06	0.06
10-2	Biology	Mendelian Genetics (unknown plants)	0.01	0.17	0.11	0.02	0.13	0.05
Average			0.03	**0.17**	0.11	0.05	0.08	0.08
Param. Count $\times 10^6$			–	**1.5**	5.5	131*	11,000	11,000

Table 2: Zero-shot performance of the agents on test variations of across all tasks. All online RL-trained agent performances are averaged over 5 independent random seeds. Results across seeds tend to have low variance, with 80% of standard deviations below 0.05, and 95% of standard deviations below 0.10. Performance for RL agents is averaged over the last 10% of evaluation episodes, while T5 performance represents average task score across all test variations of a task.* signifies that the value of 131M parameters includes the number of the parameters of the pre-trained GPT-2 action generator model. Only 6.9 million policy parameters are updated in RL training.

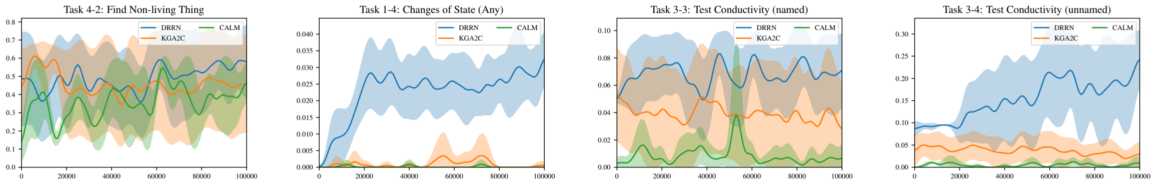


Figure 2: Episode reward curves for the DRRN, KGA2C, and CALM models on the unseen test set as a function of the number of training environment interactions. *(left)* An example of an easier task, where the agent must pick up any non-living thing, and place it in a specific box in the environment. *(center-left)* An example challenge task, where the agent must perform any change of state (melt, boil, freeze) on a specific substance. *(right)* Performance on two variations of a task where the agent must determine whether a specific substance is electrically conductive or not. In one variation *(center-right)*, the substance is named (e.g. metal fork), while in the other variation *(right)* the substance is randomly generated (e.g. unknown substance B).

observe this behavior with the agents under examination. They generally struggle with commonsense level tasks (e.g., navigation) and are unable to reach a point where the language models (either GPT-2 in CALM, or T5 initialized with Macaw in BC and TDT) are able to leverage their internal knowledge to solve these tasks through retrieval.

5 Discussion

Elementary science tasks are challenging for text agents. With top-performing agents reaching normalized average scores of 0.17 across tasks, performance on SCIENCEWORLD is comparable to the current best-performing agents on medium-difficulty interactive fiction games such as Zork

(Ammanabrolu et al., 2020; Yao et al., 2021). Much as in interactive fiction games, examining agent trajectories reveals that while agents appear to struggle with science-domain inference procedures such as *how to heat a substance* or *how to grow a seed*, they also currently lack a fluency with commonsense skills such as *navigating the environment* or *storing liquids in containers*. This underscores the need for models that can incorporate commonsense and science-domain knowledge, and integrate that declarative knowledge into actionable procedures to progress towards goals in the environment.

Larger models are not necessarily better. While larger models generally perform better in question answering tasks (e.g., Raffel et al., 2020), here we observe that larger models do not always increase performance. Our best-performing model, the DRRN, has only 1.5 million parameters – *four orders of magnitude* less than the T5 models. Both models also receive the same number of gradient updates (10^6) with respect to SCIENCEWORLD training tasks—though the T5 models have the added benefit of pre-training both from science exam QA and a large number of expert demonstrations.[4] This underscores that how a model approaches modeling state spaces and action sequences may be more important than the scope of its pre-training. Online, interactive training enables models such as the DRRN and KG-A2C to perform tasks requiring long action sequences more efficiently in terms of both samples and parameters.

Limitations of agents and environments. While agents still find text environments challenging, it is important to recognize that even this modest performance is achieved through a number of simplifying properties. For example, because agents frequently generate plausible but invalid actions, all but two agents benchmarked here depend on SCIENCEWORLD's valid action detection aid at test time, substantially simplifying their search problem in the action space. Similarly, while SCIENCEWORLD achieves a high environment fidelity for a text simulation, this is still tempered by pragmatic concerns, such as generating comparatively short descriptions of environments that can fit into the sequence lengths of most transformer models. As such, even environments with complex physical, chemical, and biological processes underlying their simulations (such as SCIENCEWORLD) still ultimately must limit the vividness of their descriptions, until these technical limitations in modelling can be surpassed. Hybrid environments (e.g., Shridhar et al., 2020) that concurrently model the same environment as both a high-fidelity 3D world and comparatively low-fidelity text-based simulation have shown that text environments can be used to provide useful task pre-training that can transfer back to the 3D environment with relatively low simulation compute cost.

Explanations as action sequences. Explanations take on a variety of roles (Lombrozo, 2006; Gilpin et al., 2018), from detailed human-readable descriptions of classification processes that describe *how* a decision was made (e.g., Ribeiro et al., 2016), to higher-level appeals to scientific processes to explain *why* an answer is correct (e.g., Jansen et al., 2018; Dalvi et al., 2021). Here, the action procedures generated by agents act as *manner* explanations for how to solve a particular task – but while they describe *how* to accomplish something, they don't explain at a high-level why those actions accomplish the task. For example, action sequences lack high-level goal information such as *"melting a substance requires heating it, so first the agent needs to heat the substance with a heating device, like a stove."*. Similar to how cuing agents to answer contextual questions can help improve their task performance (Peng et al., 2021), cuing agents to generate these explanatory scaffolds may help future agents increase task performance, while structuring their action sequence explanations for better human interpretability.

6 Conclusion

Despite recent progress in both interactive text agents and scientific text processing via transformers, current models are unable to reason about fundamental science concepts in a grounded and reusable manner—calling into question how much they are actually understanding the tasks at hand. To better measure such scientific reasoning abilities, we introduce SCIENCEWORLD, an interactive text environment derived from an elementary school science curriculum—with tasks ranging from electrical conductivity to Mendelian genetics.

We evaluate three state-of-the-art reinforcement learning text game agents: DRRN, KG-A2C, and CALM; and further introduce two large-scale transformer-based agents inspired by recent ad-

---

[4] See the APPENDIX for additional experiments evaluating performance versus model size.

vances such as Behavior Cloning and the Decision Transformer and trained for scientific reasoning in SCIENCEWORLD. While we find that overall performance on unseen tasks that require using science-domain knowledge is low across all agents, our results also suggest that agents that learn interactively in a grounded environment are more sample and parameter efficient than large language models that learn offline by reading text from static sources. The best agent performance is still modest — and on-par with medium-difficulty interactive fiction environments such as Zork — highlighting the need for agents that can integrate declarative scientific and world knowledge with procedural action sequences in virtual environments.

7 Broader Impacts

Interactive text environments can provide a faster and cheaper alternative to 3D environments to teach agents how to plan via sequential decision making. They allow better control over the level of abstraction desired to approach a task (i.e., `go to kitchen`, vs. `put hand on door's knob`, `turn knob clockwise`, `pull door`, `let go of the knob`, `walk through the door`). We believe making a plan in this abstract language space is simpler and more interpretable.

With respect to risks, we consider this current work as exploratory only. ScienceWorld is primarily intended for training agents to learn reasoning capabilities in the science domain, with limited immediate utility to human science students. Agents trained on ScienceWorld should not be used to provide advice for the real world. The environment in ScienceWorld has been made safer compared to the real world. For instance, the agent can't accidentally burn itself while boiling a substance on a campfire, and its actions should not be taken as demonstrations of safe procedures for students.

Acknowledgements

This work supported in part by National Science Foundation (NSF) award #1815948 to PJ, Google Cloud Compute, and the Allen Institute for AI. The authors would also like to thank Liwei Jiang, Jack Hessel, and Oyvind Tafjord for their very timely technical assistance and advice, giving us the ability to train our larger, transformer-based agents. We'd also like to thank Michal Guerquin for technical assistance with the project website.

References

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. 2021. Explanations for CommonsenseQA: New Dataset and Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3050–3065, Online. Association for Computational Linguistics.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*.

Prithviraj Ammanabrolu, Renee Jia, and Mark O Riedl. 2022. Situated dialogue learning through procedural environment generation. In *Association for Computational Linguistics (ACL) 2022*.

Prithviraj Ammanabrolu, Ethan Tien, Matthew Hausknecht, and Mark O Riedl. 2020. How to avoid being eaten by a grue: Structured exploration strategies for textual worlds. *arXiv preprint arXiv:2006.07409*.

Prithviraj Ammanabrolu, Jack Urbanek, Margaret Li, Arthur Szlam, Tim Rocktäschel, and Jason Weston. 2021. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 807–833, Online. Association for Computational Linguistics.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Mattia Atzeni, Shehzaad Dhuliawala, Keerthiram Murugesan, and Mrinmaya Sachan. 2021. Case-based reasoning for better generalization in text-adventure games. In *Proceedings of the International Conference on Learning Representations*.

Michael Boratko, Harshit Padigela, Divyendra Mikkilineni, Pritish Yuvraj, Rajarshi Das, Andrew McCallum, Maria Chang, Achille Fokoue-Nkoutche, Pavan Kapanipathi, Nicholas Mattei, Ryan Musa, Kartik Talamadupula, and Michael Witbrock. 2018. A systematic classification of knowledge, reasoning, and context within the ARC dataset. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 60–70, Melbourne, Australia. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Peter Clark, Oren Etzioni, Tushar Khot, Daniel Khashabi, Bhavana Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, et al. 2020. From 'f'to 'a'on the ny regents science exams: An overview of the aristo project. *AI Magazine*, 41(4):39–53.

Peter Clark, Philip Harrison, and Niranjan Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *AKBC '13*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910.

Ji He, Mari Ostendorf, Xiaodong He, Jianshu Chen, Jianfeng Gao, Lihong Li, and Li Deng. 2016. Deep reinforcement learning with a combinatorial action space for predicting popular Reddit threads. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1838–1848, Austin, Texas. Association for Computational Linguistics.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*.

Infocom. 1989. Learning zil.

Naoya Inoue, Pontus Stenetorp, and Kentaro Inui. 2020. R4C: A benchmark for evaluating RC systems to get the right answer for the right reason. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6740–6750, Online. Association for Computational Linguistics.

Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu, and Peter Clark. 2016. What's in an explanation? characterizing knowledge and inference requirements for elementary science exams. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2956–2965, Osaka, Japan. The COLING 2016 Organizing Committee.

Peter Jansen, Kelly J. Smith, Dan Moreno, and Huitzilin Ortiz. 2021. On the challenges of evaluating compositional explanations in multi-hop inference: Relevance, completeness, and expert ratings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7529–7542, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Peter Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton Morrison. 2018. WorldTree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Peter A Jansen. 2021. A systematic survey of text worlds as embodied natural language environments. *arXiv preprint arXiv:2107.04132*.

Harsh Jhamtani and Peter Clark. 2020. Learning to explain: Datasets and models for identifying valid reasoning chains in multihop question-answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 137–150, Online. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Matthew Lamm, Jennimaria Palomaki, Chris Alberti, Daniel Andor, Eunsol Choi, Livio Baldini Soares, and Michael Collins. 2021. QED: A framework and dataset for explanations in question answering. *Transactions of the Association for Computational Linguistics*, 9:790–806.

P David Lebling, Marc S Blank, and Timothy A Anderson. 1979. Zork: a computerized fantasy simulation game. *Computer*, 12(04):51–59.

Tania Lombrozo. 2006. The structure and function of explanations. *Trends in cognitive sciences*, 10(10):464–470.

Chris Martens. 2015. Ceptre: A language for modeling generative interactive systems. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 11.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2020a. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines. *arXiv preprint arXiv:2010.03790*.

Keerthiram Murugesan, Mattia Atzeni, Pushkar Shukla, Mrinmaya Sachan, Pavan Kapanipathi, and Kartik Talamadupula. 2020b. Enhancing text-based reinforcement learning agents with commonsense knowledge. *arXiv preprint arXiv:2005.00811*.

Graham Nelson. 2006. Natural language, semantic analysis, and interactive fiction. *IF Theory Reader*, 141:99–104.

Graham Nelson. 2014. The z-machine standards document version 1.1.

Xiangyu Peng, Mark O Riedl, and Prithviraj Ammanabrolu. 2021. Inherently explainable reinforcement learning in natural language. *arXiv preprint arXiv:2112.08907*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Oyvind Tafjord and Peter Clark. 2021. General-purpose question-answering with Macaw. *arXiv preprint arXiv:2106.01345*.

Ronen Tamari, Fan Bai, Alan Ritter, and Gabriel Stanovsky. 2021. Process-level representation of scientific protocols with interactive annotation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2190–2202, Online. Association for Computational Linguistics.

Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4950–4957. International Joint Conferences on Artificial Intelligence Organization.

Marco Valentino, Mokanarangan Thayaparan, and André Freitas. 2021. Unification-based reconstruction of multi-hop explanations for science questions. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 200–211, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter Jansen. 2020. WorldTree v2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5456–5473, Marseille, France. European Language Resources Association.

Dongfang Xu, Peter Jansen, Jaycie Martin, Zhengnan Xie, Vikas Yadav, Harish Tayyar Madabushi, Oyvind Tafjord, and Peter Clark. 2020. Multi-class hierarchical question classification for multiple choice science exams. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5370–5382, Marseille, France. European Language Resources Association.

Weiwen Xu, Yang Deng, Huihui Zhang, Deng Cai, and Wai Lam. 2021a. Exploiting reasoning chains for multi-hop science question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1143–1156, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Weiwen Xu, Huihui Zhang, Deng Cai, and Wai Lam. 2021b. Dynamic semantic graph construction and reasoning for explainable multi-hop science question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1044–1056, Online. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Karthik Narasimhan, and Matthew Hausknecht. 2021. Reading and acting while blindfolded: The need for semantics in text game agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3097–3102.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754, Online. Association for Computational Linguistics.

Xusen Yin and Jonathan May. 2019. Learn how to cook a new recipe in a new house: Using map familiarization, curriculum learning, and bandit feedback to learn families of text-based adventure games. *arXiv preprint arXiv:1908.04777*.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*.

A SCIENCEWORLD Description

SCIENCEWORLD is a simulation of the world abstracted through a complex interactive text environment with many objects (Sec. A.2), actions (Sec. A.3), and simulation engines (Sec. A.4). The object-oriented (Sec. A.1) simulator is written in SCALA and offers a PYTHON interface to interact with it. SCIENCEWORLD's flexibility makes it simple to create new science-domain tasks (Sec. B.2) and to evaluate the correctness of agents' solutions (App. B.4).

A.1 Object Model

Objects in SCIENCEWORLD are represented using an object-oriented model and are implemented as classes. SCIENCEWORLD objects can be thought of as collections of sets of properties (e.g., material properties, life properties, device properties, etc.). All objects implement common functions, such as those that produce textual descriptions of the object, or that provide one or more possible referents for the object based on its current state (e.g., the `water substance` in the solid state could generate the referents `ice`, `solid water`, and `substance`, each of which could be used by the agent to refer to that object in an action). Similar to Z-machine games (Infocom, 1989), objects are stored in an object tree representing the object's current container (its immediate parent object in the tree), and any objects it contains (child nodes in the tree).

A.2 Environment and Objects

SCIENCEWORLD is composed of a map of 10 locations centered around a house theme (`kitchen`, `bathroom`, `workshop`, `art studio`, `greenhouse`, `outside`, etc.), as shown in Figure 1. While the rooms and how they interconnect is static, the environment is randomly populated with different combinations of relevant contextual items each time it is initialized – for example, in one run, the living room may have a `bookcase` with three `books`. In other runs, the bookcase may have different books, no books, or not be present in the environment. This parametric variation discourages agents from memorizing the specific environment, and encourages robustness in task performance.

The environment is populated with up to 195 specific types of objects (excluding variations of those objects that change names or task properties, i.e., `red wires` and `black wires` belong to the same object type). This includes 23 animals, 11 plants, 25 substances, 10 canonical liquid containers (like `tin cups` or `glass jars`), 13 electrical components (such as `light bulbs`, `motors`, `wires`, and `generators`), 16 devices (including a `stove`, `thermometer`, and `stopwatch`), and 15 common pieces of furniture. To support these objects, the simulator includes a variety of other properties, including (for example) plant/animal life cycles, and 80 material properties (including water, glass, iron, and wood) that pure substances or physical objects (e.g., `tables`) can be made from.

Action	Description
`open/close OBJ`	open/close a container
`de/activate OBJ`	activate/deactivate a device
`connect OBJ to OBJ`	connect electrical components
`disconnect OBJ`	disconnect electrical components
`use OBJ [on OBJ]`	use a device/item
`look around`	describe the current room
`look at OBJ`	describe an object in detail
`look in OBJ`	describe a container's contents
`read OBJ`	read a note or book
`move OBJ to OBJ`	move an object to a container
`pick up OBJ`	move an object to the inventory
`put down OBJ`	drop an inventory item
`pour OBJ into OBJ`	pour a liquid into a container
`dunk OBJ into OBJ`	dunk a container into a liquid
`mix OBJ`	chemically mix a container
`go to LOC`	move to a new location
`teleport to LOC` *	teleport to a specific room
`eat OBJ`	eat a food
`flush OBJ`	flush a toilet
`focus on OBJ`	signal intent on a task object
`wait [DURATION]`	take no action for some duration
`task`	describe current task
`inventory`	list agent's inventory

Table 3: The 25 actions in the action space of SCIENCEWORLD. Actions can take up to two parameters, referencing objects the action should interact with. * signifies that the *teleport* action is only available to agents in a simplified mode.

A.3 Action Space

The simulator implements 25 actions, shown in Table 3, including generic actions common in interactive text environments (e.g., opening a door, moving to a location), as well as science-domain specific actions (e.g., connecting electrical components, chemically mixing items, pouring liquids). Five actions take two arguments, 16 take one argument, and four actions take zero arguments. Given the approximately 200 possible objects (excluding parametric variations) in SCIENCEWORLD, the action space can naively be estimated to be approximately 200,000 possible unique action possibilities at each step, though only a small subset of these would be meaningful. Similar to the Jericho framework, the simulator can provide valid action detection as an aid to agents (such as the DRRN) that require selecting their next action from a list of possible known-valid actions at a given step.

Input Parser At each step, an input parser attempts to parse user or agent input into a single unique action. Actions are specified as templates that can take on a variety of surface forms (e.g., `move to LOCATION` or `go to LOCATION`), and that include placeholders for object referents. At runtime, the parser examines all valid referents for visible objects from the agent's point of view, and if a given input string can produce more than one valid action, the parser will ask for clarification[5].

A.4 Simulation Engines

SCIENCEWORLD supports actions commonly found in interactive text environments – for example, objects can be moved or examined, foods can be eaten, and books can be read. In addition, the environment contains a number of elementary science-domain specific processes that either occur automatically (e.g., thermodynamics) or are coupled to actions (e.g., using devices, mixing chemicals). Those simulation engines are:[6]

Thermodynamics: All objects have temperatures and other thermal properties based on their materials. All objects within a container are considered in thermal contact with each other, and transfer heat energy using a simplified conductive heat model. The proportion of heat transferred between objects at each step is mediated by the object's *thermal conduction coefficient*, allowing thermal conductors (like metal pots) and insulators (like ceramics) to be modelled. Every material has phase transition points *(i.e., melting point, boiling point)* and *combustion points* populated based on the best-known or approximate physical values for those materials. Objects that move past these thresholds will change state of matter (e.g., from a solid to a liquid), or begin a combustion process that ultimately ends in the object turning to ash unless its fire is put out. Convective heat transfer is also modelled in the form of heat sources (e.g., `oven`, `stove`) and heat sinks (e.g., `fridge`, `freezer`) that transfer heat energy to or from objects. Rooms also transfer ambient heat energy to/from the objects they contain.

Electricity: The simulator models simple series electrical circuits, where electrically-powered devices (e.g., `light bulb`, `motor`) can be powered by being connected to electrical energy sources (e.g., `battery`, `solar panel`) through electrical conductors (nominally, `wires`). Polarized and unpolarized components are modelled, with each object having exactly two terminals (either an anode and cathode for polarized components, or terminals 1 and 2 for unpolarized components). Connection happens through explicit terminal-to-terminal connection actions (e.g., `connect battery anode to blue wire terminal 1`). Every non-electrical object in SCIENCEWORLD has virtual unpolarized terminals, allowing circuits to be build with valid electrical conductors (e.g., using a `metal fork` in place of a `wire`), and for the agent to build circuits that test conductivity by (for example) observing if a light bulb illuminates when a plastic versus metal fork is used in the circuit.

Devices: Many objects are also considered devices, that can be activated or deactivated by the agent (e.g., `stove`, `electrical switch`), or may have environment-specific conditions to being activated (e.g., a `light bulb` will only activate if it is properly electrically connected; a `solar panel` will only produce power if it is outside). Objects can also be used with other objects in specific contexts (e.g., a `thermometer`, to measure an object's temperature; a `shovel`, to dig soil from the ground).

Chemistry: A subset of specific chemical reactions are modelled, where mixing a set of substances together in a container will produce a resultant substance (e.g., `salt` and `water` mix to produce `salt water`). Common chemical reactions described in elementary science questions (e.g., water reactions, rust, food reactions, paint mixing) are modelled.

Life Stages: Living things (plants and animals) progress through life stages (e.g., seed, seedling, juvenile plant, adult plant, reproducing plant, dead plant). Progression through life stages happens over time by continuing to meet the needs of that

---

[5] For example, if the agent is in a room with two apples, one on a table and one in a bowl, the command `take apple` will cause the parser to ask the agent to clarify which apple they mean by selecting possible alternatives from a numbered list.

[6] To maintain tractability in implementation, simulation engines are implemented with a fidelity at the level of an elementary science curriculum. Thermal transfer uses a simplified equation, biological changes happen in stages rather than gradually, only series instead of arbitrary electrical circuits are simulated (and without the concepts of resistance, inductance or other advanced topics), etc.

living thing (e.g., `water`, `soil`). If the needs are not met (e.g., a plant is not watered, is removed from soil, or becomes too hot), then it dies.

Reproduction and Genetics: Living things can have genes that express traits (e.g., flower colour, seed shape, leaf size). Genes are inherited from the alleles of both parents, and genotype is determined at the time of reproduction using a Punnett square. Phenotype (expressed, visible traits) are determined based on which genes are dominant versus recessive. Currently, traits are only populated for selected plants to reproduce Mendelian-genetics experiments. Plants reproduce by exchanging pollen (containing their genes) between flowers, typically by a pollinator (such as a `bee`). Pollinated flowers eventually wilt and turn into fruits containing seeds of genetic descendants.

Friction (Inclined Plane): Forces are a significant part of an elementary science curriculum, but difficult to incorporate without 2D or 3D simulation. SCIENCEWORLD models the forces of gravity and friction in the specific context of 1-dimensional inclined plane experiments. Objects placed at the top of an inclined plane will slide down the plane at a speed proportional to the plane's angle, and the friction coefficient of its surface material. The position is described to the agent (e.g., *"an inclined plant with a block 60% of the way down the plane"*), allowing experiments to determine either the relative angle or friction coefficients of different inclined planes based on the speed the object moves down a given plane.

Containers: Containers can be always open (e.g., a `metal pot`) or closeable (e.g., a `cupboard`). Objects contained inside containers are not visible until the container is open. Some effects spread beyond a container – for example, a wooden cupboard with a hot object inside may combust, causing other objects in the kitchen to also increase temperature.

B Tasks and Competencies

To support our goal of generating a diverse set of tasks, we identified a candidate set of 10 broad science exam topics from the list of 400 fine-grained science curriculum topics of Xu et al. (2020). Topics were chosen that would be amenable to text-based simulation, and that did not have critical fine-grained spatial reasoning requirements. These topics and tasks are described in Section B.2.

Subtasks and Masked Objects: Each of the 10 broad curriculum topics is further subdivided into between 2 and 4 specific subtasks that test specific reasoning capacities (e.g., *melting, boiling, and freezing* subtasks for the change-of-state task), or ask the agent to perform the same task but with names of critical task objects masked. Some tasks are possible to partially solve by looking up critical task information (e.g., knowing that `white flowers` are a dominant trait of pea plants for the Mendelian genetics task). We include two versions of tasks, one with using masked names (e.g., growing `Unknown Plant B` instead of a `Pea Plant`) while simultaneously randomly generating the properties of those objects to provide an instrument to measure when agents are solving tasks by performing the experimental procedure, and when they are directly looking up answers.

Task Formats: Task goals are structured with the broad goal of both (a) accomplishing a task, and (b) doing so *intentionally*. Tasks typically include preliminary subgoals where the agent must signal their intent to perform the task on a specific object by first *"focusing"* on the object they intend to perform the task with (e.g., for a *boiling* task, focusing on water they intend to boil) before they perform the task.

Tasks take on two main forms: *Perform a task:* the agent must directly perform a task, that produces some measurable change in the environment (e.g., growing a fruit through pollination) that can be directly measured as an end-state. *Forced-choice:* The agent must perform a task that requires making an inference (e.g., whether an object is an electrical conductor or insulator), and provide their answer by placing the task object in a specific container (i.e., an *"answer box"*) if the object is conductive, and a different container if it is an insulator.

Task Variations: To prevent overfitting and encourage generalization, each subtask contains between 10 and 1400 parametric variations of that subtask (with 7200 total variations across all 30 subtasks). Variations change critical task objects (e.g., the specific substance to be melted), the agent's starting location in the environment, as well as randomly vary the contents of the environment itself (e.g., whether the living room contains a bookshelf, or a painting, or both).

Task Simplifications: Agents find different competencies that SCIENCEWORLD tests to be challenging. Tasks can be made easier by enabling any of 5 environment simplifications (or choosing "easy" mode, which enables all the simplifications). Examples of simplifications include a *teleport* action that lets agents instantly move to any location, and having all containers open by default.

B.1 Scoring and Evaluation Protocol

Goals and Reward Shaping: Each subtask contains a small number of method-agnostic required goals to be met (such as *focusing on the substance to melt*, and *causing that substance's state of matter to change from a solid to a liquid* for the melting task). In addition, to make rewards less sparse for agents learning these tasks, each task includes between 2 and 15 optional subgoals (such as *turning on the stove*, or *the substance increasing in temperature by 10C*) that help nudge agents in the direction of canonical solutions, if desired. Meeting required and optional subgoals increases the agent's score on a given subtask. Scores for all tasks are normalized to between 0 and 1.

B.2 Specific Tasks

Changes of State: The agent must find a named substance (e.g., `ice`), and change the state of matter of that substance (solid, liquid, gas) using the heating and cooling devices (e.g., `stove`, `freezer`) available in the environment. Subtasks require specific phase changes (melting, boiling, freezing, or the agent's choice). Variations change the substance, and ablate common devices (e.g., the stove becomes disabled) so that the agent must find alternate methods of heating or cooling.

Measurement Instrument: The agent must find a `thermometer` and use it to measure the temperature of a named object. In two additional subtasks, the agent must use the thermometer to measure the melting point of a named substance by heating it and continually monitoring the temperature. Answers are modelled as a forced-choice task, where the agent is given a predetermined temperature threshold at the start of the task (e.g., $50^{\circ}C$), and must focus on one answer box if the melting point is above the threshold, and the other answer box if the melting point is below the threshold. Variations change the substance to be measured, and the preset temperature threshold.

Electrical Circuits: The agent must build a working series electrical circuit by connecting various electrical components including power sources (e.g., `battery`, `wind mill`, `gas generator`), different coloured `wires`, and active components (e.g., `lights`, `motors`). Subtasks include (a) powering a named component, (b) powering using renewable versus nonrenewable energy, and (c) measuring whether named or unknown substances are electrically conductive. Variations change available components, colours of wire, and specific task objects required to be used in the circuit.

Classification: In four subtasks, the agent must find an object in the environment belonging to a specific category (living things, non-living things, plants, or animals), and place it in an answer box. Variations change the location and description of the answer box.

Growing plants: The agent must grow a named plant (e.g., a `peach tree`) from seed. To do this, they must place the `seed` and `soil` in a `flower pot`, and provide regular water as the plant progresses through life stages into adulthood. Failure to water appropriately causes the plant to perish. In a subtask, the agent must grow a fruit by growing several plants, then releasing pollinators (i.e., `bees`) that cross-pollinate flowers on the plants, which will eventually produce fruits. Variations change seed type, and soil location (either provided in the pot, provided in the room, or must be gathered outside using a shovel).

Chemistry: The agent must create a specific substance by mixing two or more input substances in a container. In the generic subtask, a recipe document that can be read by the agent is provided somewhere in the environment. In two paint-themed subtasks, the agent is given primary colours of paint (red, green, yellow), and must mix secondary (e.g., orange) or tertiary (e.g., orange-yellow) colours through several steps. Variations change the required output substance.

Life Spans: In three task variations, the agent must find 3 animals in the environment, then select either the shortest-lived (e.g., `bee`), longest-lived (e.g., `turtle`), or shortest-then-longest lived (bee-then-turtle). Variations change which animals are populated in the environment from a list of long, medium, and short-lived animals.

Life Stages: The agent must find a named plant or animal, and focus on its life stages, from earliest (e.g., seed) to latest (e.g., reproducing adult plant). Variations change the plant or animal involved.

Forces: The agent must use inclined planes and masses (e.g., a `block`) for experiments about forces. In one subtask the agent is given two planes, and must determine which has the steepest or shallowest angle based on the time the block takes to move down the plane. In two other subtasks, the agent must find which of two planes has a surface of highest or least friction, from either named (e.g., plastic, steel) or unknown surfaces. The agent can measure time internally (in terms of number of steps for a block to fall), or measure this explicitly with a provided stopwatch. Variations change inclined plane angles (first task) or surface material types (remaining tasks).

Mendelian Genetics: The agent must determine whether a named trait (e.g., white flower colour) is a dominant or recessive trait in a plant. Two seeds are provided (one with the trait as dominant, one recessive), and the agent must grow two generations of plants and count how often it observes a given trait in successive generations to determine whether it is dominant or recessive. Subtasks change whether the plant is known (the `pea plant`, as in Mendel's experiments) or a randomly generated plant, while variations change the trait under investigation.

B.3 Commonsense Competencies

In addition to science-domain competencies, the agent must demonstrate fluency with commonsense knowledge and procedures to complete tasks successfully. Agents must know the locations of common objects (e.g., `water` comes from a `sink`, `orange juice` is typically found in a `fridge`), the affordances of common objects (a `sink` can be turned on to create `water`, a `cup` can be used to carry a liquid), navigation (a world is made of discrete rooms that can be traversed through doors), containers need to be opened to observe or use their contents, and so forth.

B.4 Scoring and Evaluation Protocol

Goals and Reward Shaping: Each subtask contains a small number of method-agnostic required goals to be met (such as *focusing on the substance to melt*, and *causing that substance's state of matter to change from a solid to a liquid* for the melting task). In addition, to make rewards less sparse for agents learning these tasks, each task includes between 2 and 15 optional subgoals (such as *turning on the stove*, or *the substance increasing in temperature by 10C*) that help nudge agents in the direction of canonical solutions, if desired. Meeting required and optional subgoals increases the agent's score on a given subtask. Scores for all tasks are normalized to between 0 and 1.

Train, Development, Test sets: For a given subtask, variations are split into 50% training, 25% development, and 25% test sets. Variations are sorted such that critical unseen variations (e.g., substances, animals, or plants unseen during training) are found in development and test sets.

B.5 Task Simplifications

Agents find different competencies that SCIENCEWORLD tests to be challenging. Tasks can be made easier by enabling any of 5 environment simplifications (or choosing "easy" mode, which enables all the simplifications). Examples of simplifications include a *teleport* action that lets agents instantly move to any location; self-watering flowering flower pots that mean plants do not have to be frequently watered; and having all containers open by default.

C Experiment Details

C.1 Reinforcement Learning Models

For each reinforcement learning model, we ran 8 environment threads at 100k steps per thread. Training episodes reset after meeting an end state (success or failure), or after reaching 100 steps. For KG-A2C and CALM, the training episode will also reset if stuck by invalid actions for 100 steps (invalid actions are not counted by the environment). We did evaluation on the test variations every 1000 steps per environment thread. We randomly chose 10 test variations and run 1 episode of testing for each chosen variation during each evaluation period and reported the average score of the 10% test steps scores.

DRRN: We use the DRRN architecture from `https://github.com/microsoft/tdqn`, with embedding size and hidden size set as 128. The learning rate we use to train DRRN is 0.0001. The memory size is 100k, and priority fraction is 0.50. The tokenizer for the input text is a uni-gram subword tokenizer model adapted from Kudo (2018).

KG-A2C: We make two major changes to the original KG-A2C model to function with SCIENCEWORLD. (1) We replace the OpenIE knowledge graph extractor with a heuristic extractor. The heuristic extractor uses regular expressions to parse the text of the "look around" and "agent inventory" information into *(subject, relation, object)* triples. This heuristic functionally extracts the ground truth knowledge graph representation of the observable environment for the agent. (2) We change the KG-A2C agent to generate object *types* instead of references to specific objects. After selecting the action template and object type fillers that the agent has the highest confidence in, we ground those object types (e.g. apple) with specific object referents in the agent's current visible environment. If more than one referent meets that type, one is chosen at random. The learning rate we use to train the KG-A2C agent is 0.003 and the tokenizer used is the same as the DRRN agent.

CALM-GPT2: Following the original CALM paper (Yao et al., 2020), we use a 12-layer, 768-hidden, 12-head GPT-2 model. We use the default pre-trained weight offered by the Huggingface Transformers library (Wolf et al., 2020). We fine-tune this GPT-2 model on complete action sequences generated by the oracle agents. The GPT-2 input prompt is formed as "[CLS] d [SEP] o_t [SEP] o_t^{look} [SEP] o_t^{inv} [SEP] o_{t-1} [SEP] a_{t-1} [SEP]" and targets to predict a_t, where d stands for the task description and o_t, o_t^{look}, o_t^{inv}, and a_t stand for the observation (excluding the look around and inventory information), the output of a "look around" action at the agent's current location, the agent inventory, and the action at time step t. We use beam search for generation, generating 16 beams representing candidate actions for the agent to select from. We set the diversity penalty to 50.0 to encourage the GPT-2 model to generate different actions. For the GPT-2 training we use a batch size of 12 and train for 20 epochs with a learning rate of 0.00002. The learning rate we use to train the CALM agent is 0.0001 and the input tokenizer is the same as that used in the original GPT-2 paper.

Due to the high computation cost of the CALM model, and modest overall performance, performance for each task is the average of only 3 random seeds instead of the 5 used for training the DRRN and KGA2C models. During development, a small error was found in the prompt. Pilot experiments suggested this resulted in a negligible (±0.01) change in performance, so the full experiments (requiring up to 6000 GPU hours) were not regenerated.

Episode reward curves: Episode reward curves for each model across all 30 subtasks in SCIENCEWORLD are shown in Figure 3.

C.2 Behavior Cloning and Text Decision Transformer

The T5 models used to train both of these models are initialized with weights and tokenizers from the trained Macaw-11b model released at `https://github.com/allenai/macaw`. They are trained on a v3-32 TPU pod with a batch size of 16 and 32-way model parallelism for 100k gradient update steps.

At inference time, we use the model to generate actions given the observation of current and previous step. We use beam search with a beam size 16 to get the top 16 generations. We set the diversity penalty to 50.0 to encourage diversity in generation. For each subtask, we run one episode on each of its test variations and report the average score. We do not update weights of the T5 model during evaluation.

C.3 Resources

Model	**GPU**	**Runtime**
Online RL Models		
DRRN	4gb	12h
KG-A2C	16gb	20h
CALM-GPT2	16gb	40h
Offline Transformer Models		
Model Pre-training	v3-32 TPU Pod	60h
BC-T5	3x 48gb	2h
TDT-T5	3x 48gb	2h

Table 4: Approximate computational resources per model. Online RL and Offline Transformer model reflect runtimes for one full run of one subtask at one seed. Runtime estimates should be multiplied by the number of tasks (30) and number of random seeds (5) to obtain full runtime estimates. All GPU-based runs were completed on P100, V100. or A6000 GPUs.

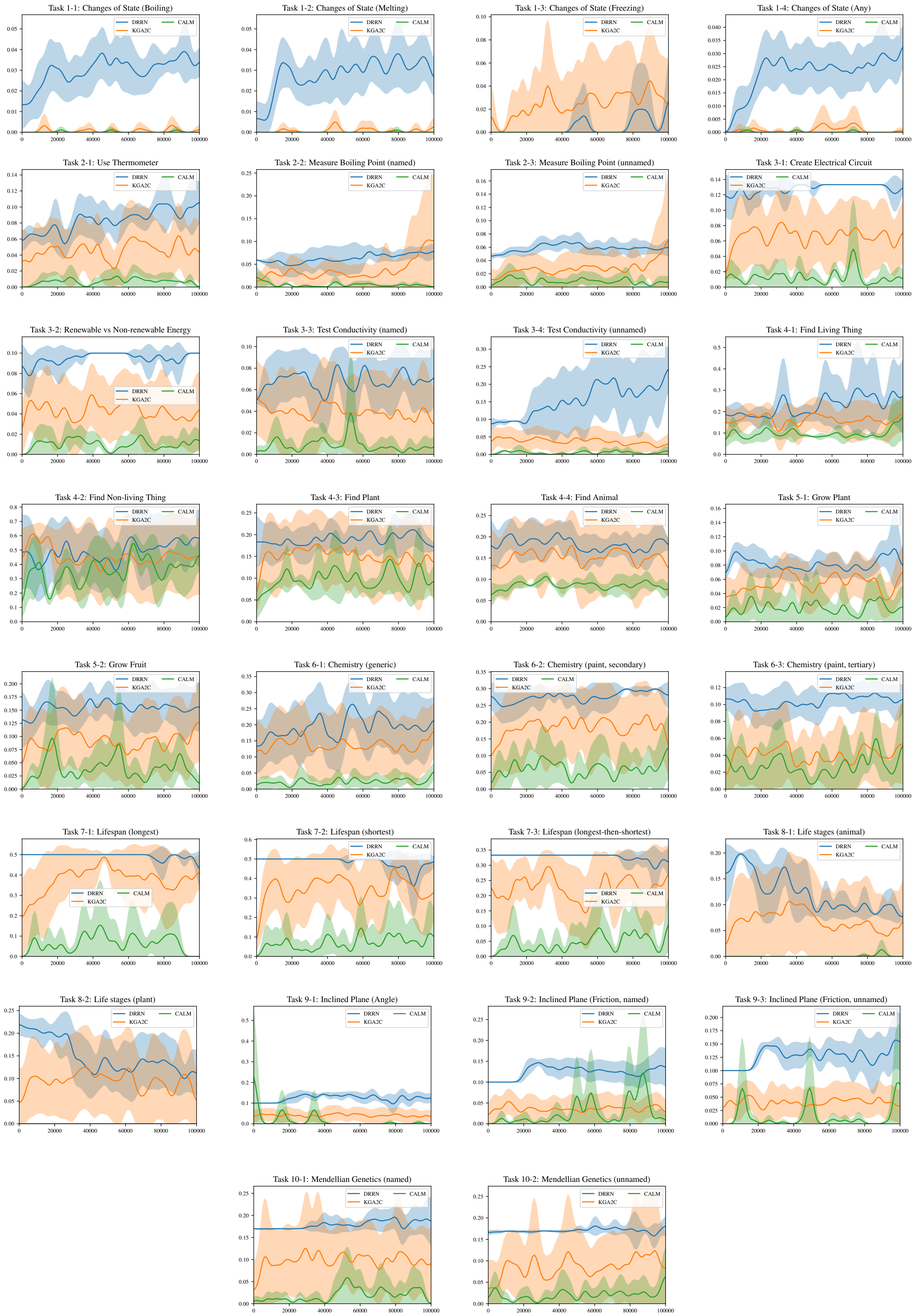


Figure 3: Episode reward curves for the DRRN, KGA2C, and CALM models on the unseen test set as a function of the number of training environment interactions (steps).

C.4 Impact of Model Size and Pre-training Methodology on Performance

Model	Average Performance	Model Parameters
DRRN	0.17	1.5M
KGA2C	0.11	5.5M
CALM	0.05	131M*
Behavior Cloned		
T5-Large	0.15	770M
Macaw-Large	0.17	770M
Macaw-11B	0.08	11B
Decision Transformer		
T5-Large	0.13	770M
Macaw-Large	0.15	770M
Macaw-11B	0.08	11B

Table 5: Average model performance versus model parameter size across all tasks and random seeds. For SCIENCEWORLD tasks, larger models do not necessarily perform better, and in some cases appear to show inverse scaling. * signifies that the value of 131M parameters includes the number of the parameters of the pre-trained GPT-2 action generator model. Only 6.9 million policy parameters are updated in RL training.

To examine the effect of model size on behavior cloned and decision transformer model performance, we ran two model sizes for the T5 models, shown in Table 5. We first observe that pre-training specifically for scientific question answering on a curated dataset (Macaw) outperforms T5 general language model pre-training. Further, we note that T5-Large and Macaw-Large, with 14 times fewer parameters (770m each), out-perform the larger 11 billion parameter models by approximately a factor of two. These results suggest that while SCIENCEWORLD can benefit from external scientific knowledge, it may also present an inverse scaling problem[7], where increasing model size can sometimes decrease overall task performance. However, these results are only suggestive of an inverse scaling problem rather than a concrete demonstration. Due to the high cost of training and inference for these models, we can't currently rule out that these differences may be due to hyperparameter differences. We leave this verification as future work.

C.5 Attribution

Graphical visualization makes use of RPG game assets developed by @Noiracide.

---

[7] `https://github.com/inverse-scaling/prize`